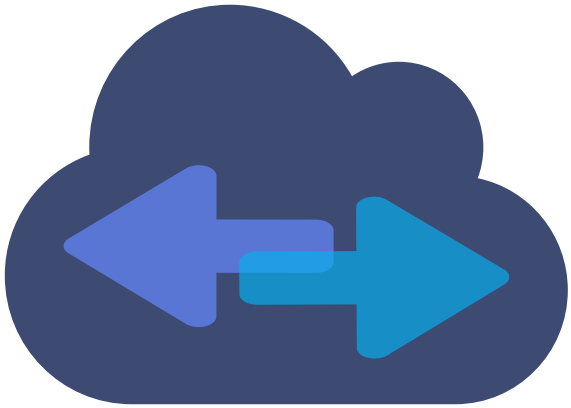


Virtual Accelerator for Ubuntu 18.04 on Intel

Release 3.2.5

6WIND

Sep 22, 2021

Contents

Thank you for choosing 6WIND Virtual Accelerator.

This document aims at explaining the basics of Virtual Accelerator and the technology behind it, how to install it, manage it and troubleshoot it. It provides examples of simple Virtual Accelerator use cases that can be used as reference to help you design and deploy your own use cases.

1. Getting Started

1.1 Introduction

1.1.1 Overview

Virtual Accelerator provides packet processing acceleration for virtual network infrastructures.

Virtual Accelerator runs inside the hypervisor and removes the performance bottlenecks by offloading virtual switching from the networking stack. The CPU (Central Processing Unit) resources necessary for packet processing are drastically reduced, so that less cores are required to process network traffic at higher rates and Linux stability is increased.

In addition to simple virtual switching (using OVS (Open vSwitch) or the Linux bridge), Virtual Accelerator supports an extensive set of networking protocols to provide a complete virtual networking infrastructure.

Virtual Accelerator is fully integrated with Linux and its environment, so that existing Linux applications do not need to be modified to benefit from packet processing acceleration.

Virtual Accelerator is available for Intel x86 servers, supports Red Hat Enterprise Linux, CentOS and Ubuntu distributions and OpenStack.

1.1.2 Features

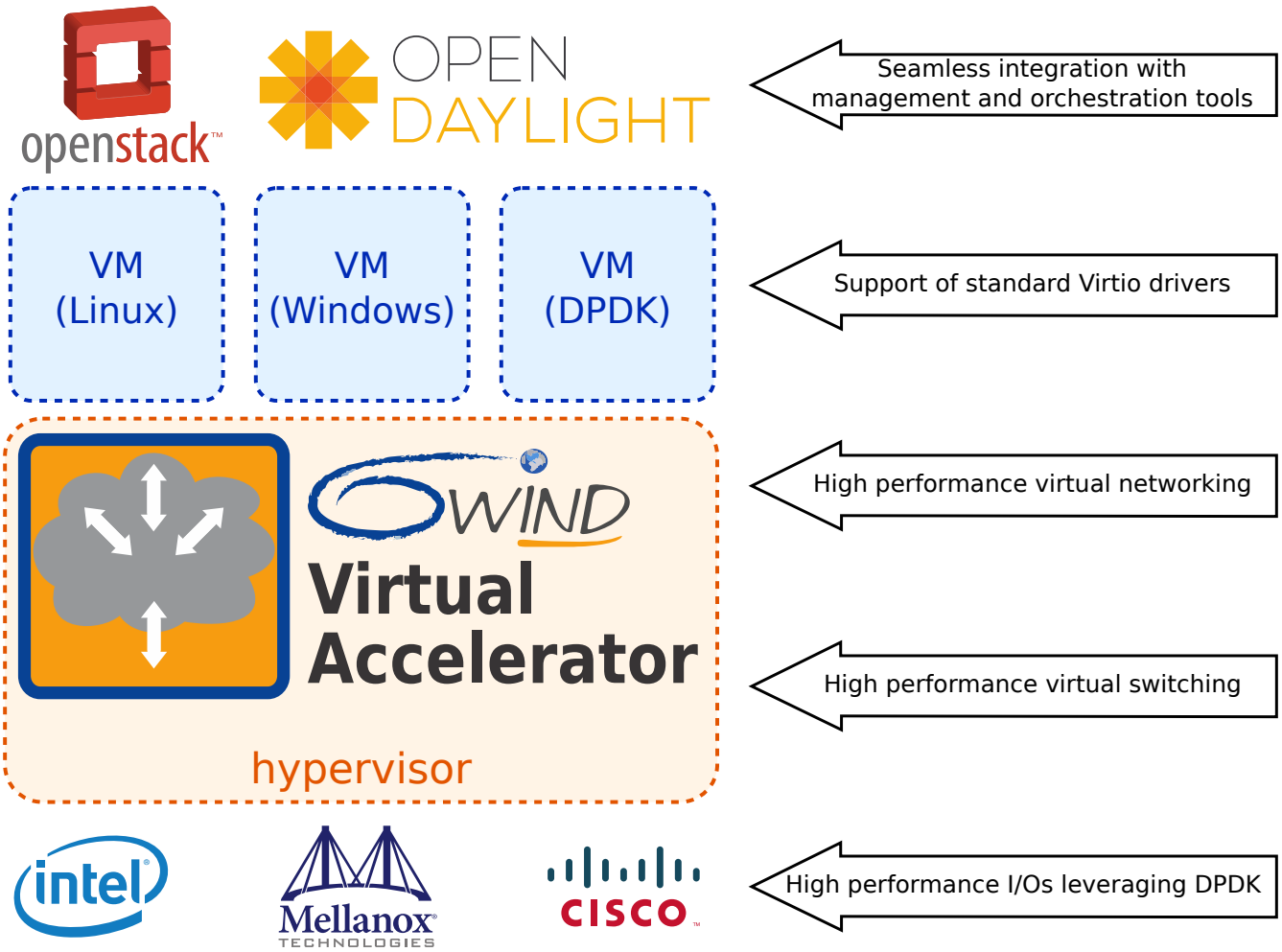


Fig. 1: Virtual Accelerator Features

High performance I/Os leveraging DPDK (Data Plane Development Kit), with multi-vendor NIC (Network Interface Card) support from Intel, Mellanox and Cisco

- Intel 1G 82575, 82576, 82580, I210, I211, I350, I354 (igb)
- Intel 10G 82598, 82599, X520, X540 (ixgbe)
- Intel 10G/40G X710, XL710, XXV710 (i40e)
- Mellanox 10G/40G Connect-X 3 (mlx4)
- Mellanox 10G/25G/40G/50G/100G Connect-X 4/5 (mlx5)
- Broadcom NetExtreme E-Series (bnxt)

High performance virtual switching

- OVS acceleration
- Linux Bridge

High performance virtual networking

In addition to virtual switching, Virtual Accelerator supports a complete set of networking protocols, based on the 6WINDGate technology, that can be used to design innovative virtual networking infrastructures.

- Layer 2 (Data Link Layer) and Encapsulations
  - GRE (Generic Routing Encapsulation)
  - VLAN (Virtual Local Area Network) (802.1Q, QinQ)
  - VXLAN (Virtual eXtensible Local Area Network)
  - LAG (Link Aggregation) (802.3ad, LACP)
  - MPLS
- IP networking
  - IPv4 and IPv6
  - IPv6 Autoconfiguration
  - VRF (Virtual Routing and Forwarding)
  - IPv4 and IPv6 Tunneling
  - NAT (Network Address Translation)
- Routing
  - BGP (Border Gateway Protocol), BGP4+
  - OSPF (Open Shortest Path First)v2, OSPFv3
  - RIP (Routing Information Protocol), RIPng (Routing Information Protocol next generation)
  - Cross-VRF (Cross Virtual Routing and Forwarding)
  - Static Routes
  - Path monitoring
  - ECMP (Equal Cost Multi Path)
  - PBR (Policy-Based Routing)
  - BFD (Bidirectional Forwarding Detection)
  - MPLS (Multiprotocol Label Switching) LDP (Label Distribution Protocol) (beta)

  - BGP L3VPN (Layer 3 Virtual Private Network) (beta)
  - VXLAN EVPN (Ethernet Virtual Private Network) (beta)
  - Point to Multipoint GRE interfaces
  - NHRP (Next Hop Routing Protocol)
  - DMVPN (Dynamic Multipoint VPN) with IPsec (Internet Protocol Security)
- IPsec[1]
  - IKE (Internet Key Exchange)v1, IKEv2 Pre-shared Keys or X509 Certificates
  - MOBIKE
  - Encryption: 3DES, AES-CBC/GCM (128, 192, 256)
  - Hash: MD-5, SHA-1, SHA-2 (256, 384, 512), AES-XCBC (128)
  - Key Management: RSA, DH MODP groups 1 (768 bits), 2 (1024 bits), 5 (1536 bits) and 14 (2048 bits), DH PFS
  - High performance (AES-NI, QAT)
  - Tunnel, Transport or BEET mode
  - SVTI (Secure Virtual Tunnel Interface), DVTI
- QoS
  - Rate limiting per interface, per VRF
  - Class-based QoS
    * Classification: ToS / IP / DSCP / CoS
    * Shaping and Policing
    * Scheduling: PQ, PB-DWRR
- Security
  - Access Control Lists
  - Unicast Reverse Path Forwarding
  - Control Plane Protection
  - BGP Flowspec
- High Availability
  - VRRP (Virtual Router Redundancy Protocol)

[1] requires a Virtual Accelerator IPsec Application License

Support of standard Virtio drivers

Virtual Accelerator comes with a high performance Virtio back-end driver for communication with any guest running Virtio front-end drivers (can be based on DPDK, Linux, or other OSes).

Management/Monitoring

- SSH (Secured SHell)v2
- SNMP
- KPIs (Key Performance Indicators) / Telemetry (YANG-based)
- Role-Based Access Control with AAA (TACACS)
- Syslog
- 802.1ab LLDP (Link Layer Discovery Protocol)
- sFlow
- Seamless integration with management and orchestration tools

  Virtual Accelerator is fully integrated with Linux and its environment, so that existing Linux applications do not need to be modified to benefit from packet processing acceleration. Standard Linux APIs are preserved, including iproute2, iptables, brctl, ovs-ofctl, ovs-vsctl, etc.

1.1.3 Supported platforms

- Virtual Accelerator is provided as a set of binary packages and is validated on the following distributions:
  - Ubuntu 18.04 for x86
  - Red Hat 8 for x86
  - CentOS 8 for x86

See also:

Refer to the Virtual Accelerator Release Notes for detailed information about the latest validated versions of the Linux distributions.

- Supported processors
  - Intel Xeon E5-1600/2600/4600 v2 family (Ivy Bridge EP)
  - Intel Xeon E5-1600/2600/4600 v3 family (Haswell EP)
  - Intel Xeon E5-1600/2600/4600 v4 family (Broadwell EP)
  - Intel Xeon E7-2800/4800 v2 family (Ivy Bridge EX)
  - Intel Xeon E7-2800/4800 v3 family (Haswell EX)

  - Intel Xeon E7-4800/8800 v4 family (Broadwell)
  - Intel Xeon Platinum/Gold/Silver/Bronze family (Skylake)
  - Intel Atom C3000 family (Denverton)
  - Intel Xeon D family
- Supported OpenStack Distributions: Ubuntu Cloud and RDO.

1.1.4 Delivery contents

Virtual Accelerator is provided as a .tgz archive file organized as follows:

software/ Virtual Accelerator binary software for your Linux distribution (.deb or .rpm packages)

doc/

- Virtual Accelerator Getting Started Guide

  This document
- Virtual Accelerator Publicly Available Software List

  List of Publicly Available Software included in the Virtual Accelerator delivery
- 6WINDGate documentation

  Detailed tgz package containing documentation of the 6WINDGate modules that compose the Virtual Accelerator product. Extract the package and open index.html in your browser.

  The mapping between Virtual Accelerator features and the corresponding 6WINDGate modules is detailed in the *Technology* section of this document.

An OpenStack application note is available, delivered separately, named `6wind-app-note-openstack-support-vX.Y.Z-ubuntu-18.04.tar.gz`.

1.1.5 Technology

Virtual Accelerator sits inside a KVM hypervisor and offloads packet processing from the Linux networking stack, thanks to the 6WINDGate technology.

This section provides notions about the 6WINDGate architecture that are necessary to understand how Virtual Accelerator works. More details about 6WINDGate can be found in the detailed documentation of each 6WINDGate module.

Data Plane

The Virtual Accelerator data plane is responsible for actual virtual switching and networking functionalities. It runs on a dedicated set of cores, isolated from the rest of the Linux operating system, and processes all incoming packets from NICs (Network Interface Cards) or vNICs (virtual Network Interface Cards). The Virtual Accelerator data plane relies on the DPDK, the 6WINDGate FPN-SDK and the 6WINDGate fast path.

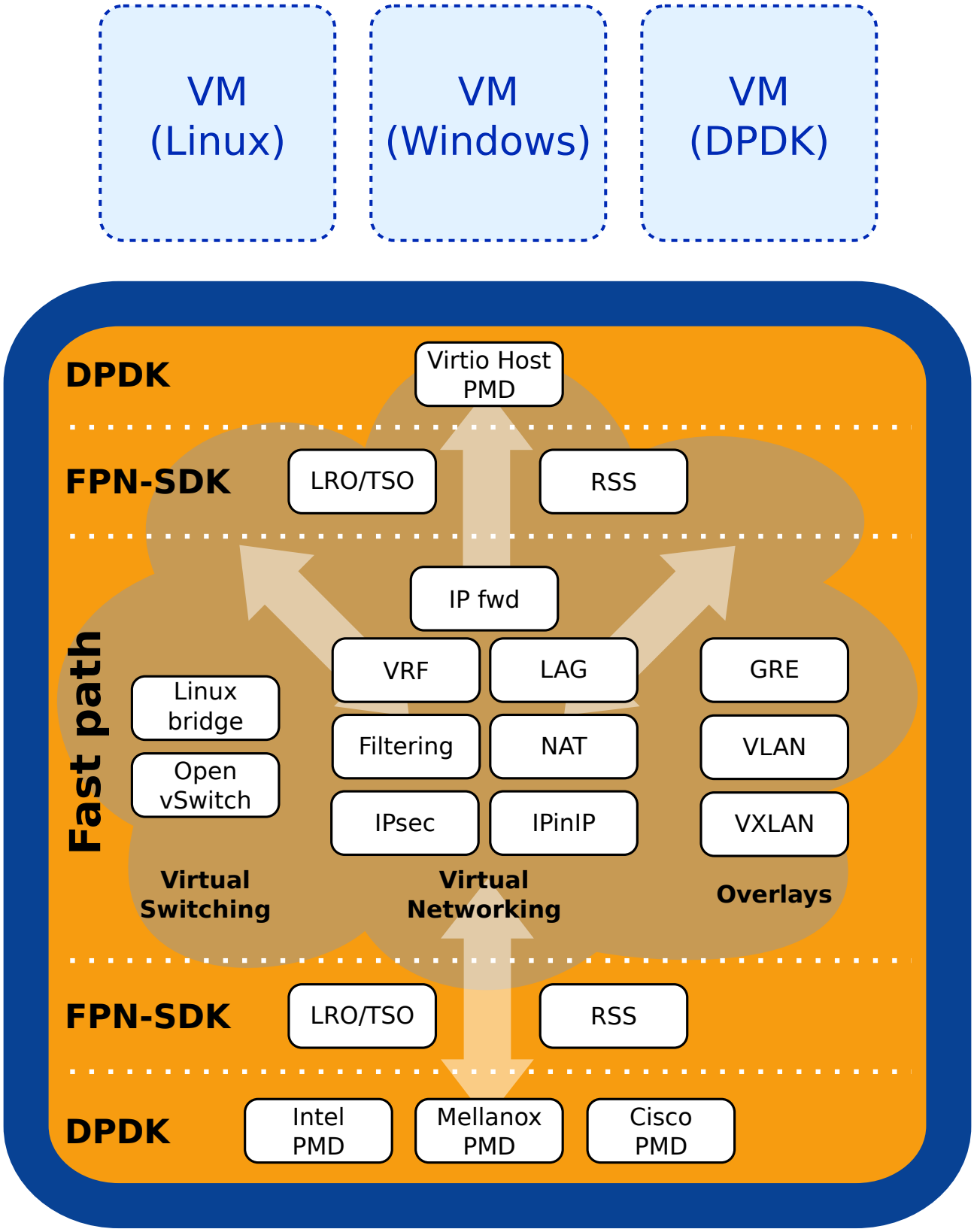


Fig. 2: Virtual Accelerator data plane architecture

The DPDK ensures high performance I/Os and is provided by the *6WINDGate DPDK* module. It includes drivers for miscellaneous NICs, called PMDs (Poll Mode Drivers). NIC support is included as part of *6WINDGate DPDK* itself, or as separate add-ons.

On top of the DPDK, the FPN-SDK provides a hardware abstraction layer implementing low level system features, such as offloads (checksum, LRO (Large Receive Offload)/TSO (TCP Segmentation Offload)), RSS (Receive-Side Scaling), handling of communication between Linux and the fast path, management of memory pools and rings, etc.

Finally, the fast path provides the high performance networking protocols for virtual switching and networking, including Linux bridge, OVS, IP (Internet Protocol) forwarding, filtering, etc.

The Virtual Accelerator data plane does not require any specific configuration as it is tranparently synchronized with the Linux data plane as described in the next section.

For more information regarding:	Refer to the documentation of:
Usage of the DPDK drivers for physical NICs	• *6WINDGate DPDK Mellanox ConnectX-3 EN series PMD* • *6WINDGate DPDK Mellanox ConnectX-4 EN series PMD*
Usage of the DPDK crypto libraries[Page 5, 1]	• *6WINDGate DPDK Intel Multi-Buffer Crypto* • *6WINDGate DPDK Intel Quickassist Crypto*
Usage of system monitoring, configuration and debug tools	*FPN-SDK Baseline*
Fast Path configuration file reference, including fast path core allocation, port/core binding, etc.	*FPN-SDK Add-on for DPDK*
Fast Path management	*Fast Path Baseline*
How to configure virtual switching and networking protocols (protocols are implemented in fast path modules, with one document per module).	The corresponding fast path modules

Linux - Fast Path Synchronization

To provide transparency with Linux, the 6WINDGate technology implements a continuous and transparent synchronization mechanism, so that all Linux configuration is synchronized into the fast path.

This is provided by the 6WINDGate *Linux - Fast Path Synchronization* module.

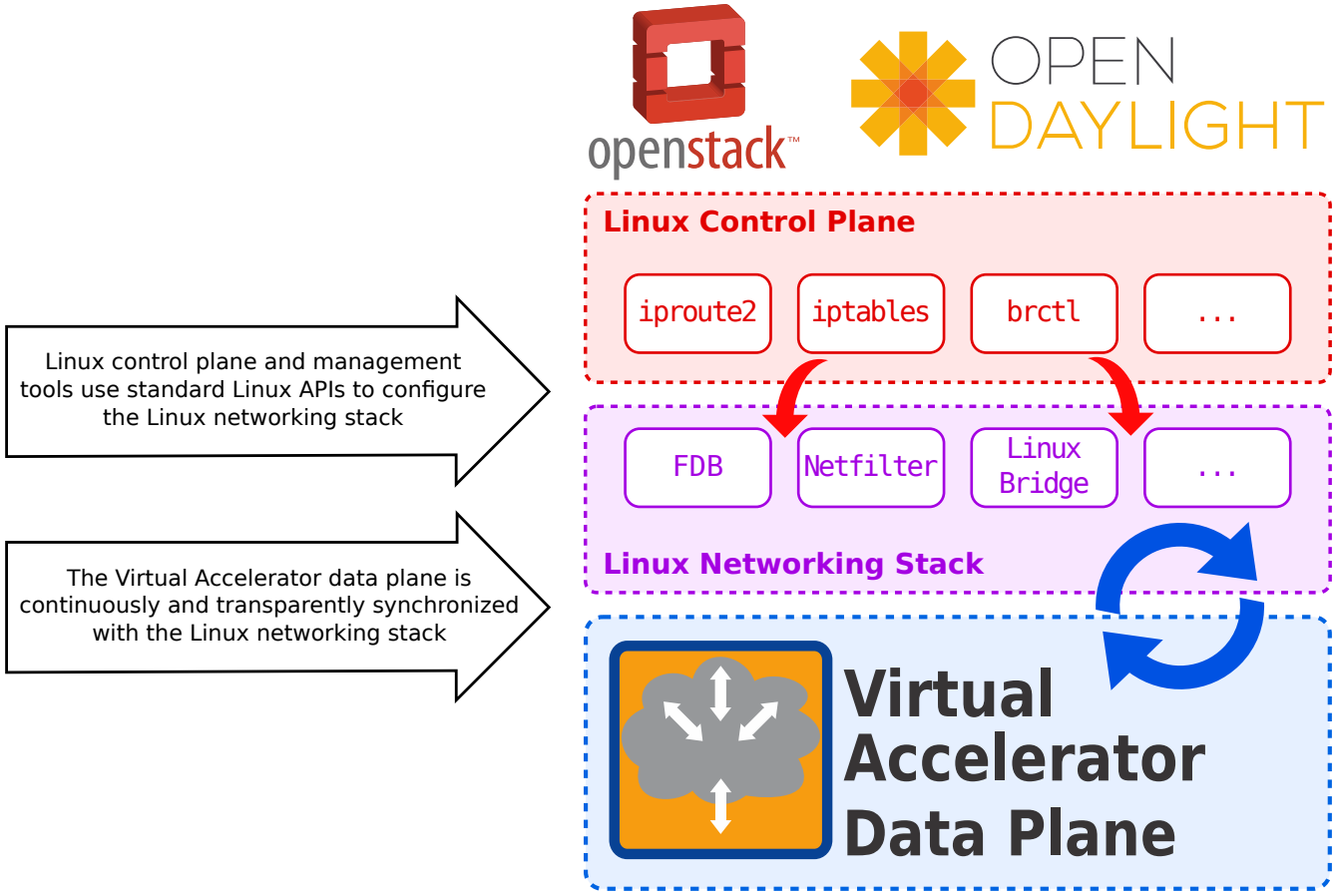


Fig. 3: Virtual Accelerator transparency

For more information regarding:	Refer to the documentation of:
Management of transparency with Linux	*Linux - Fast Path Synchronization*
How to configure VRF in Linux	*Linux - Fast Path Synchronization - VRF*

Control Plane

Virtual Accelerator provides its own control plane components, maintained and supported by 6WIND:

- *Control Plane OVS* control plane from openvswitch.org, including 6WIND's modifications for synchronization of Linux and fast path
- *Control Plane Security - IKEv1 and IKEv2*[Page 5, 1] control plane from strongswan.org, including 6WIND's extensions for support of VRF

These components are maintained and supported by 6WIND. The rest of the Virtual Accelerator data plane can be configured thanks to standard Linux tools, such as iproute2, iptables, brctl, etc. These are supported and maintained by the Linux distribution vendor.

For more information regarding:	Refer to the documentation of:
How to configure the Open vSwitch control plane	*Control Plane OVS*
How to configure the IKE control plane[Page 5, 1]	*Control Plane Security - IKEv1 and IKEv2*

vNICs

Communication with guests is provided through the *Virtio Host PMD* module, which is a DPDK add-on module providing a Virtio backend for guests.

The following offloads are advertised through the *Virtio Host PMD* so that Virtio guests can leverage them:

- Checksum offload (IP and TCP/UDP)
- LRO (based on GRO (Generic Receive Offload))
- TSO (based on GSO (Generic Segmentation Offload))
- Any of these offloads above can be leveraged inside tunnels (VLAN, VXLAN, GRE, IPinIP).

Offloads leverage hardware when possible (both the hardware and PMDs must support them); else, offloads are performed in software.

For more information regarding:	Refer to the documentation of:
Usage of the DPDK drivers for vNICs	*6WINDGate DPDK Virtio Host PMD*
Fast Path configuration file reference, including vNIC (virtual Network Interface Card) options and offloads	*FPN-SDK Add-on for DPDK*

1.2 Installation

1.2.1 Prerequisites

QEMU and libvirt

Virtual Accelerator works with the qemu and libvirt included in Ubuntu 18.04.

1. Install the following packages:

```
# apt-get install -y libvirt-bin python-libvirt qemu qemu-system-x86
```

2. Start libvirt:

```
# systemctl start libvirt-bin
```

3. Remove the libvirt default network:

```
# virsh net-destroy default
# virsh net-undefine default
```

Other prerequisites

Mellanox ConnectX-3 EN series and Mellanox ConnectX-4 EN series NICs

Drivers for these NICs require up-to-date firmware versions.

These are available for download from the Mellanox website (http://www.mellanox.com/), either using the mlxup (http://www.mellanox.com/page/mlxup_firmware_tool) tool, or through the latest Mellanox OFED (http://www.mellanox.com/page/products_dyn?product_family=26&mtag=linux_sw_drivers) for your Linux distribution.

Mellanox does not provide firmware releases for all OEM products (http://www.mellanox.com/page/oem_firmware_download), these are distributed by OEMs directly.

Note: In order to avoid installation conflicts, make sure *Mellanox OFED* itself is uninstalled before installing Virtual Accelerator packages.

1.2.2 Linux Requirements

The initial Linux requirements to use Virtual Accelerator is to have TC + EBPF support, a feature introduced in Linux 4.9. Depending on the function in use, additional bug fixes and improvements may be required. They are listed in the tables below.

Features

Table 1: Linux features patches

Functions	Description	Linux	Ubuntu HWE	Red Hat 8 CentOS 8	Link
Core	Flag reflects NIC carrier	5.0	5.0.0-23	4.18.0-80.18.el8	https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=26d31925cd5e
IPsec	IPsec output delegation using netfilter	4.20	Ubuntu 5.0	4.18.0-152.el8	https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=02b408fae3d5 https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=6c47260250fc
QoS	Mark transfer from kernel to fp	5.2	Ubuntu 5.3	4.18.0-193.el8	https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=58dfc900faff
Container	New capability CAP_BPF	5.8			https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=a17b53c4a4b5
Container	Netns: option to inherit sysctl from current netns	5.8			https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=9efd6a3cecdd

Known bug fixes

The following patches have been identified to be required.

Table 2: Linux bug fixes patches

Functions	Description	Linux	Ubuntu HWE	Red Hat 8 CentOS 8	Link
Core	Fix tun/tap async mode	4.17	4.15.0-65.74	4.18.0-80.el8	https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=2f3ab6221e4c
IPsec	XFRM interface support fixes	5.3, 4.19.89	5.0.0-32.34	4.18.0-88.el8	https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=f203b76d7809 https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=56c5ee1a5823 https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=e9e7e85d75f3 https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=e0aaa332e6a9 https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=c5d1030f2300 https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=22d6552f827e
IPv6	Packet loss during NDP resolution due to IPv6 socket API bug	5.2, 4.19.50	5.0.0-31.33, 4.15.0-66.75	4.18.0-134.el8	https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=2c6b55f45d53
IPsec VTI	xfrm interface: fix packet tx through bpf_redirect()	5.5, 5.4.18, 4.19.102	5.4.0-14.17	5.5.1-1.el8 4.18.0-193.1.2.el8	https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=f042365dbffea98fb8148c98c700402e8d099f02

1.2.3 Packages installation

Virtual Accelerator installation consists in setting up a local repository and then performing the actual package installation.

Setting up a local repository

1. Retrieve and extract the Virtual Accelerator `bin` package

```
# tar xfz /path/to/6wind-virtual-accelerator-*-bin.tgz
```

2. Set a variable for the directory containing the Virtual Accelerator software:

```
# export DELIVERY_DIR=/path/to/6wind-virtual-accelerator-*-bin
```

   In the rest of this document, the directory containing the Virtual Accelerator software will be referred to as *$DELIVERY_DIR*.

3. Install the local 6WIND *apt* repository:

```
# mkdir -p /etc/apt/sources.list.d
# echo "deb [trusted=yes] file://$DELIVERY_DIR /" >/etc/apt/sources.list.d/6wind.
↪list
# mkdir -p /etc/apt/preferences.d
# echo "Package: *"           >/etc/apt/preferences.d/6WIND
# echo "Pin: release o=6wind" >>/etc/apt/preferences.d/6WIND
# echo "Pin-Priority: 900"    >>/etc/apt/preferences.d/6WIND
# apt-get update
```

Package installation/update

1. Install or update the Virtual Accelerator package:

```
# apt-get install -y virtual-accelerator
```

2. Install or update the KPIs add-on (require python 3):

```
# apt-get install -y virtual-accelerator-addon-kpis
```

3. Install or update the QuickAssist add-on[1]:

[1] requires a Virtual Accelerator IPsec Application License

```
# apt-get install -y virtual-accelerator-addon-qat
```

4. Remove unneeded leftover dependencies:

```
# apt-get autoremove
```

1.2.4 Packages update

Note:

- Updating from a major release[1] to the next one is not supported. A new major release must be installed from scratch after deinstalling the previous one.
- Updating is only supported between successive minor releases[2], e.g. VA 1.7 -> 1.8 -> 1.9.

Since Virtual Accelerator version 1.9 it is possible to update Virtual Accelerator without deleting the VMs (Virtual Machines) with the following sequence:

1. Prepare update of the Virtual Accelerator packages:
   a. Save information about existing VMs:

```
# virtual-accelerator.sh save
```

   Note: Since Virtual Accelerator version 1.9.3 this step is not necessary, it is automatically done when the Virtual Accelerator service is stopped.

   b. Stop the Virtual Accelerator service:

```
# systemctl stop virtual-accelerator
```

2. Install Virtual Accelerator packages:

   Follow the procedure described in *Packages installation*

3. Reestablish connection with existing VMs:
   a. Restart the Virtual Accelerator and Open vSwitch services:

```
# systemctl start virtual-accelerator
# systemctl restart openvswitch
```

   b. Restore information about existing VMs:

[1] first number in the version field, i.e. `X` in `X.Y.Z`
[2] second number in the version field, i.e. `Y` in `X.Y.Z`

```
# virtual-accelerator.sh restore
```

Note: Since Virtual Accelerator version 1.9.3 this step is not necessary, it is automatically done when the Virtual Accelerator service is started.

Note: No change must be done on the VMs (creation, suspend ...) during the Virtual Accelerator update to allow a reconnection of these VMs after the update.

Warning: Minimal version of QEMU to update Virtual Accelerator without deleting the VMs is 2.12.

1.2.5 OpenStack support

One of the services provided by OpenStack Nova is to create the virtual ports of the VMs so OpenStack Neutron can build the proper networking services toward these VMs. Virtual Accelerator's virtual port support relies on the vhost-user feature, which is not natively supported by OpenStack yet. Therefore, 6WIND has built OpenStack plugins that enable the use of QEMU's and libvirt's vhost-user capabilities. They are provided separately.

6WIND OpenStack plugins should be installed at the last step of your deployment, when nova-compute, neutron-server and virtual-accelerator services are up and running:

Openstack Rocky and later releases:

```
root@compute:~# yum install -y fp-vdev-remote-stein-last.rpm
root@compute:~# yum install -y networking-6wind-stein-last.rpm
root@compute:~# yum install -y os-vif-6wind-plugin-stein-last.rpm
```

Openstack Queens and earlier releases:

```
# yum install -y 6wind-openstack-extensions
```

Note: OpenStack has to be installed before installing Virtual Accelerator and plugins.

Note: After OpenStack Queens, 6WIND neutron plugins have replaced this 6wind-openstack-extensions package.

1.2.6 Installing your license file

A license key is required to unlock Virtual Accelerator features and capacities. Refer to the *License management* section for detailed information about the Virtual Accelerator licensing mechanisms.

Note: The Virtual Accelerator service must be restarted to take the new license configuration in account.

Offline mode

In this mode, a license code and a license file are provided.

Download and install the license file in the `/etc` directory. You can assign this file any name, for example `va.lic`:

```
# wget -O /etc/va.lic <url-to-license-file>
```

Note: Any file transfer protocol (FTP, SCP, ...) can be used to transfer the license file. Copying and pasting the contents of the file is also possible.

Create a configuration file for this license:

```
# vrl-conf --product=virtual-accelerator --code=<license-code> --license-file=/etc/va.
↪lic
```

Online mode

In this mode, only a license code is provided. An Internet access is required to validate the license online.

Create a configuration file for this license:

```
# vrl-conf --product=virtual-accelerator --code=<license-code>
```

You may add additional connection parameters like `--proxy=[USER:PASSWORD@]HOST[:PORT]` if you wish to connect through a proxy, or `--netns=NETNS` to connect from a specific network namespace.

1.2.7 Uninstallation

Stop the Virtual Accelerator service

1. Make sure Virtual Accelerator is no longer running, following *the documentation*:

```
# systemctl stop virtual-accelerator
```

2. Disable Virtual Accelerator from the list of services:

```
# systemctl disable virtual-accelerator
```

Uninstall add-ons

1. Remove the QuickAssist add-on[1]:

```
# apt-get remove -y virtual-accelerator-addon-qat
```

2. Remove the KPIs add-on:

```
# apt-get  remove -y virtual-accelerator-addon-kpis
```

Uninstall Virtual Accelerator

1. If you enabled OpenStack *support*, remove it now:

```
# apt-get remove -y 6wind-openstack-extensions
```

2. Uninstall the Virtual Accelerator repository:

```
# PRODUCT=virtual-accelerator
# DISTRIB=ubuntu-18.04
# PKG=6wind-${PRODUCT}-${DISTRIB}-repository
# apt-get purge -y $PKG
# apt-get update
```

Note: Change the `PRODUCT` to `virtual-accelerator-ipsec` for Virtual Accelerator **IPsec**.

---

3. Uninstall the meta-package (which will remove most dependencies):

```
# apt-get autoremove -y $PRODUCT
```

[1] requires a Virtual Accelerator IPsec Application License

4. If you enabled OpenStack, reinstall the latest openvswitch package (previously overridden by Virtual Accelerator):

```
# apt-get upgrade -y openvswitch-switch
```

> **Warning:** Some configuration files are conflicting between 6WIND version and the official one. Use files from the package maintainer.

5. If you enabled OpenStack, remove 6WIND packages that depended on openvswitch and were thus not removed when cleaning the Virtual Accelerator meta-package:

```
# apt-get autoremove -y
```

Cleanup

1. Restart the service:

```
# systemctl restart libvirt-bin
```

1.3 Management

This section details the general Virtual Accelerator management concepts.

1.3.1 License

Overview

License key

A license key is required to unlock Virtual Accelerator features and capacities, according to the purchased licenses.

License Type	License Name	Capacity
Network	Virtual Accelerator	Throughput: 100Mbps to 100 Gbps
Application	Virtual Accelerator IPsec	Number of tunnels: 10 to 2K

A single license key is used to enable a set of features and capacities. You should have received a license file and license code at time of software delivery. If needed, you can request them again by filing a ticket on the 6WIND Customer Zone.

Activation and health check

The license key must first be configured and activated on Virtual Accelerator. This is done by installing the provided license file and creating a license configuration file using the provided license code.

See also:

The *getting started guide* for more information.

Note: In some cases, online licenses may be provided, which rely solely on a license key and an connection to the license key server to activate and validate the key.

Maintenance end date

The license key has a maintenance end date associated, which corresponds to the end of the maintenance period after the first delivery of Virtual Accelerator. The maintenance end date is updated when the maintenance agreement is renewed. The license key is not valid for releases of Virtual Accelerator released after the maintenance end date.

Showing license information

Use the `vrl-status` command to display the license status.

```
# vrl-status
Active perpetual license for virtual-accelerator
Serial number is xxxxxxxxxxxxxxxxxxxxxx
License was activated offline
Support is valid until 2022-01-01 00:00:00 (standard mode)
Max throughput 100.0G (moving average 0.0G)
IPsec activated for 100000 tunnels (currently used 0)
#
```

You will want to look for the following output to confirm that your license key is active and valid:

- `Active perpetual license for Virtual Accelerator`
- `License was activated offline`

In case of doubt, jump to the *next section* to learn how to check the license service logs.

For automation, the state of the license can also be retrieved in json format using `vrl-status -j`.

```
# vrl-status -j
{
  "status": 1,
  "status_text": "Product Authorized",
```

(continues on next page)

(continued from previous page)

```
  "short_license_type": 1,
  "short_license_type_text": "perpetual",
  "online": false,
  "computer_id": "",
  "serial": "xxxxxxxxxxxxxxxxxxxxxx",
  "connected": false,
  "last_healthcheck_date": 0,
  "lease_end_date": 0,
  "current_activations": 0,
  "allowed_activations": 0,
  "license_type": 6001,
  "license_type_text": "Perpetual, local",
  "activation_type": 6001,
  "activation_type_text": "License was activated offline",
  "support_type": "standard",
  "support_end_date": 1640991600,
  "throughput_max": 100.0,
  "ipsec_tunnels_max": 100000,
  "cgnat_conntracks_max": 0,
  "throughput_avg": 0.0,
  "ipsec_tunnels": 0,
  "ipsec_tunnels_avg": 0.0,
  "cgnat_conntracks": 0,
  "cgnat_conntracks_avg": 0.0
}
#
```

Checking license information

`journalctl -a -u vrld` provides the licensing logs.

```
-- Logs begin at Sun 2020-04-26 12:25:41 CEST, end at Wed 2020-05-06 08:28:16 CEST. --
May 06 08:28:15 localhost systemd[1]: Starting vRouter License Daemon...
May 06 08:28:15 localhost vrld[9254]: license_code: xxxxxxxxxxxxxxxxxxxxxx
May 06 08:28:15 localhost vrld[9254]: product: virtual-accelerator
May 06 08:28:15 localhost vrld[9254]: workdir: /run/license
May 06 08:28:15 localhost vrld[9254]: passive_file: /etc/va.lic
May 06 08:28:15 localhost vrld[9254]: Using passive license file
May 06 08:28:15 localhost vrld[9254]: Dumping state to /run/license/state.json
May 06 08:28:15 localhost vrld[9254]: status: Product Authorized (1)
May 06 08:28:15 localhost vrld[9254]: started
May 06 08:28:15 localhost systemd[1]: Started vRouter License Daemon.
```

In the nominal case, license key activation success is indicated by the following message:

```
May 06 08:28:15 localhost vrld[9254]: status: Product Authorized (1)
```

1.3.2 Zero configuration

The Virtual Accelerator can be started without any configuration. It is called the *zero configuration* mode. In this case, the fast path uses one physical core per socket, and takes all supported physical network ports. Virtual network ports should be created on demand before starting a VM (Virtual Machine) (see the *Hotplug a virtual port* section).

In most cases, zero configuration is enough and you'll want to jump to the *Optimizing performance* section.

Expert users may want to fine tune the configuration in order to assign more cores to the fast path or restrict the NICs polled by the fast path. In this case, move on to the *Configuration tuning* section.

Note: The fast path is the Virtual Accelerator component in charge of packet processing. More details were provided in *Data Plane* section.

1.3.3 Configuration tuning

Configuration wizard

A configuration wizard is provided to customize the fast path configuration. To launch the wizard, do:

```
# fast-path.sh config -i

Fast path configuration
=======================

1 - Select fast path ports and polling cores
2 - Select a hardware crypto accelerator
3 - Advanced configuration
4 - Advanced plugin configuration
5 - Display configuration

S - Save configuration and exit
Q - Quit

Enter selection [S]:
```

The `1 - Select fast path ports and polling cores` option takes care of the mandatory fast path configuration, which comprises:

Core allocation The fast path needs dedicated cores that are isolated from other Linux tasks.

Physical port assignation The fast path must have full control over a network port to provide acceleration on this port. At fast path start, a DPVI (Data Plane Virtual Interface) will replace each Linux interface associated to a fast path port. The new interface has the same name as the old interface. The configuration that was done on the old interface is lost (IP addresses, MTU (Maximum Transmission Unit), routes, etc).

Virtual port creation To handle traffic to and from virtual machines, the fast path must create virtual ports. Like for physical interfaces, a DPVI will be created for each virtual port. The virtual ports can be created through the wizard. However, it is usually better to allocate the ports at VM start.

See also:

the *Hotplug a virtual port* section for more information.

To make the guest network performance scale with the number of cores, you must use the multiqueue feature. To exchange packets between the host and its guests, virtual rings are used. Each virtual ring will be polled by one fast path logical core located on the same socket as the VM. If there are less fast path logical cores than virtual rings, some fast path logical cores will poll several virtual rings. The number of virtual rings is configured in the VM using `ethtool -L`.

See also:

- the *Guest multiqueue configuration* section below for more information about configuring a guest with several CPUs (Central Processing Units),
- the *Virtio Host PMD* documentation for detailed information about multiqueue configuration.

In OpenStack deployments, this step is not needed. Nova will automatically create the needed ports on demand, using the fp-vdev command itself.

Core to port mapping The fast path cores' main task is to check if packets are available on a port, and process these packets. In most use cases, good performance is obtained with the default configuration: all cores poll all ports of the same socket.

The `2 - Select a hardware crypto accelerator` option allows to select the crypto acceleration type[1].

Crypto acceleration selection In order to benefit from crypto acceleration, an acceleration engine must be selected. The supported engines are:

- Intel Multi-Buffer for software crypto acceleration using AES-NI
- Intel Coleto Creek for hardware crypto acceleration using Intel Communications Chipset 895x Series, 8925 or 8926 (Coleto Creek)

If no acceleration engine is selected, software crypto takes place.

In `3 - Advanced configuration`, the following parameters can be customized:

fast path memory allocation (FP_MEMORY) The fast path needs dedicated memory. The fast path dedicated memory is allocated in hugepages.

[1] requires a Virtual Accelerator IPsec Application License

Note: A hugepage is a page that addresses more memory than the usual 4KB. Accessing a hugepage is more efficient than accessing a regular memory page. Its default size is 2MB.

VM memory allocation (VM_MEMORY) For performance reasons, the memory used by the VMs is reserved in hugepages. By default, Virtual Accelerator allocates 4GB per socket.

Mbuf pool preallocation (NB_MBUF) The network packets manipulated by the fast path are stored in buffers named mbufs. A mbuf pool is allocated at fast path start.

Offloads configuration (FP_OFFLOAD) TCP/UDP offloads are used to maximize performance. They must be:

- enabled if the traffic is mostly composed of TCP/UDP sessions terminated on the guest (default value on a Virtual Accelerator)
- disabled if the guests are mostly forwarding traffic

By default, offloads are enabled.

The `S - Save configuration and exit` option writes the configuration file in `/etc/fast-path.env`.

See also:

The 6WINDGate *Fast Path Baseline* documentation for more information about `fast-path.env`.

After running the wizard, if you don't need additional fine tuning of the fast path configuration, you'll want to jump to the *Optimizing performance* section.

Configuration files

Experts users can fine tune the fast path configuration even further by editing its configuration files.

The table below shows which configuration file and variable are relevant for each configuration step.

Step	Variable	Configuration File
core allocation	`FP_MASK`	`/etc/fast-path.env`
memory allocation	`FP_MEMORY` and `VM_MEMORY`	`/etc/fast-path.env`
physical ports assignation	`FP_PORTS`	`/etc/fast-path.env`
core / port mapping	`CORE_PORT_MAPPING`	`/etc/fast-path.env`
mbuf preallocation	`NB_MBUF`	`/etc/fast-path.env`
offloads activation	`FP_OFFLOAD`	`/etc/fast-path.env`
cpuset activation	`CPUSET_ENABLE`	`/etc/cpuset.env`

See also:

- The 6WINDGate *Fast Path Baseline* documentation for more information about `fast-path.env` and about CPU isolation using cpuset
- The 6WINDGate FPN-SDK documentation for more information about offloads

1.3.4 Optimizing performance

To get the best performance, we need to isolate the CPUs reserved to the fast path from all other userland process and kernel threads. The most efficient way to achieve this, is to boot the hypervisor kernel with `isolcpus`, `nohz_full` and `rcu_nocbs` options.

These options must be added to the `GRUB_CMDLINE_LINUX` variable in `/etc/default/grub`. They should contain the same CPU list as in `FP_MASK` from `/etc/fast-path.env`.

```
# fp-conf-tool -D -F | grep FP_MASK
FP_MASK=4-7,12-15
# vi /etc/default/grub
...
GRUB_CMDLINE_LINUX="crashkernel=auto console=ttyS0,115200n8 \
  isolcpus=4-7,12-15 nohz_full=4-7,12-15 rcu_nocbs=4-7,12-15"
...
```

Then issue the following command as root to update the `grub` install:

```
# update-grub
```

1.3.5 Start and stop

Once the fast path configuration is complete, we can start Virtual Accelerator.

First, make sure that the Network Manager service is stopped and disabled:

```
# systemctl stop NetworkManager.service
# systemctl disable NetworkManager.service
```

Then, start Virtual Accelerator as follows:

```
# service virtual-accelerator start
```

Note: Please remember that the fast path takes control of all supported physical network ports by default. You might want to prevent the fast path from taking your management interface, using the configuration wizard.

By default, since Virtual Accelerator version 1.9.3, re-connection with existing VMs, managed by the previous Virtual Accelerator instance, is automatic. If this behavior is not expected, the previous Virtual Accelerator state must be deleted before the start operation.

```
# virtual-accelerator.sh clean
# service virtual-accelerator start
```

To stop it:

```
# service virtual-accelerator stop
```

Note: If stop operation is done to update some fast path parameters, existing VMs can be kept. For that VMs information are automatically saved before the stop and restored after the start.

```
# service virtual-accelerator stop
# # update system configuration
# service virtual-accelerator start
```

No change can be done on VMs between the save and restore operations.

Note: When the Virtual Accelerator service is started or stopped, it will attempt to restart all interfaces configured in the system by calling `ifdown -a` followed by `ifup -a`.

Note: libvirt must be restarted after the first Virtual Accelerator start to actually see the hugepages allocated in `VM_MEMORY`.

Note: If your setup comprises Open vSwitch bridges, the Open vSwitch service must be restarted each time Virtual Accelerator is restarted.

1.3.6 Networking configuration

Virtual Accelerator transparently configures the fast path when Linux is configured, using the *Linux - Fast Path Synchronization* module. The standard Linux tools can be used (`iproute2`, `iptables`, `brctl`, `ovs-ofctl`, `ovs-vsctl`, etc.).

Static IPsec configuration relies on the standard Linux `ip xfrm` commands. IKE configuration is based on strongSwan (https://strongswan.org). IPsec or IKE requires a Virtual Accelerator IPsec Application License.

Offloads: TSO and GRO increase throughput of end-to-end communications by taking care of packet segmentation/aggregation in Virtual Accelerator, instead of doing it in the VM. These mechanisms maximize throughput by minimizing the number of packets processed and the number of transactions on the virtual NIC.

TSO/GRO can dramatically improve performance in some cases, but it may also have the opposite effect. To determine whether to enable or disable TSO/GRO, take a look at how the VM processes traffic:

- If the VM terminates the traffic (HTTP servers, databases, etc.), **enable** TSO/GRO (default state).
- If the VM performs packet by packet processing (L2/L3 forwarding, IPsec processing, firewalling, etc.), **disable** TSO/GRO. As the VM simply makes a processing decision on each packet, TSO/GRO adds unnecessary processing cycles and increases latency.

See also:

- the Linux documentation and man pages for iproute2, iptables, brctl, etc.
- the *fp-shmem-ports* section below
- the 6WINDGate FPN-SDK documentation for more information about offloads
- the *Virtio Host PMD* documentation for detailed information about the NFV profile
- the 6WINDGate *Control Plane Security - IKEv1 and IKEv2* documentation for more information about IKE

1.3.7 Starting a VM

This section describes how to run and manage VMs.

Hotplug a virtual port

VMs will need virtual ports to communicate with the outside world. To create such ports dynamically in the fast path, the `fp-vdev` command can be used. When the VM will start, each virtual ring will be polled by one fast path logical core located on the same socket as the VM. If there are less fast path logical cores than virtual rings, some fast path logical cores will poll several virtual rings. The number of virtual rings is configured in the VM using `ethtool -L`.

A hotplug port is removed when the fast path is restarted. In that case, to restore it, it has to be recreated, and any configuration made on the associated FPVI (Fast Path Virtual Interface) interface has to be remade. VMs using this interface must be restarted as well.

Note: This is the preferred alternative, but it is possible to create virtual ports at fast path start using the configuration wizard.

For instance, the command below creates a virtual port named `tap0`, that will connect to the `/tmp/pmd-vhost0` vhost-user socket.

```
# fp-vdev add tap0 --sockpath /tmp/pmd-vhost0
devargs:
  sockmode: client
  sockname: /tmp/pmd-vhost0
driver: pmd-vhost
ifname: tap0
rx_cores: all
```

Note: OpenStack Nova calls `fp-vdev` to create the needed ports itself, so it should not be done manually.

Note: Make sure that the fast path has been started before you create hotplug ports with `fp-vdev` commands

See also:

The 6WINDGate *Fast Path Managing virtual devices* documentation for more information about `fp-vdev` command.

Libvirt configuration

It is recommended to use libvirt (https://libvirt.org/) to start VMs with Virtual Accelerator.

Note: Please make sure that the fast path CPU isolation feature is disabled (see *Configuration files*).

XML (eXtended Markup Language) domain file

Interfaces

The fast path virtual ports are compatible with the vhost-user virtio backend of QEMU.

Example of an interface compatible with a fast path virtual port:

```
<interface type='vhostuser'>
  <source type='unix' path='/tmp/pmd-vhost0' mode='server'/>
  <model type='virtio'/>
</interface>
```

Example of a 4 queues interface compatible with a fast path virtual port:

```
<interface type='vhostuser'>
  <source type='unix' path='/tmp/pmd-vhost0' mode='server'/>
  <model type='virtio'/>
  <driver queues='4'/>
</interface>
```


Hugepages

The VM should be started with memory allocated from hugepages, in shared mode.

Example of a VM with 1GB of memory using 2MB hugepages taken from NUMA node `0` of the host:

```
<cpu>
  <numa>
    <cell id="0" cpus="0" memory="1048576" memAccess="shared"/>
  </numa>
</cpu>
<numatune>
  <memory mode='strict' nodeset='0'/>
</numatune>
<memoryBacking>
  <hugepages>
    <page size="2048" unit="KiB"/>
  </hugepages>
</memoryBacking>
```

See also:

The libvirt XML documentation: https://libvirt.org/formatdomain.html

Guest multiqueue configuration

The multiqueue feature allows the guest network performance to scale with the number of CPUs. To enable multiqueue in guest configuration, a script to automatically configure the right number of queues at interface creation must be added.

For Centos-like distributions

Copy the file below to `/sbin/ifup-pre-local`:

```
#!/bin/bash

. /etc/init.d/functions

cd /etc/sysconfig/network-scripts
. ./network-functions

[ -f ../network ] && . ../network

CONFIG=${1}
```

(continues on next page)

(continued from previous page)

```
need_config "${CONFIG}"

source_config

if [ -n "$DEVICE" ]; then
    nb_queues=`ethtool -l $DEVICE | grep Combined: | awk '{print $2}' | head -n1`
    ethtool -L $DEVICE combined $nb_queues

    # configure tx queues
    nb_processor=`cat /proc/cpuinfo | grep processor | wc -l`
    nb_xps=$nb_processor
    if [ "$nb_queues" -lt $nb_xps ]; then
        nb_xps=$nb_queues
    fi

    last_xps=$(($nb_xps-1))
    for i in `seq 0 $last_xps`;
    do
        let "mask_cpus=1 << $i"
        echo $mask_cpus >  /sys/class/net/$DEVICE/queues/tx-$i/xps_cpus
    done
fi
```

For Ubuntu-like distributions

Copy the file below to `/etc/network/if-pre-up.d/enable_multiqueue`:

```
#!/bin/sh

ETHTOOL=/sbin/ethtool

test -x $ETHTOOL || exit 0

[ "$IFACE" != "lo" ] || exit 0

nb_queues=`ethtool -l $IFACE | grep Combined: | awk '{print $2}' | head -n1`
ethtool -L $IFACE combined $nb_queues

# configure tx queues
nb_processor=`cat /proc/cpuinfo | grep processor | wc -l`
nb_xps=$nb_processor
if [ "$nb_queues" -lt $nb_xps ]; then
    nb_xps=$nb_queues
```

(continues on next page)

(continued from previous page)

```
fi

last_xps=$(($nb_xps-1))
for i in `seq 0 $last_xps`;
do
    let "mask_cpus=1 << $i"
    echo $mask_cpus >  /sys/class/net/$IFACE/queues/tx-$i/xps_cpus
done
```

Note: This script will be called for interfaces configured in the `/etc/network/interfaces` file. It will be called when the networking service is started, or when `ifup <ifname>` is called. Please check `man interfaces` for more informations.

OpenStack

Hugepages

1. Enable hugepages in the flavor of the VM:

```
# openstack flavor set --property hw:mem_page_size=large myflavor
```

Note: If a VM is spawned without hugepages, its virtual ports will be created, but will not be functional.

Note: It is required to have hugepages activated on Compute node.

Multiqueue

This section explains how to spawn VMs with multiple queues. The multiqueue feature allows the guest network performance to scale with the number of CPUs.

1. Enable multiqueue in the VM's image template:

```
# openstack image set --property hw_vif_multiqueue_enabled=true myimage
```

Note: The image template must have been prepared according to *Guest multiqueue configuration*.

> **Warning:** `hw_vif_multiqueue_enabled` must be set to false if monoqueue VM is needed.

The VMs will be booted with a number of queues equal to the number of vCPUs.

CPU isolation

1. To make sure that Nova does not spawn VMs on the CPUs dedicated to the fast path, set the `vcpu_pin_set` attribute in the DEFAULT section of `/etc/nova/nova.conf`.

```
# crudini --set /etc/nova/nova.conf DEFAULT vcpu_pin_set 0-1
```

1.4 Useful tools

This section gives an overview of the scripts, tools and commands that are used to manage Virtual Accelerator, and where to find their documentation.

1.4.1 `fast-path.sh`

The `fast-path.sh` script manages the fast path module. Several commands are available.

- config to help filling the fast path configuration file
- start, stop, restart
- status to check on the different components of the fast path module
- dump the current configuration and the effective value of automatic parameters

Example: start the fast path configuration wizard

```
# fast-path.sh config -i
```

Example: dump useful informations about the machine

```
# fast-path.sh config -m
Configuring Fast Path...

General information
-------------------

```

(continues on next page)

(continued from previous page)

```
  Processor name: Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz
  Sockets 2, Cores 20, Hyperthreads 40
  Multi channel memory architecture: 4 channels

socket 0
--------

Cores list:
  Core   0 = [  0- 20]  Core   1 = [  1- 21]  Core   2 = [  2- 22]
  Core   3 = [  3- 23]  Core   4 = [ 24-  4]  Core   8 = [ 25-  5]
  Core   9 = [ 26-  6]  Core  10 = [ 27-  7]  Core  11 = [  8- 28]
  Core  12 = [  9- 29]

Ethernet PCI information:
  Bus id       Interface  NIC full name
  ============ ========== ====================================================
  0000:05:00.0 mgmt0      Intel Corporation I350 Gigabit Network Connection (rev
  0000:05:00.1 eth2       Intel Corporation I350 Gigabit Network Connection (rev
  0000:05:00.2 eth4       Intel Corporation I350 Gigabit Network Connection (rev
  0000:05:00.3 eth5       Intel Corporation I350 Gigabit Network Connection (rev

Available crypto:
  Intel Multibuffer

socket 1
--------

Cores list:
  Core   0 = [ 10- 30]  Core   1 = [ 11- 31]  Core   2 = [ 32- 12]
  Core   3 = [ 33- 13]  Core   4 = [ 34- 14]  Core   8 = [ 35- 15]
  Core   9 = [ 16- 36]  Core  10 = [ 17- 37]  Core  11 = [ 18- 38]
  Core  12 = [ 19- 39]

Ethernet PCI information:
  Bus id       Interface  NIC full name
  ============ ========== ====================================================
  0000:83:00.0 eth1       Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network
  0000:83:00.1 eth3       Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network

Available crypto:
  Intel Multibuffer
```

1.4.2 linux-fp-sync.sh

The `linux-fp-sync.sh` script manages the *Linux - Fast Path Synchronization* module. Several commands are available:

- start, stop, restart
- status to check the status of the different components of the *Linux - Fast Path Synchronization* module

`linux-fp-sync.sh` should be started after `fast-path.sh`, to start the applications that will synchronize the fast path with Linux. It should be stopped before `fast-path.sh`.

Example: start the Linux - Fast Path Synchronization module

```
# linux-fp-sync.sh start
```

1.4.3 fp-cli

The `fp-cli` tool interacts with the fast path. It is used to:

- display statistics
- display debugging information

Example: dump the interfaces synchronized from Linux in the fast path:

```
# fp-cli iface
1:eth1 [VR-0] ifuid=0x15867c2 (virtual) <FWD4|FWD6> (0x60)
        type=ether mac=90:e2:ba:6d:c5:30 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
        if_ops: rx_dev=none tx_dev=none ip_output=none
250:eth4 [VR-0] ifuid=0xfa4c59d2 (port 1) <FWD4|FWD6> (0x60)
        type=ether mac=00:1e:67:68:0b:1f mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
        if_ops: rx_dev=none tx_dev=none ip_output=none
418:eth5 [VR-0] ifuid=0xa29daa82 (virtual) <FWD4|FWD6> (0x60)
        type=ether mac=00:1e:67:68:0b:20 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
        if_ops: rx_dev=none tx_dev=none ip_output=none
425:eth2 [VR-0] ifuid=0xa9a9b872 (port 0) <FWD4|FWD6> (0x60)
        type=ether mac=00:1e:67:68:0b:1e mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
        if_ops: rx_dev=none tx_dev=none ip_output=none
650:lo [VR-0] ifuid=0x8a0a5828 (virtual) <UP|RUNNING|FWD4|FWD6> (0x63)
```

(continues on next page)

(continued from previous page)

```
        type=loop mac=00:00:00:00:00:00 mtu=0 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
        if_ops: rx_dev=none tx_dev=none ip_output=none
849:eth3 [VR-0] ifuid=0x51fb0922 (virtual) <FWD4|FWD6> (0x60)
        type=ether mac=90:e2:ba:6d:c5:31 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
        if_ops: rx_dev=none tx_dev=none ip_output=none
888:mgmt0 [VR-0] ifuid=0x78fb6b7a (virtual) <UP|RUNNING|FWD4|FWD6> (0x63)
        type=ether mac=00:1e:67:68:0b:1d mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=4  IPv6 routes=0
        if_ops: rx_dev=none tx_dev=none ip_output=none
931:fpn0 [VR-0] ifuid=0xa307dac2 (virtual) <UP|RUNNING|FWD4|FWD6> (0x63)
        type=ether mac=00:00:46:50:4e:00 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
        if_ops: rx_dev=none tx_dev=none ip_output=none
```

Example: display the fast path ip stack statistics:

```
# fp-cli ip4-stats
  IpForwDatagrams:0
  IpInReceives:0
  IpInDelivers:0
  IpInHdrErrors:0
  IpInAddrErrors:0
  IpDroppedNoArp:0
  IpDroppedNoMemory:0
  IpDroppedForwarding:0
  IpDroppedIPsec:0
  IpDroppedBlackhole:0
  IpDroppedInvalidInterface:0
  IpDroppedNetfilter:0
  IpDroppedNoRouteLocal:0
  IpReasmTimeout:0
  IpReasmReqds:0
  IpReasmOKs:0
  IpReasmFails:0
  IpReasmExceptions:0
  IpFragOKs:0
  IpFragFails:0
  IpFragCreates:0
```


Example: dump the fast path license status (no license installed here):

```
# fp-cli license
VALID: License is valid.
- Product: on
- IPsec: on
```

The `fp-cli` help command gives a list of the available commands and their syntax.

Example: display ip4-stats syntax

```
# fp-cli help ip4-stats
ip4-stats                : Display IPv4 statistics
                ip4-stats [percore] [all]
```

1.4.4 fp-shmem-ports

The `fp-shmem-ports` tool helps to dump and configure the network ports managed by the fast path.

Example: dump the network ports detected by the fast path:

```
# fp-shmem-ports -d
core freq : 2133430791
offload : enabled
vxlan ports :
port 4789 (set by user)
port 8472 (set by user)
port 0: ntfp1 mac de:ed:01:bb:42:77 driver rte_em_pmd GRO timeout 10us
RX queues: 1 (max: 1)
TX queues: 1 (max: 1)
GRO on
port 1: ntfp2 mac de:ed:02:e4:cd:e2 driver rte_em_pmd GRO timeout 10us
RX queues: 1 (max: 1)
TX queues: 1 (max: 1)
GRO on
```


Example: disable GRO on all fast path ports:

```
# fp-shmem-ports -e all -K gro off
```

1.4.5 fp-cpu-usage

From Linux point of view, the fast path process always appears to consume 100% of the CPUs that are allocated to it. This is because even if the fast path does not receive packets, it continuously polls the NICs for new packets. Therefore, a tool such as `top` always shows 100% CPU usage.

The `fp-cpu-usage` tool allows to see the actual load of the fast path CPUs.

Example: an idle fast path using 2 cores:

```
# fp-cpu-usage
Fast path CPU usage:
cpu: %busy     cycles
 20:   <1%    2043136
 21:   <1%    2034404
average cycles/packets received from NIC: --- (4077540/0)
# top -n 1
(...)
   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 20490 root      20   0  902232 314380 295140 R 207.5  1.0   8:58.49 fp-rte:20
(...)
```

The following table shows where more complete documentation can be found for every tool described in this section.

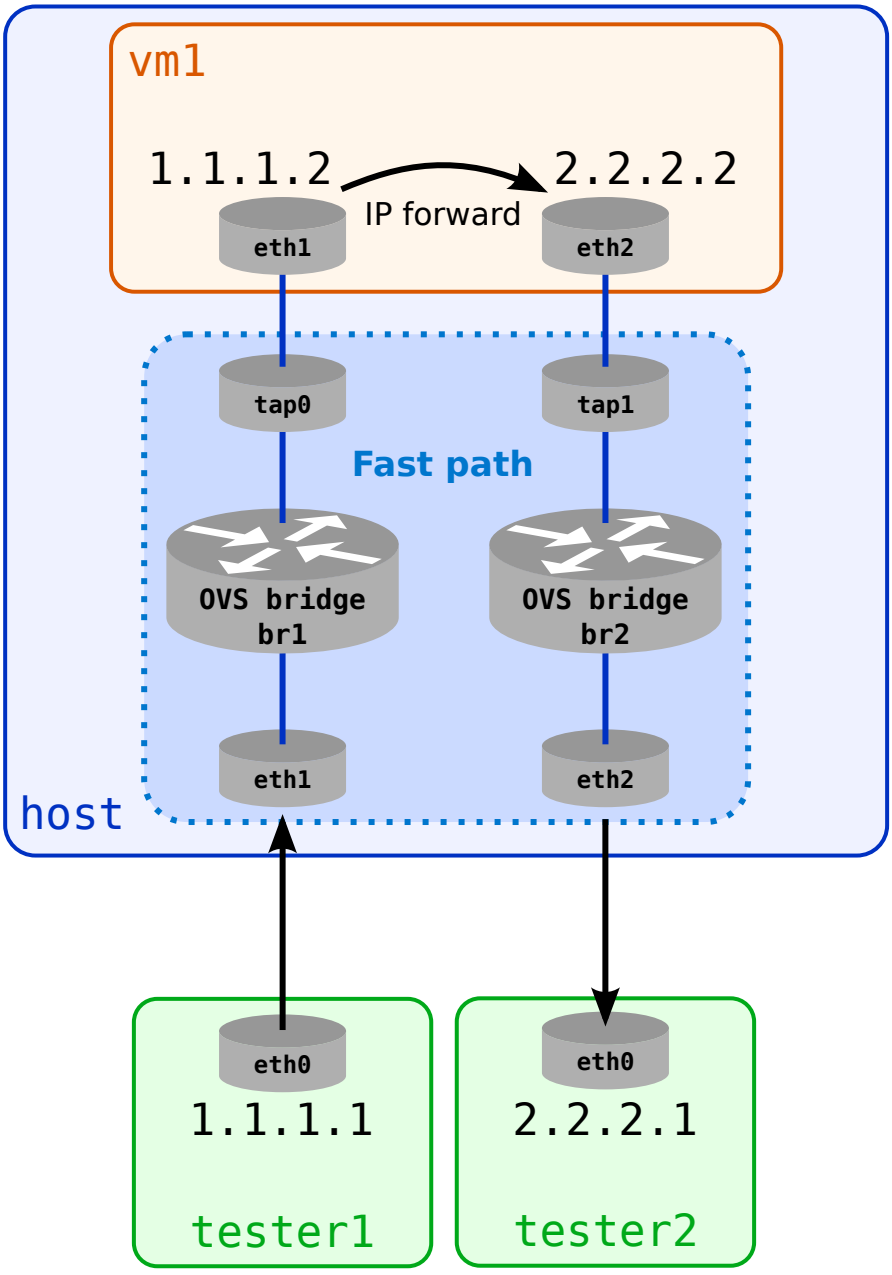
Tool	Related 6WINDGate module documentation
`fast-path.sh`	Fast Path Baseline
`linux-fp-sync.sh`	Linux - Fast Path Synchronization
`fp-cli`	Fast Path Baseline
`fp-shmem-ports`	FPN-SDK Baseline and FPN-SDK Baseline for DPDK
`fp-cpu-usage`	FPN-SDK Baseline

1.5 Use cases

This section details the configuration of Virtual Accelerator in the following use cases:

1.5.1 One VM, one core, Virtio monoqueue NICs, forwarding traffic, with Open vSwitch

A single VM forwards traffic from `tester1` to `tester2`. The hypervisor uses Open vSwitch bridges to connect its physical interfaces and the VM virtual interfaces. The VM runs on one core, and two Virtio interfaces using one virtual ring each. Virtual Accelerator runs on two cores (cores 1 and 2).



Virtual Accelerator configuration

In this first use case, the default fast path configuration is used:

- one physical core per socket is dedicated to the fast path
- all supported physical ports are polled by the logical cores located on the same socket
- 4GB of hugepages memory is reserved on each socket for the VMs.

1. Show the default fast path configuration

```
root@host:~# fast-path.sh config
Info: no configuration file /etc/fast-path.env for fast-path.sh, using defaults
Configuring Fast Path...
Fast path configuration info
============================

  Selected ethernet card
  ----------------------

  Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection
     (rev 01) PCI card mounted on ens1f0 with cores 7,23
  Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection
     (rev 01) PCI card mounted on ens1f1 with cores 7,23
  Intel Corporation I350 Gigabit Network Connection (rev 01) PCI card
     mounted on mgmt0 with cores 7,23
  Intel Corporation I350 Gigabit Network Connection (rev 01) PCI card
     mounted on enp6s0f1 with cores 7,23
  Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 01)
     PCI card mounted on ens7f0 with cores 15,31
  Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 01)
     PCI card mounted on ens7f1 with cores 15,31
  Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection
     (rev 01) PCI card mounted on ens5f0 with cores 15,31
  Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection
     (rev 01) PCI card mounted on ens5f1 with cores 15,31

   The logical cores 7 and 23 are located on the first socket, they poll
   the ports located on the first socket. The logical cores 15 and 31 are
   located on the second socket, they poll the ports located on the second
   socket.
```

2. If you are connected this machine using a network interface supported by the fast path (for instance using SSH), you should avoid the fast path to take control of this port and thus deleting its network configuration, which would break the connection.

```
root@host:~# ethtool -i mgmt0
[...]
bus-info: 0000:06:00.0
[...]
root@host:~# FP_PORTS="all -0000:06:00.0" fast-path.sh config --update
```

3. As we will not terminate traffic in the VMs, it is recommended to disable offloads to get better performance. See *Configuration files* for more information.

```
root@host:~# FP_OFFLOAD="off" fast-path.sh config --update
```

4. libvirt does not support the cpuset isolation feature; it has to be disabled in `/etc/cpuset.env`.

```
-#: ${CPUSET_ENABLE:=1}
+: ${CPUSET_ENABLE:=0}
```

5. Start Virtual Accelerator.

```
root@host:~# service virtual-accelerator start
```

6. Restart the Open vSwitch control plane.

```
root@host:~# service openvswitch-switch restart
```

7. The hugepages are allocated by Virtual Accelerator at startup and libvirt cannot detect them dynamically. libvirt must be restarted to take the hugepages into account.

```
root@host:~# service libvirt-bin restart
```

> **Warning:** If you restart Virtual Accelerator, you must restart openvswitch and libvirt (and its VMs) as well.

8. Create two virtio interfaces to communicate with the VM. The `sockpath` argument will be used in the libvirt XML file later.

   These new ports will be polled by fast path logical cores located on the same socket as the VM. The number of fast path logical cores polling a port depends on the number of virtual rings. Each virtual ring is polled by one fast path logical core. If there are less fast path logical cores than virtual rings, some fast path logical cores poll several virtual rings. The number of virtual rings is configured in the VM using `ethtool -L`.

```
root@host:~# fp-vdev add tap0 --sockpath=/tmp/pmd-vhost0 --profile=nfv
devargs:
  profile: nfv
  sockmode: client
  sockname: /tmp/pmd-vhost0
```

(continues on next page)

(continued from previous page)

```
  txhash: l3l4
  verbose: 0
driver: pmd-vhost
ifname: tap0
rx_cores: all
root@host:~# fp-vdev add tap1 --sockpath=/tmp/pmd-vhost1 --profile=nfv
devargs:
  profile: nfv
  sockmode: client
  sockname: /tmp/pmd-vhost1
  txhash: l3l4
  verbose: 0
driver: pmd-vhost
ifname: tap1
rx_cores: all
```

Note: Make sure that the fast path has been started before you create hotplug ports with `fp-vdev` commands

See also:

The 6WINDGate *Fast Path Managing virtual devices* documentation for more information about the `fp-vdev` command.

Linux configuration

VM Creation (if needed)

If you don't have a VM ready, you can use a cloud image. See *VM Creation* to create one VM with the following libvirt configuration sections:

- hostname vm1

```
<name>vm1</name>
```

- two vhost-user interfaces:

```
<interface type='vhostuser'>
   <source type='unix' path='/tmp/pmd-vhost0' mode='server'/>
   <model type='virtio'/>
</interface>
<interface type='vhostuser'>
   <source type='unix' path='/tmp/pmd-vhost1' mode='server'/>
```

(continues on next page)

(continued from previous page)

```
   <model type='virtio'/>
</interface>
```

- 1048576 bytes (1GB) of memory:

```
<memory>1048576</memory>
```

and

```
<numa>
  <cell id="0" cpus="0" memory="1048576" memAccess="shared"/>
</numa>
```

Configuration of the host and of VM1

Now that we have access to the VM, we can setup Linux with the configuration needed for forwarding.

1. Inside the VM, enable forwarding, set interfaces up and add addresses on both `1.1.1.0` and `2.2.2.0` subnets.

```
root@vm1:~# echo 1 > /proc/sys/net/ipv4/ip_forward
root@vm1:~# ip link set eth1 up
root@vm1:~# ip link set eth2 up
root@vm1:~# ip addr add 1.1.1.2/24 dev eth1
root@vm1:~# ip addr add 2.2.2.2/24 dev eth2
```

2. On the host, check the names of the fast path interfaces at the end of the port description in the `fp-shmem-ports -d` output.

```
root@host:~# fp-shmem-ports -d
core freq : 2693512037
offload : disabled
vxlan ports :
port 4789 (set by user)
port 8472 (set by user)
port 0: eth1 mac 00:1b:21:74:59:2c driver rte_ixgbe_pmd
RX queues: 2 (max: 128)
TX queues: 4 (max: 64)
RX vlan strip off
RX IPv4 checksum on
RX TCP checksum on
RX UDP checksum on
LRO off
port 1: eth2 mac 00:1b:21:74:59:2d driver rte_ixgbe_pmd
RX queues: 2 (max: 128)
```

(continues on next page)

(continued from previous page)

```
TX queues: 4 (max: 64)
RX vlan strip off
RX IPv4 checksum on
RX TCP checksum on
RX UDP checksum on
LRO off
(...)
port 7: tap0 mac 02:09:c0:8a:8a:94 driver pmd-vhost
   (args sockmode=client,sockname=/tmp/pmd-vhost0)
RX queues: 1 (max: 1)
TX queues: 4 (max: 64)
RX TCP checksum off
RX UDP checksum off
LRO off
port 8: tap1 mac 02:09:c0:f4:f5:fd driver pmd-vhost
   (args sockmode=client,sockname=/tmp/pmd-vhost1)
RX queues: 1 (max: 1)
TX queues: 4 (max: 64)
RX TCP checksum off
RX UDP checksum off
LRO off
```

3. On the host, set interfaces up.

```
root@host:~# ip link set eth1 up
root@host:~# ip link set eth2 up
root@host:~# ip link set tap0 up
root@host:~# ip link set tap1 up
```

4. On the host, create two OVS bridges, each containing a pair of physical/virtual interfaces.

```
root@host:~# ovs-vsctl add-br br1
root@host:~# ovs-vsctl add-br br2
root@host:~# ovs-vsctl add-port br1 tap0
root@host:~# ovs-vsctl add-port br1 eth1
root@host:~# ovs-vsctl add-port br2 tap1
root@host:~# ovs-vsctl add-port br2 eth2
```


Configuration of tester1 and tester2

1. On `tester1`, configure the `1.1.1.0` subnet and add a route to `2.2.2.0` subnet.

```
root@tester1:~# ip link set eth0 up
root@tester1:~# ip add add 1.1.1.1/24 dev eth0
root@tester1:~# ip route add 2.2.2.0/24 via 1.1.1.2
```

2. On `tester2`, configure the `2.2.2.0` subnet and add a route to `1.1.1.0`.

```
root@tester2:~# ip link set eth0 up
root@tester2:~# ip add add 2.2.2.1/24 dev eth0
root@tester2:~# ip route add 1.1.1.0/24 via 2.2.2.2
```

The Linux configuration is finished.

Testing

We can send traffic from `tester1` to `tester2` and check that the fast path switches it to and from the VM. First, let's do a ping to check the setup.

1. Reset the fast path statistics first.

```
root@host:~# fp-cli stats-reset
```

2. Ping the `tester2` address.

```
root@tester1:~# ping -c60 2.2.2.1
```

3. During traffic, you can check that the flows are available in the kernel.

```
root@host:~# ovs-dpctl dump-flows
recirc_id(0),in_port(6),eth(src=90:e2:ba:0e:4e:45,dst=52:54:00:fe:13:6f),
     eth_type(0x0800),ipv4(frag=no), packets:15, bytes:1470, used:0.000s,
     actions:5
recirc_id(0),in_port(5),eth(src=52:54:00:fe:13:6f,dst=90:e2:ba:0e:4e:45),
     eth_type(0x0800),ipv4(frag=no), packets:15, bytes:1470, used:0.000s,
     actions:6
recirc_id(0),in_port(4),eth(src=90:e2:ba:0e:4e:44,dst=52:54:00:11:91:7d),
     eth_type(0x0800),ipv4(frag=no), packets:15, bytes:1470, used:0.000s,
     actions:3
recirc_id(0),in_port(3),eth(src=52:54:00:11:91:7d,dst=90:e2:ba:0e:4e:44),
     eth_type(0x0800),ipv4(frag=no), packets:15, bytes:1470, used:0.000s,
     actions:4
```

4. The statistics are increased in the fast path, showing that the fast path processed the packets.


```
root@host:~# fp-cli fp-vswitch-stats
  flow_not_found:4
  output_ok:244
```

Note: The flow_not_found statistics is increased for the first packets of each flow, that are sent to Linux because they don't match any known flow in the fast path. Linux receives the packets and sends them to the ovs-vswitchd daemon (it is the standard Linux processing). The daemon creates a flow in the OVS kernel data plane. The flow is automatically synchronized to the fast path, and the next packets of the flow are processed by the fast path.

Now that we checked the setup, we can try `iperf`.

1. Reset the fast path statistics first.

```
root@host:~# fp-cli stats-reset
```

2. Install `iperf` on both servers and start the server on `tester2`.

```
root@tester1:~# apt-get install -y iperf
root@tester2:~# apt-get install -y iperf
root@tester2:~# iperf -s
```

3. Start `iperf` on `tester1`.

```
root@tester1:~# iperf -c 2.2.2.1
------------------------------------------------------------
Client connecting to 2.2.2.1, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  4] local 1.1.1.1 port 40338 connected with 2.2.2.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec  6.18 GBytes  5.31 Gbits/sec
```

4. During traffic, on the `host`, you can check the fast path usage.

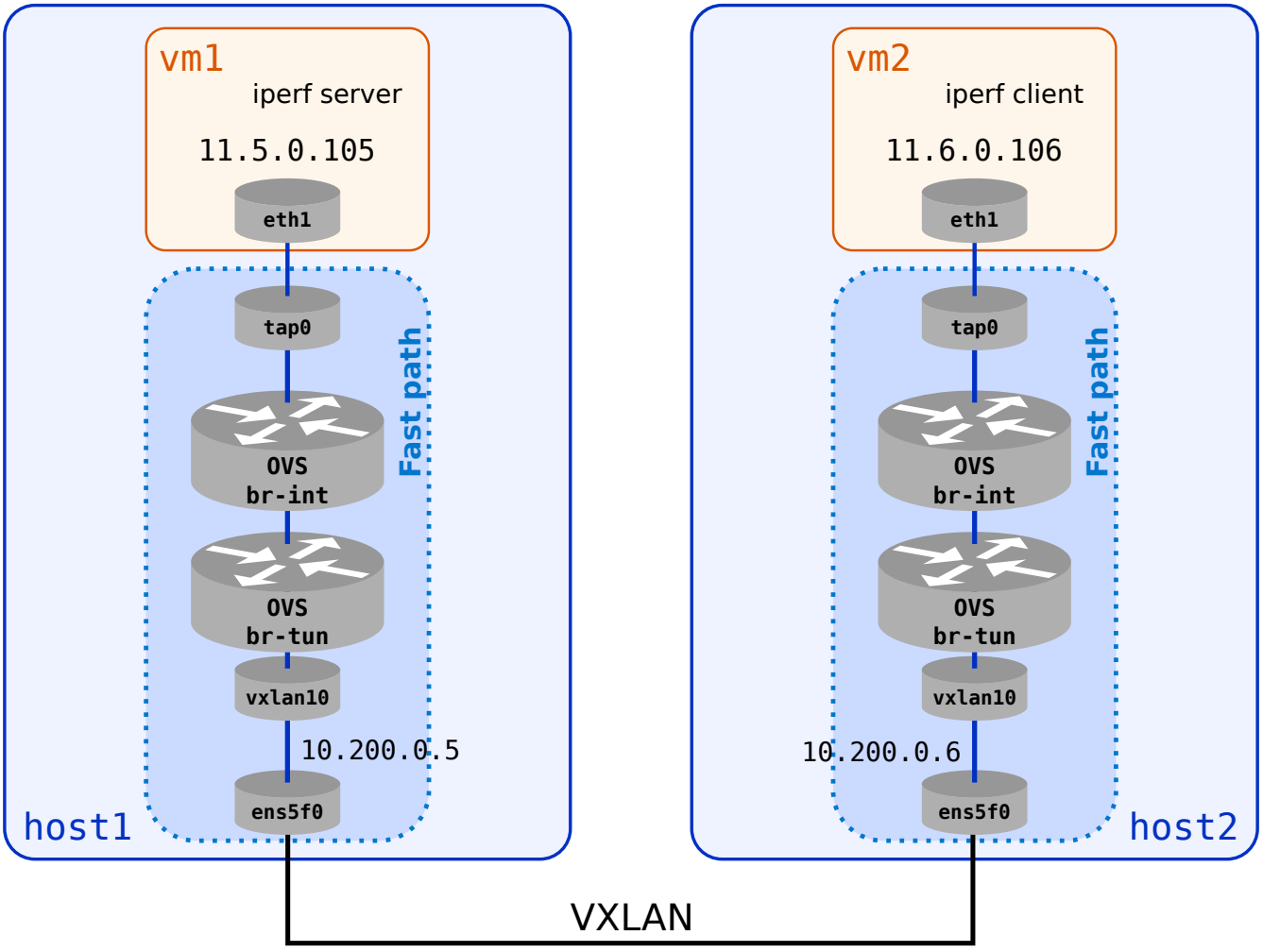
```
root@host:~# fp-cpu-usage
Fast path CPU usage:
cpu: %busy     cycles   cycles/packet
  7:    6%   34111856             5430
 15:   <1%    4730352               0
 23:   79%  429230848             2112
 31:   <1%    4616412               0
average cycles/packets received from NIC: 2256 (472689468/209463)
```

5. After traffic, you can check the fast path statistics.

```
root@host:~# fp-cli fp-vswitch-stats
  flow_not_found:8
  output_ok:19509389
```

1.5.2 Two VMs, one core, Virtio monoqueue NICs, VXLAN termination, offloading enabled, with Open vSwitch

Two VMs on different hypervisors exchange traffic on a private subnet. Hypervisors encapsulate traffic in a VXLAN tunnel. Each hypervisor uses Open vSwitch bridges to connect its physical interface and the VM virtual interface. Each VM runs on one core, and one Virtio interface using one virtual ring. Virtual Accelerator runs on two cores (cores 1 and 2).



Virtual Accelerator configuration

host1

In this second use case, the fast path configuration will be customized using the interactive wizard:

- Assign one physical core to the fast path
- Poll one physical port only
- Adapt the amount of memory reserved for the VM to our needs

1. Start the fast path configuration wizard tool

```
root@host1:~# fast-path.sh config -i
```

2. Select which physical ports the fast path should use.

   Enter into the first sub-menu:

```
Fast path configuration
=======================

1 - Select fast path ports and polling cores
2 - Select a hardware crypto accelerator
3 - Advanced configuration
4 - Advanced plugin configuration
5 - Display configuration

S - Save configuration and exit
Q - Quit

Enter selection [S]: 1
```

   Select `Manually configure FP_PORTS with e`. We see that `ens5f0` has the PCI bus identifier `0000:85:00.0`:

```
Fast path port selection
========================

Core/port mapping mode is auto

== ============ == ========== ============== ===================================
#  PCI or Name  Id Interface  Selected cores NIC full name
== ============ == ========== ============== ===================================
1  0000:03:00.0 0  ens1f0     auto: 7,23     Intel Corporation 82599ES 10-Gigabi
2  0000:03:00.1 0  ens1f1     auto: 7,23     Intel Corporation 82599ES 10-Gigabi
3  0000:06:00.0 0  mgmt0      auto: 7,23     Intel Corporation I350 Gigabit Netw
```

(continues on next page)

(continued from previous page)

```
4  0000:06:00.1 0  enp6s0f1   auto: 7,23     Intel Corporation I350 Gigabit Netw
5  0000:83:00.0 0  ens7f0     auto: 15,31    Intel Corporation Ethernet Controll
6  0000:83:00.1 0  ens7f1     auto: 15,31    Intel Corporation Ethernet Controll
7  0000:85:00.0 0  ens5f0     auto: 15,31    Intel Corporation 82599ES 10-Gigabi
8  0000:85:00.1 0  ens5f1     auto: 15,31    Intel Corporation 82599ES 10-Gigabi
== ============ == ========== ============== ===================================

A - Add a virtual device
C - Switch to manual core/port mapping mode
M - Set the fast path core mask. Current value is "auto" (7,15,23,31)
E - Manually configure FP_PORTS. Current value is "all"

B - Back

Enter selection [B]: e
```

Enable the `ens5f0` port using its PCI bus identifier:

```
Set the fast path ports
=======================

## FP_PORTS defines the list of ports enabled in the fast path.
##
## The value can be:
## - 'all' or commented out: all supported physical ports are enabled.
## - a space-separated list of keys, defining the list of ports. A key
##   adds one or several ports to the current list, or removes them if
##   it is prefixed by '-'. The valid keys are: 'all', a pci identifier,
##   a linux interface name.
##
##   Example: "" means no port
##   Example: "all -mgmt0" means all ports except the one associated to
##   the interface called mgmt0 in Linux.
##   Example: "0000:03:00.0 0000:03:00.1" means the ports whose pci bus
##   addresses match.
##
##   A PCI bus can be suffixed by driver-specific arguments. For instance:
##   "0000:03:00.0,role=right,verbose=1".
##
##   The list is evaluated from left to right, so that
##   "eth0 0000:03:00.0 -all" means no port are enabled.
##
##   Note: be careful when using Linux interface names in configuration,
##   as they are subject to changes, depending on system configuration.
```

(continues on next page)

(continued from previous page)

```
##
## This parameter is evaluated at fast path start and is converted into
## a whitelist or blacklist of PCI devices, that is passed to the fast
## path command line. Therefore, it is not possible to enable only some
## ports of a PCI device.
##
## In expert mode (EXPERT_FPCONF=on), this parameter is mandatory and
## must contain a list of PCI devices only.
##

Current value for FP_PORTS: all
Use 'none' to remove all ports
Enter new value [no change] > 0000:85:00.0
```

3. Our PCI device is located on the second socket. To achieve the best performance, it has to be polled by cores from the same socket. We choose to use the first physical core of the second socket (logical cores 10 and 30).

Note: Use `fast-path.sh config -m` to display the machine topology.

Select `m` to modify the fast path core mask:

```
Fast path port selection
========================

Core/port mapping mode is auto

== ============ == ========== ============== ==================================
#  PCI or Name  Id Interface  Selected cores NIC full name
== ============ == ========== ============== ==================================
1  0000:03:00.0 0  ens1f0                    Intel Corporation 82599ES 10-Gigabi
2  0000:03:00.1 0  ens1f1                    Intel Corporation 82599ES 10-Gigabi
3  0000:06:00.0 0  mgmt0                     Intel Corporation I350 Gigabit Netw
4  0000:06:00.1 0  enp6s0f1                  Intel Corporation I350 Gigabit Netw
5  0000:83:00.0 0  ens7f0                    Intel Corporation Ethernet Controll
6  0000:83:00.1 0  ens7f1                    Intel Corporation Ethernet Controll
7  0000:85:00.0 0  ens5f0     auto: 15,31    Intel Corporation 82599ES 10-Gigabi
8  0000:85:00.1 0  ens5f1                    Intel Corporation 82599ES 10-Gigabi
== ============ == ========== ============== ==================================

A - Add a virtual device
C - Switch to manual core/port mapping mode
M - Set the fast path core mask. Current value is "auto" (7,15,23,31)
E - Manually configure FP_PORTS. Current value is "0000:03:00.0"
```

(continues on next page)

(continued from previous page)

```
B - Back

Enter selection [B]: m
```

Set its value to `10,26`. The syntax is detailed below:

```
Set the fast path core mask
===========================

## FP_MASK defines which logical cores run the fast path.
##
## The value can be:
## - 'auto' or commented out: on a Turbo Router, all cores are enabled, except
##   the first one. On a Virtual Accelerator, one physical core per node is
##   enabled.
## - a list of logical cores ranges.
##   Example: "1-4,6,8" means logical cores 1,2,3,4,6,8.
## - an hexadecimal mask starting with '0x': it represents the mask of logical
##   cores to be enabled.
##   Example: "0x1e" means logical cores 1,2,3,4.
##
## Note: the core 0 is usually reserved for Linux processes and DPVI, so
## it's not advised to use it in FP_MASK.
##
## In expert mode (EXPERT_FPCONF=on), this parameter is mandatory and must not
## be auto.
##

Enter value [auto]: 10,26
```

The core to port mapping is updated:

```
Fast path port selection
========================

Core/port mapping mode is auto

== ============ == ========== ============== ==================================
#  PCI or Name  Id Interface  Selected cores NIC full name
== ============ == ========== ============== ==================================
1  0000:03:00.0 0  ens1f0                    Intel Corporation 82599ES 10-Gigabi
2  0000:03:00.1 0  ens1f1                    Intel Corporation 82599ES 10-Gigabi
3  0000:06:00.0 0  mgmt0                     Intel Corporation I350 Gigabit Netw
```

(continues on next page)

(continued from previous page)

```
4  0000:06:00.1 0  enp6s0f1                  Intel Corporation I350 Gigabit Netw
5  0000:83:00.0 0  ens7f0                    Intel Corporation Ethernet Controll
6  0000:83:00.1 0  ens7f1                    Intel Corporation Ethernet Controll
7  0000:85:00.0 0  ens5f0     auto: 10,26    Intel Corporation 82599ES 10-Gigabi
8  0000:85:00.1 0  ens5f1                    Intel Corporation 82599ES 10-Gigabi
== ============ == ========== ============== ===================================

A - Add a virtual device
C - Switch to manual core/port mapping mode
M - Set the fast path core mask. Current value is "10,26"
E - Manually configure FP_PORTS. Current value is "0000:83:00.0"
```

4. Reduce the amount of memory reserved for the VM. By default, 4GB per socket is reserved but 1GB on the second socket is enough in our use case.

   Enter `Advanced configuration`:

```
Fast path configuration
=======================

1 - Select fast path ports and polling cores
2 - Select a hardware crypto accelerator
3 - Advanced configuration
4 - Advanced plugin configuration
5 - Display configuration

S - Save configuration and exit
Q - Quit

Enter selection [S]: 3
```

   Ask to modify the amount of memory reserved for VMs:

```
Advanced configuration
======================

1 - Set the amount of fast path memory (current value is auto)
2 - Enable/disable fast path memory allocation (current value is on)
3 - Set the number of mbufs (current value is auto)
4 - Enable/disable fast path offloads (current value is auto)
5 - Set the number of memory channels (current value is auto)
6 - Set the hugepage directory (current value is /mnt/huge)
7 - Set the amount of VM memory (current value is auto)

B - Back
```

(continues on next page)

(continued from previous page)

```
Enter selection [B]: 7
```

Set its value to `0,1024`. The syntax is detailed below:

```
Set amount of memory reserved for Virtual Machines
==================================================

## VM_MEMORY defines how much memory from the hugepages to allocate for
## virtual machines.
##
## When running the fast path as a host managing VMs, the fast path
## startup script is able to reserve additional memory stored in huge
## pages. This memory can be used by Qemu or libvirt for the virtual
## machines.
##
## The value can be:
## - auto or commented out: on a Virtual Accelerator, 4GB per socket will
##   be reserved, on other products no VM memory will be reserved.
## - an integer: it represents the amount of memory in MB to reserve
##   for VMs. This amount will be spread equally on all NUMA nodes.
##   Example: "4096" asks to reserve 4GB for the virtual machines, distributed
##   on all the NUMA nodes of the machine (2GB per node if the machine has
##   2 nodes).
## - a list of integers, representing the amount of memory in MB
##   to reserve on each NUMA node.
##   Example: "2048,2048" asks to reserve 2GB on node0 and 2GB on node1
##   in huge pages for the virtual machines.
##
## In expert mode (EXPERT_FPCONF=on), the parameter is mandatory and its format
## must be a list of integer, one per socket.
##

Enter value [auto]: 0,1024
```

5. The configuration can be saved. It will generate the `/etc/fast-path.env` file:

```
Fast path configuration
=======================

1 - Select fast path ports and polling cores
2 - Select a hardware crypto accelerator
3 - Advanced configuration
4 - Advanced plugin configuration
```

(continues on next page)

(continued from previous page)

```
5 - Display configuration

S - Save configuration and exit
Q - Quit

Enter selection [S]: S
```

6. libvirt does not support the cpuset isolation feature; it has to be disabled in `/etc/cpuset.env`.

```
-#: ${CPUSET_ENABLE:=1}
+: ${CPUSET_ENABLE:=0}
```

7. Start Virtual Accelerator.

```
root@host1:~# service virtual-accelerator start
```

8. Restart the Open vSwitch control plane.

```
root@host1:~# service openvswitch-switch restart
```

9. The hugepages are allocated by Virtual Accelerator at startup and libvirt cannot detect them dynamically. libvirt must be restarted to take the hugepages into account.

```
root@host1:~# service libvirt-bin restart
```

> **Warning:** If you restart Virtual Accelerator, you must restart openvswitch and libvirt (and its VMs) as well.

10. Create a virtio interface to communicate with the VM. The `sockpath` argument will be used in the libvirt XML file later.

```
root@host:~# fp-vdev add tap0 --sockpath=/tmp/pmd-vhost0
devargs:
  profile: endpoint
  sockmode: client
  sockname: /tmp/pmd-vhost0
  txhash: l3l4
  verbose: 0
driver: pmd-vhost
ifname: tap0
rx_cores: all
```


Note: Make sure that the fast path has been started before you create hotplug ports with `fp-vdev` commands

See also:

The 6WINDGate *Fast Path Managing virtual devices* documentation for more information about the `fp-vdev` command.

host2

Follow on `host2` the same configuration steps as on `host1`:

- Using the configuration wizard:
  - select a physical port and select the fast path cores
  - change the amount of VM memory
  - save the resulting configuration file
- Disable the cpuset feature in libvirt
- Start or restart services:
  - restart Open vSwitch
  - start Virtual Accelerator
  - restart libvirt

Hosts and VMs configuration

VM creation on both hosts (if needed)

If you don't have a VM ready, you can use a cloud image. See *VM Creation* to create one VM on each host with the following libvirt configuration sections:

- hostname `vm1` on `host1` and `vm2` on `host2`

```
<name>vm1</name>
```

and

```
<name>vm2</name>
```

- one `vhost-user` interface:

```
<interface type='vhostuser'>
   <source type='unix' path='/tmp/pmd-vhost0' mode='server'/>
   <model type='virtio'/>
</interface>
```

- 1048576 bytes (1GB) of memory:

```
<memory>1048576</memory>
```

and

```
<numa>
    <cell id="0" cpus="0" memory="1048576" memAccess="shared"/>
</numa>
```

- Setup memory to be on socket 1:

```
<numatune>
<!-- adapt to set the host node where hugepages are taken -->
  <memory mode='strict' nodeset='1'/>
</numatune>
```

Configuration of host1 and vm1

Now that we have access to the VM, we can setup Linux with the configuration needed for `iperf`.

1. Inside the VM, set interface up and add address on `11.0.0.0/8` subnet.

```
root@vm1:~# ip link set eth1 up
root@vm1:~# ip addr add 11.5.0.105/8 dev eth1
```

2. On `host1`, check the name of the fast path interfaces at the end of the port description in the `fp-shmem-ports -d` output.

```
root@host1:~# fp-shmem-ports -d
core freq : 2693510304
offload : enabled
vxlan ports :
port 4789 (set by user)
port 8472 (set by user)
port 0: ens5f0 mac 90:e2:ba:29:df:58 driver rte_ixgbe_pmd GRO timeout 10us
RX queues: 2 (max: 128)
TX queues: 2 (max: 64)
RX vlan strip off
RX IPv4 checksum on
```

(continues on next page)

(continued from previous page)

```
RX TCP checksum on
RX UDP checksum on
GRO on
LRO off
TX vlan insert on
TX IPv4 checksum on
TX TCP checksum on
TX UDP checksum on
TX SCTP checksum on
TSO on
port 1: tap0 mac 02:09:c0:ea:ef:37 driver pmd-vhost
   (args sockmode=client,sockname=/tmp/pmd-vhost0) GRO timeout 10us
RX queues: 1 (max: 1)
TX queues: 2 (max: 64)
RX TCP checksum on
RX UDP checksum on
GRO on
LRO on
TX TCP checksum on
TX UDP checksum on
TSO on
```

Note: `tap0` is the vhost port bound to the interface of `vm1` and `ens5f0` is the host external interface.

3. On `host1`, set interfaces up, add an IP address to the host interface and tune its MTU

```
root@host1:~# ip link set ens5f0 up
root@host1:~# ip link set tap0 up
root@host1:~# ip addr add 10.200.0.5/24 dev ens5f0
root@host1:~# ip link set mtu 1600 dev ens5f0
```

Note: We need to increase MTU size so that encapsulated frames fit in without causing additional fragmentation

4. On `host1`, create two chained OVS bridges. The first one (br-int) connects the vhost interface, the second one (br-tun) encapsulates the traffic in a VXLAN tunnel starting at the host interface.

```
root@host1:~# ovs-vsctl add-br br-int
root@host1:~# ovs-vsctl add-port br-int patch-tun -- set Interface \
  patch-tun type=patch options:peer=patch-int
root@host1:~# ovs-vsctl add-br br-tun
```

(continues on next page)

(continued from previous page)

```
root@host1:~# ovs-vsctl add-port br-tun patch-int -- set Interface \
  patch-int type=patch options:peer=patch-tun
root@host1:~# ip link set up dev tap0
root@host1:~# ovs-vsctl add-port br-int tap0
root@host1:~# ovs-vsctl add-port br-tun vxlan10 -- set Interface vxlan10 \
  type=vxlan options:remote_ip=10.200.0.6 options:local_ip=10.200.0.5 \
  options:out_key=flow
root@host1:~# ip link set up dev br-tun
```

Configuration of host2 and vm2

Now that we have access to the VM, we can setup Linux with the configuration needed for `iperf`.

1. Inside the VM, set interface up and add address on `11.0.0.0/8` subnet.

```
root@vm2:~# ip link set eth1 up
root@vm2:~# ip addr add 11.6.0.106/8 dev eth1
```

2. On `host2`, check the name of the fast path interfaces at the end of the port description in the `fp-shmem-ports -d` output.

```
root@host2:~# fp-shmem-ports -d
core freq : 2693520583
offload : enabled
vxlan ports :
port 4789 (set by user)
port 8472 (set by user)
port 0: ens5f0 mac 00:1b:21:74:5b:58 driver rte_ixgbe_pmd GRO timeout 10us
RX queues: 2 (max: 128)
TX queues: 2 (max: 64)
RX vlan strip off
RX IPv4 checksum on
RX TCP checksum on
RX UDP checksum on
GRO on
LRO off
TX vlan insert on
TX IPv4 checksum on
TX TCP checksum on
TX UDP checksum on
TX SCTP checksum on
TSO on
port 1: tap0 mac 02:09:c0:88:f7:5d driver pmd-vhost
```

(continues on next page)

(continued from previous page)

```
   (args sockmode=client,sockname=/tmp/pmd-vhost0) GRO timeout 10us
RX queues: 1 (max: 1)
TX queues: 2 (max: 64)
RX TCP checksum on
RX UDP checksum on
GRO on
LRO on
TX TCP checksum on
TX UDP checksum on
TSO on
```

Note: `tap0` is the vhost port bound to the interface of `vm2` and `ens5f0` is the host external interface.

3. On `host2`, set interfaces up, add an IP address to the host interface and tune its MTU

```
root@host2:~# ip link set ens5f0 up
root@host2:~# ip link set tap0 up
root@host2:~# ip addr add 10.200.0.6/24 dev ens5f0
root@host2:~# ip link set mtu 1600 dev ens5f0
```

Note: We need to increase MTU size so that encapsulated frames fit in without causing additional fragmentation

4. On `host2`, create two chained OVS bridges. The first one (`br-int`) connects the vhost interface, the second one (`br-tun`) encapsulates the traffic in a VXLAN tunnel starting at the host interface.

```
root@host2:~# ovs-vsctl add-br br-int
root@host2:~# ovs-vsctl add-port br-int patch-tun -- set Interface \
  patch-tun type=patch options:peer=patch-int
root@host2:~# ovs-vsctl add-br br-tun
root@host2:~# ovs-vsctl add-port br-tun patch-int -- set Interface \
  patch-int type=patch options:peer=patch-tun
root@host2:~# ip link set up dev tap0
root@host2:~# ovs-vsctl add-port br-int tap0
root@host2:~# ovs-vsctl add-port br-tun vxlan10 -- set Interface vxlan10 \
  type=vxlan options:remote_ip=10.200.0.5 options:local_ip=10.200.0.6 \
  options:out_key=flow
root@host2:~# ip link set up dev br-tun
```


Testing

We can send traffic from vm1 to vm2 and check that each fast path switches it to and from the VM. First, let's do a ping to check the setup.

1. Reset the fast path statistics first on each host.

```
root@host:~# fp-cli stats-reset
```

2. Ping the vm1 address from vm2.

```
root@vm2:~# ping 11.5.0.105
PING 11.5.0.105 (11.5.0.105) 56(84) bytes of data.
64 bytes from 11.5.0.105: icmp_seq=1 ttl=64 time=5.27 ms
64 bytes from 11.5.0.105: icmp_seq=2 ttl=64 time=0.465 ms
64 bytes from 11.5.0.105: icmp_seq=3 ttl=64 time=0.442 ms
[...]
```

3. During traffic, you can check that the flows are available in the kernel on each host.

```
root@host1:~# ovs-dpctl dump-flows
recirc_id(0),tunnel(),in_port(3),eth(src=52:54:00:ee:fb:f5,
   dst=52:54:00:e8:0b:95),eth_type(0x0800),ipv4(tos=0/0x3,frag=no),
   packets:23, bytes:2254, used:0.045s, actions:set(tunnel(
   tun_id=0x0,src=10.200.0.5,dst=10.200.0.6,ttl=64,flags(df|key))),4
recirc_id(0),tunnel(tun_id=0x0,src=10.200.0.6,dst=10.200.0.5,ttl=64,
   flags(-df-csum+key)),in_port(4),skb_mark(0),eth(src=52:54:00:e8:0b:95,
   dst=52:54:00:ee:fb:f5),eth_type(0x0800), ipv4(frag=no),
   packets:23, bytes:3082, used:0.045s, actions:3

root@host2:~# ovs-dpctl dump-flows
recirc_id(0),tunnel(),in_port(3),eth(src=52:54:00:e8:0b:95,
   dst=52:54:00:ee:fb:f5),eth_type(0x0800),ipv4(tos=0/0x3,frag=no),
   packets:17, bytes:1666, used:0.000s, actions:set(tunnel(
   tun_id=0x0,src=10.200.0.6,dst=10.200.0.5,ttl=64,flags(df|key))),4
recirc_id(0),tunnel(tun_id=0x0,src=10.200.0.5,dst=10.200.0.6,ttl=64,
   flags(-df-csum+key)),in_port(4),skb_mark(0),eth(src=52:54:00:ee:fb:f5,
   dst=52:54:00:e8:0b:95),eth_type(0x0800),ipv4(frag=no),
   packets:17, bytes:2278, used:0.000s, actions:3
```

4. The statistics are increased in the fast path, showing that the fast path processed the packets.

```
root@host1:~# fp-cli fp-vswitch-stats
  flow_not_found:7
  output_ok:81
  set_tunnel_id:41
```

(continues on next page)

(continued from previous page)

```
root@host2:~# fp-cli fp-vswitch-stats
  flow_not_found:7
  output_ok:85
  set_tunnel_id:42
```

Note: The flow_not_found statistics is increased for the first packets of each flow, that are sent to Linux because they don't match any known flow in the fast path. Linux receives the packets and sends them to the ovs-vswitchd daemon (it is the standard Linux processing). The daemon creates a flow in the OVS kernel data plane. The flow is automatically synchronized to the fast path, and the next packets of the flow are processed by the fast path.

Now that we checked the setup, we can try `iperf`.

1. Reset the fast path statistics first on each host.

```
root@host1:~# fp-cli stats-reset
```

```
root@host2:~# fp-cli stats-reset
```

2. Start the `iperf` server on `vm1`.

```
root@vm1:~# iperf -s
```

3. Start `iperf` client on `vm2`.

```
root@vm2:~# iperf -c 11.5.0.105 -i 10
------------------------------------------------------------
Client connecting to 11.5.0.105, TCP port 5001
TCP window size: 85.0 KByte (default)
------------------------------------------------------------
[  3] local 11.6.0.106 port 57649 connected with 11.5.0.105 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  10.6 GBytes  9.07 Gbits/sec
[  3]  0.0-10.0 sec  10.6 GBytes  9.07 Gbits/sec
```

4. During traffic, you can check the fast path CPU usage on each host.

```
root@host1:~# fp-cpu-usage
Fast path CPU usage:
cpu: %busy     cycles   cycles/packet
 10:  100%  538671904            3210
 26:   <1%    5098764               0
average cycles/packets received from NIC: 3240 (543770668/167807)
```

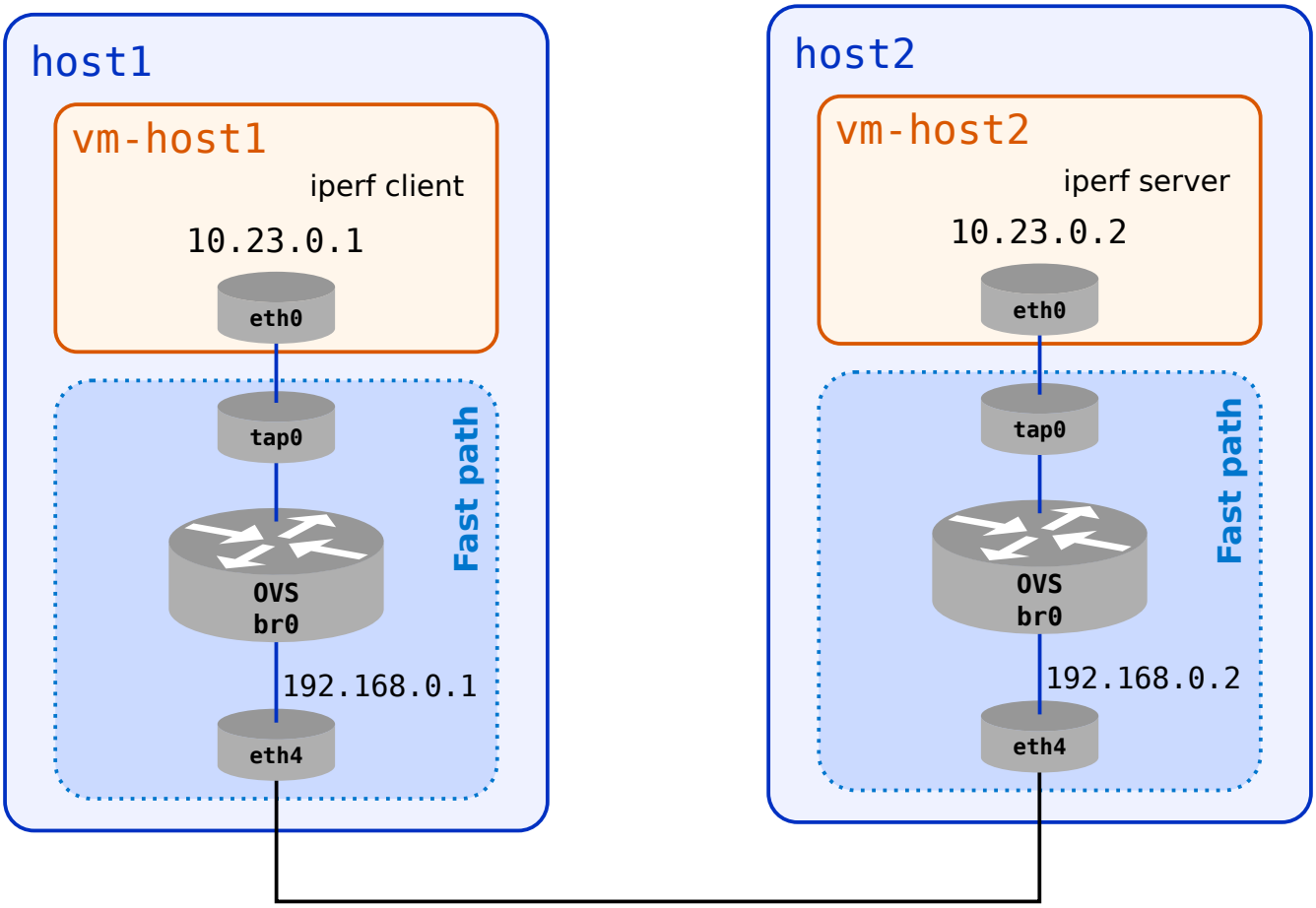
5. After traffic, you can check the fast path statistics on each host.

```
root@host1:~# fp-cli fp-vswitch-stats
  flow_not_found:4
  output_ok:801241
  set_tunnel_id:347493
```

1.5.3 Two VMs, 4 logical cores, Virtio multiqueue NICs, offloading enabled, with Open vSwitch

Two VMs on different hypervisors exchange traffic on a private subnet. Each hypervisor uses Open vSwitch bridges to connect its physical interface and the VM virtual interface. Each VM runs on two physical cores (four logical cores). Virtual Accelerator runs on two physical cores too (logical cores 10, 11, 22, 23).

`iperf` TCP is started between the two spawned VMs.



We use the following prompts in this section:

```
root@hosts:~#      #=> all hosts
root@vms:~#        #=> all vms
root@host1:~#      #=> host1
root@host2:~#      #=> host2
root@vm-host1:~#   #=> VM on host1
root@vm-host2:~#   #=> VM on host2
```

In this usecase, physical machines used are the same.

Virtual Accelerator configuration

1. Show the default fast path configuration:

```
root@hosts:~# fast-path.sh config
Info: no configuration file /etc/fast-path.env for fast-path.sh, using defaults
Configuring Fast Path...
Fast path configuration info
============================

  Selected ethernet card
  ----------------------

  Intel Corporation Ethernet 10G 2P X520 Adapter (rev 01) PCI card (not mounted␣
↪on any eth) with cores 11,23
  Intel Corporation Ethernet 10G 2P X520 Adapter (rev 01) PCI card mounted on␣
↪eth4 with cores 11,23
  Intel Corporation Ethernet Controller 10-Gigabit X540-AT2 (rev 01) PCI card␣
↪mounted on eth5 with cores 11,23
  Intel Corporation Ethernet Controller 10-Gigabit X540-AT2 (rev 01) PCI card␣
↪mounted on eth6 with cores 11,23
```

   The logical cores 11 and 23 are located on the first socket, they poll the ports located on the first socket.

2. If you are connected to a machine using a network interface supported by the fast path (for instance using SSH), you should avoid the fast path to take control of this port and thus deleting its network configuration, which would break the connection.

```
root@hosts:~# ethtool -i mgmt0
[...]
bus-info: 0000:01:00.0
[...]

root@hosts:~# FP_PORTS="all -0000:01:00.0" fast-path.sh config --update
```

3. Update `/etc/fast-path.env` in order to use 4 cores:

```
root@hosts:~# FP_MASK="10,11,22,23" fast-path.sh config --update
```

4. libvirt does not support the cpuset isolation feature; it has to be disabled in `/etc/cpuset.env`:

```
-#: ${CPUSET_ENABLE:=1}
+: ${CPUSET_ENABLE:=0}
```

5. Start Virtual Accelerator:

```
root@hosts:~# service virtual-accelerator start
```

6. Restart the Open vSwitch control plane.

```
root@hosts:~# service openvswitch-switch restart
```

7. The hugepages are allocated by Virtual Accelerator at startup and libvirt cannot detect them dynamically. libvirt must be restarted to take the hugepages into account.

```
root@hosts:~# service libvirt-bin restart
```

Warning: If you restart Virtual Accelerator, you must restart openvswitch and libvirt (and its VMs) as well.

Linux configuration

1. VM configuration
    1. Add a new virtio interface on each host to communicate with the VM:

       See also:

       The *Hotplug a virtual port* section of the guide.

       The `sockpath` argument will be used in the libvirt XML file later.

       These new ports will be polled by fast path logical cores located on the same socket as the VM. The number of fast path logical cores polling a port depends on the number of virtual rings. Each virtual ring is polled by one fast path logical core. If there are less fast path logical cores than virtual rings, some fast path logical cores poll several virtual rings. The number of virtual rings is configured in the VM using `ethtool -L`.

```
root@hosts:~# fp-vdev add tap0 --sockpath=/tmp/pmd-vhost0
devargs:
  profile: endpoint
  sockmode: client
  sockname: /tmp/pmd-vhost0
```

(continues on next page)

(continued from previous page)

```
  txhash: l3l4
  verbose: 0
driver: pmd-vhost
ifname: tap0
rx_cores: all
root@hosts:~# ip link set dev tap0 up
```

Note: Make sure that the fast path has been started before you create hotplug ports with `fp-vdev` commands

---

See also:

The 6WINDGate *Fast Path Managing virtual devices* documentation for more information about the `fp-vdev` command.

2. Create VMs configuration files using libvirt:

   Create `vm-host1.xml` file on `host1` and `vm-host2.xml` file on `host2` using the *VM Creation* section of this guide.

3. Launch VMs:

   1. On `host1`:

      ```
      root@host1:~# virsh create vm-host1.xml
      ```

   2. On `host2`:

      ```
      root@host2:~# virsh create vm-host2.xml
      ```

4. Then, connect to the VMs:

   See also:

   The *Connection to the VM* section of this guide

   ```
   root@hosts:~# telnet 127.0.0.1 10000
   ```

2. Enable multiqueue on VMs:

```
root@vms:~# ethtool -L eth0 combined 4

root@vms:~# ethtool -l eth0
Channel parameters for eth0:
Pre-set maximums:
RX:             0
TX:             0
```

(continues on next page)

(continued from previous page)

```
Other:          0
Combined:       4
Current hardware settings:
RX:             0
TX:             0
Other:          0
Combined:       4
```

Then check the number of queues:

```
root@hosts:~# fp-cli dpdk-core-port-mapping
port 0:  (rte_ixgbe_pmd)
  nb_rxq=4 nb_txq=4 rxq_shared=0 txq_shared=0
  rxq0=c10 rxq1=c11 rxq2=c22 rxq3=c23
  txq0=c10 txq1=c11 txq2=c22 txq3=c23
port 1: eth4 (rte_ixgbe_pmd)
  nb_rxq=4 nb_txq=4 rxq_shared=0 txq_shared=0
  rxq0=c10 rxq1=c11 rxq2=c22 rxq3=c23
  txq0=c10 txq1=c11 txq2=c22 txq3=c23
port 2: eth5 (rte_ixgbe_pmd)
  nb_rxq=4 nb_txq=4 rxq_shared=0 txq_shared=0
  rxq0=c10 rxq1=c11 rxq2=c22 rxq3=c23
  txq0=c10 txq1=c11 txq2=c22 txq3=c23
port 3: eth6 (rte_ixgbe_pmd)
  nb_rxq=4 nb_txq=4 rxq_shared=0 txq_shared=0
  rxq0=c10 rxq1=c11 rxq2=c22 rxq3=c23
  txq0=c10 txq1=c11 txq2=c22 txq3=c23
```

3. Networking setup:

    1. Specific networking setup on `host1`:

    ```
    root@host1:~# ip link set dev eth4 up
    root@host1:~# ip addr add 192.168.0.1/24 dev eth4
    ```

    2. Specific networking setup on `host2`:

    ```
    root@host2:~# ip link set dev eth4 up
    root@host2:~# ip addr add 192.168.0.2/24 dev eth4
    ```

    3. Open vSwitch setup on all hosts:

    ```
    root@hosts:~# ovs-vsctl add-br br0
    root@hosts:~# ovs-vsctl add-port br0 tap0
    root@hosts:~# ovs-vsctl add-port br0 eth4
    ```

(continues on next page)

(continued from previous page)

```
root@hosts:~# ip link set dev br0 up
```

4. Specific networking setup on vm-host1:

```
root@vm-host1:~# ip link set dev eth0 up
root@vm-host1:~# ip addr add 10.23.0.1/24 dev eth0
```

5. Specific networking setup on vm-host2:

```
root@vm-host2:~# ip link set dev eth0 up
root@vm-host2:~# ip addr add 10.23.0.2/24 dev eth0
```

Testing

1. Check the connectivity between vm-host1 and vm-host2:

```
root@vm-host1:~# ping 10.23.0.2 -i 0.01 -c 1000
[...]
64 bytes from 10.23.0.2: icmp_seq=998 ttl=64 time=0.095 ms
64 bytes from 10.23.0.2: icmp_seq=999 ttl=64 time=0.095 ms
64 bytes from 10.23.0.2: icmp_seq=1000 ttl=64 time=0.096 ms

--- 10.23.0.2 ping statistics ---
1000 packets transmitted, 1000 received, 0% packet loss, time 9989ms
rtt min/avg/max/mdev = 0.090/0.095/0.873/0.026 ms

root@host1:~# fp-cli stats
==== interface stats:
lo-vr0 port:254
mgmt0-vr0 port:254
eth1-vr0 port:254
eth2-vr0 port:254
eth3-vr0 port:254
fpn0-vr0 port:254
eth4-vr0 port:1
eth5-vr0 port:2
eth6-vr0 port:3
eth0-vr0 port:0
tap0-vr0 port:4
ovs-system-vr0 port:254
br0-vr0 port:254
==== global stats:
==== exception stats:
```

(continues on next page)

(continued from previous page)

```
  LocalBasicExceptions:4
  LocalExceptionClass:
  FPTUN_EXC_SP_FUNC:4
  LocalExceptionType:
  FPTUN_BASIC_EXCEPT:4
==== IPv4 stats:
==== arp stats:
==== IPv6 stats:
==== L2 stats:
==== fp-vswitch stats:
  flow_not_found:4
  output_ok:1998
```

Note: The flow_not_found statistics is increased for the first packets of each flow, that are sent to Linux because they don't match any known flow in the fast path. Linux receives the packets and sends them to the ovs-vswitchd daemon (it is the standard Linux processing). The daemon creates a flow in the OVS kernel data plane. The flow is automatically synchronized to the fast path, and the next packets of the flow are processed by the fast path.

---

2. During traffic, you can check that the flows are available in the kernel on each host:

```
root@host1:~# ovs-dpctl dump-flows
in_port(2),eth(src=52:54:00:50:87:e5,dst=52:54:00:d5:75:6e),eth_type(0x0800),
↪ipv4(frag=no)
           packets:191535, bytes:12341830191, used:0.000s, actions:3
in_port(3),eth(src=52:54:00:d5:75:6e,dst=52:54:00:50:87:e5),eth_type(0x0800),
↪ipv4(frag=no),
           packets:240937, bytes:15983383, used:0.000s, actions:2

root@host2:~# ovs-dpctl dump-flows
in_port(2),eth(src=52:54:00:d5:75:6e,dst=52:54:00:50:87:e5),eth_type(0x0800),
↪ipv4(frag=no),
           packets:253909, bytes:16840441, used:0.000s, actions:3
in_port(3),eth(src=52:54:00:50:87:e5,dst=52:54:00:d5:75:6e),eth_type(0x0800),
↪ipv4(frag=no),
           packets:421568, bytes:12983559073, used:0.000s, actions:2
```

3. Reset the fast path statistics first on each host:

```
root@hosts:~# fp-cli stats-reset
```

4. Start the `iperf` server on `vm-host2`:

```
root@vm-host2:~# iperf3 -s
```

5. Start the `iperf` TCP client on `vm-host1`:

```
root@vm-host1:~# iperf3 -c 10.23.0.2
Connecting to host 10.23.0.2, port 5201
[  4] local 10.23.0.1 port 60743 connected to 10.23.0.2 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  1.10 GBytes  9.44 Gbits/sec  155   1.12 MBytes
[  4]   1.00-2.00   sec  1.10 GBytes  9.41 Gbits/sec   90   1.14 MBytes
[  4]   2.00-3.00   sec  1.09 GBytes  9.41 Gbits/sec   90   1.16 MBytes
[  4]   3.00-4.00   sec  1.10 GBytes  9.42 Gbits/sec  122   1.20 MBytes
[  4]   4.00-5.00   sec  1.10 GBytes  9.42 Gbits/sec  113   1.07 MBytes
[  4]   5.00-6.00   sec  1.10 GBytes  9.42 Gbits/sec   90   1.12 MBytes
[  4]   6.00-7.00   sec  1.10 GBytes  9.41 Gbits/sec   90   1.14 MBytes
[  4]   7.00-8.00   sec  1.10 GBytes  9.41 Gbits/sec  112   1020 KBytes
[  4]   8.00-9.00   sec  1.09 GBytes  9.37 Gbits/sec  737    885 KBytes
[  4]   9.00-10.00  sec  1.10 GBytes  9.42 Gbits/sec  127    901 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
  [ ID] Interval           Transfer     Bandwidth       Retr
  [  4]   0.00-10.00  sec  11.0 GBytes  9.41 Gbits/sec  1726             sender
  [  4]   0.00-10.00  sec  11.0 GBytes  9.41 Gbits/sec                  receiver

iperf Done.
```

6. During traffic, you can check the fast path CPU usage on each host:

```
root@host1:~# fp-cpu-usage
Fast path CPU usage:
cpu: %busy     cycles   cycles/packet
 10:    9%   50198333             13794
 11:    1%    6972454             3309
 22:    8%   46654144             20855
 23:    9%   51496212             9958

root@host2:~# fp-cpu-usage
Fast path CPU usage:
cpu: %busy     cycles   cycles/packet
 10:   11%   60073363             2517
 11:   12%   64039855             2635
 22:   24%  128062956             2264
 23:   30%  157335765             2344
```

What is important to see is that all the cores are working. The traffic is spread on all the cores, even if the distribution is not fair in our case.

7. After traffic, you can check the fast path statistics:

```
root@hosts:~# fp-cli stats
==== interface stats:
lo-vr0 port:254
mgmt0-vr0 port:254
eth1-vr0 port:254
eth2-vr0 port:254
eth3-vr0 port:254
fpn0-vr0 port:254
eth4-vr0 port:1
eth5-vr0 port:2
eth6-vr0 port:3
eth0-vr0 port:0
tap0-vr0 port:4
ovs-system-vr0 port:254
br0-vr0 port:254
  ifs_ipackets:22
  ifs_ibytes:2013
==== global stats:
==== exception stats:
  LocalBasicExceptions:51
  LocalFPTunExceptions:22
  LocalExceptionClass:
  FPTUN_EXC_SP_FUNC:29
  FPTUN_EXC_ETHER_DST:20
  FPTUN_EXC_IP_DST:2
  LocalExceptionType:
  FPTUN_BASIC_EXCEPT:29
  FPTUN_IPV6_INPUT_EXCEPT:2
  FPTUN_ETH_INPUT_EXCEPT:20
==== IPv4 stats:
==== arp stats:
==== IPv6 stats:
==== L2 stats:
==== fp-vswitch stats:
  flow_not_found:28
  flow_pullup_too_small:1
  output_ok:6731979
```

Packets are correctly forwarded by the fast path.

Appendix

1.5.4 VM Creation

Libvirt

If you don't have a VM ready, you can use a cloud image. The next steps explain how to use an Ubuntu trusty cloud-image. You can skip those steps if you already have one.

See also:

The cloud-init documentation (https://cloudinit.readthedocs.org/en/latest/topics/datasources.html#no-cloud).

1. Write cloud-init user-data and meta-data files to setup the password and set the hostname (change vm1 according to your needs).

```
root@host:~# cat << EOF > user-data
#cloud-config
ssh_pwauth: True
chpasswd:
  list: |
    ubuntu:ubuntu
  expire: False
EOF

root@host:~# cat << EOF > meta-data
instance-id: vm1
local-hostname: vm1
EOF
```

2. Build an iso image containing the meta-data and user-data and put it in the libvirt `images` directory.

```
root@host:~# apt-get install -y genisoimage
root@host:~# genisoimage -output seed.iso -volid cidata \
                         -joliet -rock user-data meta-data
root@host:~# cp seed.iso /var/lib/libvirt/images/
```

3. Download the latest Ubuntu cloud image into the libvirt images directory.

```
root@host:~# url=https://cloud-images.ubuntu.com/trusty/current
root@host:~# curl $url/trusty-server-cloudimg-amd64-disk1.img \
                  -o /var/lib/libvirt/images/ubuntu-14.04.img
```

Now that you have a VM template, we can start the virtual machine using libvirt.

1. Create an XML domain file named <your_vm_hostname>.xml (according to your VM hostname), containing at least:

   - a libvirt domain name (<name>) adapted to the VM hostname

- a Virtio interface for management (<interface type='user'>) configuration wizard (<interface type='vhostuser'>)
- two disks, one for the cloud-init iso, one for the cloud-image iso (<disk type='file' device='disk'>)
- serial port forwarding to port 1000 (<serial type='tcp'> and <console type='tcp'>)
- hugepages (<numa> and <hugepages>)
- as much vhost-user interfaces as you have sockets displayed in the fast path configuration wizard

See also the libvirt XML documentation: https://libvirt.org/formatdomain.html

The template for this file is the following:

```
<domain type='kvm'>
  <name>vm1</name> <!-- change the name according to your needs-->
  <memory>1048576</memory> <!-- adapt to the desired memory size -->
  <os>
    <type>hvm</type>
    <boot dev="hd"/>
  </os>
  <vcpu placement='static'>1</vcpu>
  <vcpupin vcpu='0' cpuset='0'/>
  <cpu>
    <numa>
      <!-- adapt to the desired memory size -->
      <cell id="0" cpus="0" memory="1048576" memAccess="shared"/>
    </numa>
  </cpu>
  <!--

  IN ORDER TO USE MULTIQUEUE, MANY CORES ARE NECESSARY
  REPLACE THE <vcpu>/<vcpupin>/<cpu> section by the following template where:
    - N in the desired number of vcpus
    - A, B, C are the desired host core IDs on which vcpus are pinned
    - E, F, G are other host core IDs on which emulator threads are pinned
    - N = X * Y * Z, for example <topology sockets='1' cores='2' threads='2' /> for 4
↪vcpus

  <vcpu placement='static'>N</vcpu>
  <cputune>
    <vcpupin vcpu='0' cpuset='A'/>
    <vcpupin vcpu='1' cpuset='B'/>
    ...
    <vcpupin vcpu='N' cpuset='C'/>
    <emulatorpin cpuset='E, F, G, ...'/>
```

(continues on next page)

(continued from previous page)

```
  </cputune>
  <cpu>
    <topology sockets='X' cores='Y' threads='Z' />
    <numa>
      <cell id="0" cpus="0-N" memory="1048576" memAccess="shared"/>
    </numa>
  </cpu>

  -->
  <numatune>
    <!-- adapt to set the host node where hugepages are taken -->
    <memory mode='strict' nodeset='0'/>
  </numatune>
  <memoryBacking>
    <hugepages>
      <page size="2048" unit="KiB"/>
    </hugepages>
  </memoryBacking>
  <features>
    <acpi/>
  </features>
  <devices>
    <disk type='file' device='disk'>
      <source file='/var/lib/libvirt/images/seed.iso'/>
      <target dev='vdb' bus='virtio'/>
    </disk>
    <disk type='file' device='disk'>
      <driver type='qcow2' cache='none'/>
      <source file='/var/lib/libvirt/images/ubuntu-14.04.img'/>
      <target dev='vda' bus='virtio'/>
    </disk>
    <interface type='user'>
      <model type='virtio'/>
    </interface>
    <!--

    INSERT HERE YOUR VHOST USER INTERFACES

     These interfaces are declared with the following template where:
       - N is the desired number of queues (optional, only in multiqueue cases)

    <interface type='vhostuser'>
      <source type='unix' path='/path/to/the/vhostuser/socket' mode='server'/>
      <model type='virtio'/>
```

(continues on next page)

(continued from previous page)

```
      <driver queues='N'>
    </interface>

    -->
    <serial type='tcp'>
      <source mode='bind' host='0.0.0.0' service='10000'/>
      <protocol type='raw'/>
      <target port='0'/>
      <alias name='serial0'/>
    </serial>
    <serial type='pty'>
      <source path='/dev/pts/4'/>
      <target port='1'/>
      <alias name='serial1'/>
    </serial>
    <console type='tcp'>
      <source mode='bind' host='0.0.0.0' service='10000'/>
      <protocol type='raw'/>
      <target type='serial' port='0'/>
      <alias name='serial0'/>
    </console>
  </devices>
</domain>
```

2. Start the virtual machine using the previously created XML file.

```
root@host:~# virsh create <your_vm_hostname>.xml
```

3. Forward the ssh port to local port 2222.

```
root@host:~# virsh qemu-monitor-command --hmp <your_vm_hostname> \
                                        'hostfwd_add ::2222-:22'
```

OpenStack

You can use your VM or download a cloud image like described in the previous section.

See also:

The *Libvirt* section of this guide

For this example, a cloud image is used.

1. Download the cloud image:

```
root@host:~# url=https://cloud-images.ubuntu.com/bionic/current
root@host:~# curl $url/bionic-server-cloudimg-amd64.img \
                  -o /tmp/ubuntu-cloud.img
```

2. Create the glance image to be able to boot this VM in an OpenStack setup:

```
root@host:~# source /root/admin-openrc.sh
root@host:~# openstack image create "ubuntu-virtio" \
                      --file /tmp/ubuntu-cloud.img \
                      --disk-format qcow2 --container-format bare \
                      --property hw_vif_multiqueue_enabled=false \
                      --public
+---------------------------+--------------------------------------+
| Property                  | Value                                |
+---------------------------+--------------------------------------+
| checksum                  | bf0d07acee853f95ff676c95f28aec6e     |
| container_format          | bare                                 |
| created_at                | 2016-04-06T08:41:03Z                 |
| disk_format               | qcow2                                |
| hw_vif_multiqueue_enabled | false                                |
| id                        | f97506ae-7d64-465d-ac74-6effa039b3ec |
| min_disk                  | 0                                    |
| min_ram                   | 0                                    |
| name                      | ubuntu-virtio                        |
| owner                     | 3e3b085aa3ea445f8beabc369963d63b     |
| protected                 | False                                |
| size                      | 320471040                            |
| status                    | active                               |
| tags                      | []                                   |
| updated_at                | 2016-04-06T08:41:24Z                 |
| virtual_size              | None                                 |
| visibility                | public                               |
+---------------------------+--------------------------------------+
```

Note: `hw_vif_multiqueue_enabled` must be set to false if multiqueue is not enabled.

---

1. Use a cloud-init file when booting a VM with nova

   When booting the VM with nova, the `--user-data` option can be used to provide a local file. It is mainly used to pass a configuration file for cloud-init image. The cloud-init service allows customizing a VM at boot time (like setting a password).

   Example of `cloud.cfg`:

```
---
disable_root: false
ssh_pwauth: True
chpasswd:
  list: |
     root:root
  expire: False
runcmd:
  - sed -i -e '/^PermitRootLogin/s/^.*$/PermitRootLogin yes/' /etc/ssh/sshd_config
  - sed -i -e '/^#UseDNS/s/^.*$/UseDNS no/' /etc/ssh/sshd_config
  - sed -i -e '/^GSSAPIAuthentication/s/^.*$/GSSAPIAuthentication no/' /etc/ssh/
↪sshd_config
  - restart ssh
  - systemctl restart ssh
```

2. Run the nova VM:

```
root@host:~# openstack network create private
root@host:~# openstack subnet create --network private --subnet-range 11.0.0.0/24␣
↪private_subnet
root@host:~# openstack router create router
root@host:~# openstack router add subnet router private_subnet

root@host:~# openstack server create --flavor m1.vm_huge \
      --image ubuntu-virtio \
      --nic net-id=$(openstack network list -- name private (c ID -f value) \
      --user-data cloud.cfg \
      vm1
```

Connection to the VM

1. Serial console can be obtained using telnet on port **10000**.

```
root@host:~# telnet 127.0.0.1 10000
```

Note: In order to get the VM port, use the following command:

```
root@host:~# ps ax | grep -e qemu -e port=
```

---

2. A SSH console is also available.

```
root@host:~# ssh -p 2222 ubuntu@127.0.0.1
```

Note: According to some reports, the cloud init configuration that happens at first boot may sometimes fail to enforce the user and password set in the user-data file. In that case, you won't be able to log in your VM. You might need to patch the image file with some additional script: https://trickycloud.wordpress.com/2013/11/09/default-user-and-password-in-ubuntu-cloud-images/

Installation of iperf and iperf3

Once connected to the VM, install the iperf and iperf3 packages:

```
root@vm:~# apt-get install -y iperf iperf3
```

2. 6WINDGate Modules

2.1 Processor SDK

2.1.1 6WINDGate DPDK

Overview

6WINDGate DPDK provides drivers and libraries for high performance I/Os on Intel and Arm platforms.

6WINDGate DPDK is based on the open source DPDK from dpdk.org (http://dpdk.org), validated, maintained and supported by 6WIND.

Optional add-ons can be added to 6WINDGate DPDK for the support of non-Intel NICs, crypto and vNICs.

Features

Included

Standard features of DPDK from dpdk.org (http://dpdk.org)

- Data Plane Libraries and Optimized NIC Drivers in Linux User Space
- Poll mode drivers (PMD) for direct access to the networking hardware
- Support for multiple Intel NICs
- Scalable performance for multi-core platforms
- Environment Abstraction Layer for portability

6WINDGate DPDK features that are not part of standard DPDK

- Virtio inner checksum offload, LRO, TSO
- Crypto Acceleration Framework

Main unsupported features

- Packet framework libraries
- KNI
- Examples

Refer to `/usr/local/dpdk/<dpdk-target>/.config` for a complete list of supported features.

Additional features available through 6WINDGate DPDK add-ons

Poll Mode Drivers for multi-vendor NICs

- *Mellanox ConnectX-3 EN series PMD*
- *Mellanox ConnectX-4 EN series PMD*
- *Cavium QLogic FastLinQ 4xxxx PMD*

Note: *Cavium QLogic FastLinQ 4xxxx PMD* requires up-to-date Storm firmware version (8.30.12.0). Refer to DPDK documentation (https://doc.dpdk.org/guides-18.11/nics/qede.html#prerequisites) to ensure you have the correct firmware.

Performance acceleration for virtualized networking

- *Virtio Host PMD*

Crypto acceleration modules

- *Intel Multi-Buffer Crypto*
- *Intel Quickassist Crypto*

Supported Platforms

Processors

- Intel Xeon E5-1600/2600/4600 v2 family (Ivy Bridge EP)
- Intel Xeon E5-1600/2600/4600 v3 family (Haswell EP)
- Intel Xeon E5-1600/2600/4600 v4 family (Broadwell EP)
- Intel Xeon E7-2800/4800 v2 family (Ivy Bridge EX)
- Intel Xeon E7-2800/4800 v3 family (Haswell EX)
- Intel Xeon E7-4800/8800 v4 family (Broadwell EX)
- Intel Xeon Platinium/Gold/Silver/Bronze family (Skylake)
- Intel Atom C2000 family for Communications (Rangeley)
- Intel Xeon D-1500 family (Broadwell DE)

Ethernet NICs

- Intel 1G 82575, 82576, 82580, I210, I211, I350, I354 (igb)
- Intel 10G 82598, 82599, X520, X540 (ixgbe)
- Intel 10G/40G X710, XL710, XXV710 (i40e)

- Mellanox 10G/40G Connect-X 3 (mlx4)
- Mellanox 10G/25G/40G/50G/100G Connect-X 4/5 (mlx5)
- Broadcom NetExtreme E-Series (bnxt)

Virtual NICs

- Virtio
- Vhost (only with *Virtio Host PMD* 6WINDGate DPDK add-on)
- Vmxnet3
- AWS ENA
- Azure NetVSC

The PMDs that are not in this list are not supported.

Refer to the Features availability in networking drivers table (https://doc.dpdk.org/guides-18.11/nics/overview.html) for a per-PMD list of supported features.

Dependencies

None.

Usage

See http://dpdk.org/doc/guides/ for more details.

2.1.2 Intel Multi-Buffer Crypto

This module requires a Virtual Accelerator IPsec Application License.

Overview

Intel Multi-Buffer Crypto allows DPDK applications running on Intel platforms to leverage the Intel Multi-Buffer IPsec library.

Features

- Support of Intel Multi-Buffer library leveraging AES-NI (Advanced Encryption Standard - New Instructions), SSE (Streaming SIMD Extensions), AVX (Advanced Vector Extensions), AVX2 (Advanced Vector Extensions 2) and AVX512 (Advanced Vector Extensions 512).
- Encryption algorithms:
  - AES-CBC/CTR (Advanced Encryption Standard - Cipher Block Chaining/CounTeR) (128, 192, 256), AES-GCM (Advanced Encryption Standard - Galois/Counter Mode) (128, 192, 256)
  - NULL
- Authentication algorithms:
  - HMAC-MD5 (Hash Message Authentication Code - Message Digest 5), HMAC-SHA-1 (Hash Message Authentication Code - Secure Hash Algorithm 1), HMAC-SHA-2 (Hash Message Authentication Code - Secure Hash Algorithm 2) (224, 256, 384, 512)
  - AES-XCBC (Advanced Encryption Standard - XOR Cipher Block Chaining) (128)
  - NULL
- Supported platforms

  All Intel processor platforms supported by *6WINDGate DPDK* and supporting AES-NI and at least one of SSE, AVX, AVX2 and AVX512 instructions sets.
- Dynamically loadable plugin for *6WINDGate DPDK*

Dependencies

6WINDGate modules

- *6WINDGate DPDK*

Other dependencies

- Intel Multi-Buffer library IPsec

Usage

Enabling Intel Multi-Buffer Crypto at runtime

1. Specify the `-d librte_ext_crypto_multibuffer.so` argument in the `EAL_OPTIONS` section of the fast path configuration file:

```
: ${EAL_OPTIONS:=-d librte_ext_crypto_multibuffer.so}
```

2.1.3 Virtio Host PMD

Overview

Virtio Host PMD provides a virtual NIC Poll Mode Driver for high performance communication between a host and Virtio guests. The host leverages *6WINDGate DPDK*. Both standard Linux guests running the Linux Virtio driver and guests using *6WINDGate DPDK* running Virtio Guest XEN-KVM PMD are supported.

Virtio Host PMD is usually combined with 6WINDGate *Fast Path OVS Acceleration* to enable high performance virtual switching between VMs and the outside world.

Features

- Packet Rx/Tx with configurable queue/ring to core mapping
- Jumbo frames
- Software counters and statistics
- Start, stop, and close operations
- Offload support: checksum, TSO
- Live migration
- Packet distribution using round-robin or flow hash computed from Ethernet protocol, vlan ID, IP addresses, L4 protocols and ports out of the most inner packet inside Ethernet, VLAN, IPv4, IPv6, IP tunnels, GRE.
- Dynamic multi-queue configuration, based on Qemu 2.5 API

Dependencies

6WINDGate modules

- *6WINDGate DPDK*

QEMU

- QEMU >= 2.1
- To support multiple queues with QEMU < 2.5, apply the following patch against QEMU:

  https://patchwork.ozlabs.org/patch/432404/
- To support migration with QEMU < 2.5, apply the following patch against QEMU:

  https://patchwork.ozlabs.org/patch/488672/
- To support *vhost-user* PMDs on guests with more than 2 GB memory, QEMU versions lower than 2.2 may need the patch below. For more information, contact your Linux distribution provider.

  http://git.qemu.org/?p=qemu.git;a=commitdiff;h=d3f16ec887bb58b19eac94fcae139a39a7933e15

libvirt

- `libvirt` >= 1.2.10
- To support live migration with `libvirt` <= 1.2.12, apply the following patch against `libvirt`:

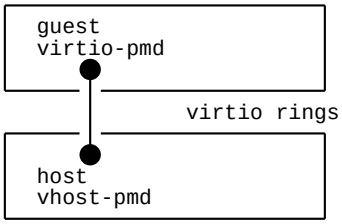
  http://libvirt.org/git/?p=libvirt.git;a=commitdiff;h=bd1a133fc8aa3ef847046094f01133708f455d20

Use Cases

PMD in the guest

To load Virtio Host PMD in a *6WINDGate DPDK* application, pass the *6WINDGate DPDK* add-on library to the *6WINDGate DPDK* application with the `-d` option. To instantiate a virtual device, use the `--vdev` EAL option (one per device).

- A *6WINDGate DPDK* application runs on one core on the host
- A *6WINDGate DPDK* application runs on two cores on the guest
- Packets are sent/received using 2 rings for each direction
- There is no notification (this is not needed with PMDs)



1. On the host, reserve huge pages and mount a `hugetlbfs` file system.

```
# echo 4096 > /sys/devices/system/node/node1/hugepages/hugepages-2048kB/nr_
↪hugepages
# mkdir -p /mnt/huge-2M && mount -t hugetlbfs none /mnt/huge-2M
```

2. Start the *6WINDGate DPDK* application on the host.

```
# QUEUE_CONFIG="rxqmap=manual:rr/1:3,txqmap=manual:rr/0:2"
# cd /path/to/dpdk/x86_64-native-linuxapp-gcc
# ./app/testpmd -c 0x3000 -n 3 --socket-mem=0,512 --huge-dir=/mnt/huge-2M \
      -d /path/to/librte_pmd_vhost.so \
      --vdev pmd-vhost0,${QUEUE_CONFIG},sockname=/tmp/vhost_sock0 \
      -- --socket-num=1 --port-numa-config=0,1 --port-topology=chained -i
```

We use `testpmd`, provided in the *6WINDGate DPDK* package. In this example, it uses 2 cores (mask is 0x3000) but only one is used for the data plane, the other is for the command line.

In this example, two RX queues are created, corresponding to the ring indexes 1 and 3, and two TX queues are created, corresponding to the ring indexes 0 and 2. The indexes of RX rings must be odd, and the indexes of TX rings must be even.

Note:

- The argument `sockname=/tmp/vhost_sock0` specifies the path of the `vhost-user` Unix socket to create. QEMU will connect to this socket to negociate features and configure the `vhost` PMD (Poll Mode Driver) driver.
- To optimize performance, it is a good idea to take care of where memory is allocated, using `numactl --cpunodebind=X --membind=X` when starting the *6WINDGate DPDK* application, specifying `--socket-mem=X`, and allocating huge pages on the proper socket. In this example, all is allocated on the second socket (socket 1).

---

3. Start QEMU, simulating a 4 cores machine, with a `vhost-user` virtio device providing two pairs of queues:

```
# numactl --cpunodebind=1 --membind=1 \
   qemu-system-x86_64 --enable-kvm -k fr -m 2G \
   -cpu host -smp cores=4,threads=1,sockets=1 \
   -serial telnet::4445,server,nowait -monitor telnet::5556,server,nowait \
```

(continues on next page)

(continued from previous page)

```
  -hda vm.qcow2 \
  -object memory-backend-file,id=mem,size=2G,mem-path=/mnt/huge-2M,share=on \
  -numa node,memdev=mem \
  -chardev socket,path=/tmp/vhost_sock0,id=chr0,server \
  -netdev type=vhost-user,id=net0,chardev=chr0,queues=2,vhostforce=on \
  -device virtio-net-pci,netdev=net0,vectors=5,mq=on,ioeventfd=on
```

Note:

- To manage more than one pair of queues, you must patch QEMU.
- For maximum performance, every pthread of the QEMU process (except management pthreads) has to be pinned to a CPU and must be the only application running on that CPU. This can be done with several `tasksets`. The pthread IDs associated with vCPU can be retrieved from the QEMU console using the `info cpu` command.

---

4. In the guest, the virtual `virtio` PCI device should be listed:

```
# lspci -nn
[...]
00:03.0 Ethernet controller [0200]: Red Hat, Inc Virtio network device [1af4:1000]
```

5. On the guest, reserve the huge pages and start the *6WINDGate DPDK* application:

```
# modprobe uio
# cd /path/to/dpdk/x86_64-native-linuxapp-gcc
# insmod kmod/igb_uio.ko
# python ../tools/dpdk_nic_bind.py -b igb_uio 0000:00:03.0
# echo 512 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB/nr_
↪hugepages
# mkdir -p /mnt/huge-2M && mount -t hugetlbfs none /mnt/huge-2M
# ./app/testpmd -c 0x7 -n 3 --huge-dir=/mnt/huge-2M \
    --pci-whitelist 0000:00:03.0 \
    -- --rxd=256 --txd=256 --txqflags=0xf00 -i --port-topology=chained \
    --rxq=2 --txq=2
```

6. Transmit packets between host and guest. For instance:

   a. On the guest, enter:

```
testpmd> set fwd rxonly
testpmd> set corelist 1,2
testpmd> start
```

   b. On the host, enter:

```
testpmd> set fwd txonly
testpmd> start
[wait 10 seconds]
testpmd> stop
```

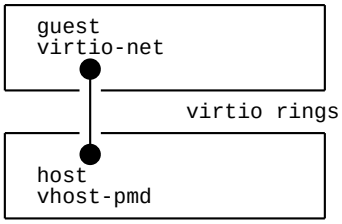
See also:

- *PMD configuration*
- *QEMU configuration*
- The *6WINDGate DPDK* documentation

Linux kernel `netdevice` in the guest

This is another example where the packets are transmitted between a host running Virtio Host PMD and a guest running a Linux with a `virtio-net` kernel module.

- A *6WINDGate DPDK* application runs on one core on the host
- The guest runs a `virtio-net` kernel module, on 4 cores
- Packets are sent/received using 4 rings from host to guest and 4 rings from guest to host
- Notification when the host sends packets to the guests



1. On the host, reserve huge pages and mount a `hugetlbfs` file system.

```
# echo 4096 > /sys/devices/system/node/node1/hugepages/hugepages-2048kB/nr_
↪hugepages
# mkdir -p /mnt/huge-2M && mount -t hugetlbfs none /mnt/huge-2M
```

2. Start the *6WINDGate DPDK* application on the host.

```
# QUEUE_CONFIG="rxqmap=manual:rr/1:3:5:7,txqmap=manual:rr/0:2:4:6"
# cd /path/to/dpdk/x86_64-native-linuxapp-gcc
# ./app/testpmd -c 0x3000 -n 3 --socket-mem=0,512 --huge-dir=/mnt/huge-2M \
      -d /path/to/librte_pmd_vhost.so \
      --vdev pmd-vhost0,${QUEUE_CONFIG},sockname=/tmp/vhost_sock0 \
      -- --socket-num=1 --port-numa-config=0,1 --port-topology=chained -i
```

We use `testpmd`, provided in the *6WINDGate DPDK* package. In this example, it uses 2 cores (mask is 0x3000) but only one is used for the data plane, the other is for the command line.

Note: The argument `sockname=/tmp/vhost_sock0` specifies the path of the `vhost-user` Unix socket to create. QEMU will connect to this socket to negociate features and configure Virtio Host PMD.

---

3. On the host, configure `testpmd` to answer to all ICMP echo requests:

```
testpmd> set fwd icmpecho
testpmd> start
```

4. Start QEMU, simulating a 4 cores machine with a `vhost-user` virtio device providing 4 pairs of queues:

```
# numactl --cpunodebind=1 --membind=1 \
   qemu-system-x86_64 --enable-kvm -k fr -m 2G \
   -cpu host -smp cores=4,threads=1,sockets=1 \
   -serial telnet::4445,server,nowait -monitor telnet::5556,server,nowait \
   -hda vm.qcow2 \
   -object memory-backend-file,id=mem,size=2G,mem-path=/mnt/huge-2M,share=on \
   -numa node,memdev=mem \
   -chardev socket,path=/tmp/vhost_sock0,id=chr0,server \
   -netdev type=vhost-user,id=net0,chardev=chr0,queues=4 \
   -device virtio-net-pci,netdev=net0,vectors=9,mq=on,ioeventfd=on
```

Note:

- To manage more than one pair of queues, you must patch QEMU.
- For maximum performance, every pthread of the QEMU process (except management pthreads) has to be pinned to a CPU and must be the only application running on that CPU. This can be done with several `tasksets`. The pthread IDs associated with vCPU can be retrieved from the QEMU console using the `info cpu` command.
- QEMU must support `vhost-user`, which is the case starting from QEMU 2.1.

---

5. In the guest, the virtual `virtio` PCI device should be listed:

```
# lspci -nn
[...]
00:03.0 Ethernet controller [0200]: Red Hat, Inc Virtio network device [1af4:1000]
```

6. The `virtio-net` kernel module should be automatically loaded. Configure the network in this interface with 4 queues and an IP address:

```
# ETH=eth0
# ls /sys/class/net/${ETH}/queues
# ethtool -L ${ETH} combined 4
# ethtool -l ${ETH}
# ip a a 1.1.1.1/24 dev ${ETH}
# ip l set ${ETH} up
# arp -s 1.1.1.2 00:00:00:00:00:01
# ping 1.1.1.1
```

7. Configure the affinity of the MSI-x interrupts to increase performance, spreading `virtio0-input.X` on all cores:

```
# service irqbalance stop
# grep virtio0-input /proc/interrupts
# echo 01 > /proc/irq/41/smp_affinity
# echo 02 > /proc/irq/43/smp_affinity
# echo 04 > /proc/irq/45/smp_affinity
# echo 08 > /proc/irq/47/smp_affinity
```

   In this example, the packets sent by the host will be distributed over the 4 Linux cores in round-robin mode.

Note: To spread TX (Transmission) packets over multiple queues in the guest, use the Linux XPS feature (available in Linux version 2.6.38 and higher). You must enable the Linux kernel configuration item `CONFIG_XPS`. This item allows selecting a transmission queue from the CPU that transmits a packet. Without XPS, you can use only one queue when transmitting over a `virtio` interface.

To configure XPS, specify the CPU/queue association *via* the `sysfs` interface:

```
# echo ${COREMASK} > /sys/class/net/${DEV}/queues/tx-${QNUM}/xps_cpus
```

In the example above, to associate `cpu0` to `queue0`, `cpu1` with `queue1`, and so on, enter:

```
# echo 1 > /sys/class/net/eth0/queues/tx-0/xps_cpus
# echo 2 > /sys/class/net/eth0/queues/tx-1/xps_cpus
# echo 4 > /sys/class/net/eth0/queues/tx-2/xps_cpus
# echo 8 > /sys/class/net/eth0/queues/tx-3/xps_cpus
```

See also:

- *PMD configuration*
- *QEMU configuration*
- The *6WINDGate DPDK* documentation

Usage

PMD configuration

Virtio Host PMD supports several parameters on the command line. The arguments are comma-separated after the name of the device on the `--vdev` DPDK application command line. For instance: `--vdev pmd-vhost0, verbose=1,sockname=/tmp/sock`.

Parameters

sockname=[path]
: Set the path of the UNIX socket. This socket is used to negociate features and configure Virtio Host PMD using the `vhost-user` protocol.

sockmode=[server|client]
: Configure the socket in server or client (default) mode. In server mode, the Virtio Host PMD listens on the socket and waits that QEMU to connect on it.

  In client mode, the Virtio Host PMD connects directly to the socket. Thus QEMU needs to be started before the Virtio Host PMD is configured. And the `vhost-user` netdevice created by QEMU needs to be configured in server mode.

sockuser=[username|uid]
: Set the owner of the UNIX socket file to sockname. This option can only be used if sockmode option is set to server.

sockgroup=[groupname|gid]
: Set the group of the UNIX socket file to sockgroup. This option can only be used if sockmode option is set to server.

sockperm=[permission]
: Set the permission of the UNIX socket file to sockperm. This option can only be used if sockmode option is set to server.

verbose=[0|1]
: Display the debug messages of the Virtio Host PMD. The dpdk log-level needs also be set to debug.

macaddr=[macaddr]
: Set the MAC address of the host interface.

rxqmap
: Configure the list of rings to be polled by receive queues. See below for details.

txqmap
: Configure the list of rings to transmit on for transmit queues. See below for details.

profile
: Define the usage profile. Valid values are *endpoint* or *nfv*. In *nfv* mode, the Virtio Host PMD is optimized for the NFV use-case:

- the mergeable buffer feature is disabled, allowing the guest to use a faster RX (Reception) function that does not support large receive offload.
- the number of vring referenced by a TX `qmap` is lower to decrease lock contention. See the `qmap` configuration section for details.

`txhash`
Select which packet layers are used to calculate the Tx hash. It can be `l3l4` (default) or `l3`.

Qmaps

About qmaps

You can associate *6WINDGate DPDK* queues with virtio rings via `qmap` structures.

A virtio ring is a software structure used to exchange packets between a host and a guest.

A *6WINDGate DPDK* queue is a software abstraction layer that is part of the *6WINDGate DPDK* PMD API: packets are received from and transmitted to *6WINDGate DPDK* queues. A queue is accessed by one core at a time. For physical hardware, a software queue is usually associated with an hardware queue. In Virtio Host PMD, the software queue is associated with one or several virtio rings for more flexibility. For example, it allows to spread the traffic sent to one queue over several virtio rings, providing a RSS-like feature to the guest.

You can set up more complex associations with `qmap` configurations:

1-1 association One queue is associated with one ring.

1-N association One queue is associated with several rings, selected via a round-robin or a hash mechanism.

N-1 association N queues are associated with the same ring.

N-M association N queues are associated with several rings, selected via a round-robin or a hash mechanism.

A `qmap` structure is associated with one *6WINDGate DPDK* queue; therefore it is only accessed by one core.

A `rxqmap` is a list of rings polled by one *6WINDGate DPDK* queue in a round-robin fashion.

A `txqmap` is a list of rings filled when transmitting on one *6WINDGate DPDK* queue. The rings are selected using round-robin or based on the flow hash of the packet. In the latter case, the flow hash depends on:

- the Ethernet protocol
- the most inner vlan id
- the most inner IPv4 or IPv6 addresses
- the Layer 4 (Transport Layer) protocol
- the Layer 4 ports if protocol is TCP or UDP

The following network headers are recognized:

- Ethernet

- VLAN
- IPv4, IPv6
- IPv4 and IPv6 tunnels
- GRE tunnels
- TCP, UDP

It is important to understand how queues are allocated in your application (for instance `testpmd`, or the fast path). In the fast path, there is one TX queue per core, and you can configure the number of RX queues via the `CORE_PORT_MAPPING` parameter.

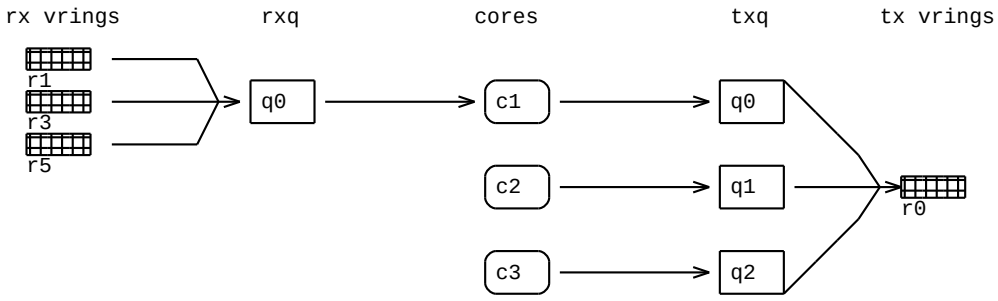
The virtio specifications states that:

- RX rings identifiers are odd
- TX rings identifiers are even

Example

The example below illustrates the concept of `qmap`. It is based on the following configuration:

- One port and 3 cores
- One RX queue (`q0`), polled by core 1
- 3 RX virtio rings, polled in round-robin mode when reading from RX queue 0
- 3 TX queues (one for each core)
- One TX ring, filled when transmitting to any TX queue



When configuring Virtio Host PMD, you can specify:

- which rings are polled for each `rxqmap`,
- to which rings packets will be transmitted for each `txqmap`.

To avoid packet losses, check the `qmap` configuration to ensure that:

- the virtio rings filled by the host are all emptied by the guest,
- the virtio rings filled by the guest are all emptied by the host.

To enhance performance, if possible, set up the fast path `CORE_PORT_MAPPING` value and the driver `qmap` configuration so that:

- Work is equally distributed among all cores.
- Inter-sockets memory transfer is kept to a minimum: cores polling a given virtio ring should be on the same socket as cores transmitting to the virtio ring.
- Concurrent accesses on virtio rings are kept to a minimum: virtio rings are single-producer and single-consumer, therefore a lock is automatically added when several cores can access the same virtio ring.
- The number of virtio rings is kept to a minimum: when virtio rings are full, in-flight packets can consume much memory. If memory consumption exceeds the size of the CPU cache, performance drops.

See also:

- Virtual I/O Device (VIRTIO) Specifications
- The *6WINDGate DPDK* documentation
- *FPN-SDK Add-on for DPDK* for details about the fast path queue allocation policy

Default configuration

The default configuration (no `rxqmap` or `txqmap` argument passed to the device) depends on the QEMU version.

With QEMU >= 2.5, the association of *6WINDGate DPDK* queues with virtio rings use the *Dynamic auto mode*: the qmap configuration is automatically adapted to the number of virtio rings requested by the guest.

With QEMU < 2.5, the default configuration uses the *Static auto mode*, with round-robin and one ring for rxqmap and txqmap. This configuration works only when QEMU is configured with 1 virtio ring, as the fast path sends and receives data only on the first virtio ring. If QEMU is configured to have several virtio rings this default configuration must not be used (otherwise packets sent by the guest may be lost).

Dynamic auto mode

This mode always ensures that:

- the virtio rings filled by the host are all emptied by the guest,
- the virtio rings filled by the guest are all emptied by the host.

To fullfill this requirement the applied configuration always matches the following rules:

- For TX: In default mode, *6WINDGate DPDK* queues are connected to all virtio rings, and packets are spread depending on the flow hash. In *nfv* mode, *6WINDGate DPDK* queues are connected to some of the virtio rings, and packets are spread depending on the flow hash.
- For RX: A newly activated virtio ring is added to the least used *6WINDGate DPDK* queue.

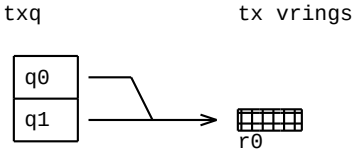
If the number of activated virtio rings is changed by the guest (with the command `ethtool -L ${ETH} combined X`), the qmap configuration is modified dynamically.

Example for QEMU with 3 vring pairs (called queue pairs by Qemu), 6WINDGate DPDK with 2 rx-queues and 2 tx-queues

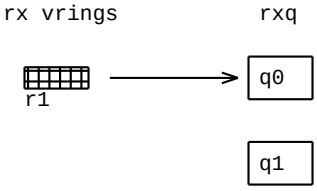
1 activated queue on the guest

```
# ethtool -L eth0 combined 1
```

TX mapping, using *default* or *nfv* profile:



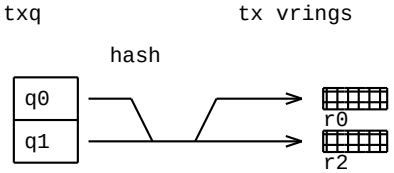
RX mapping:



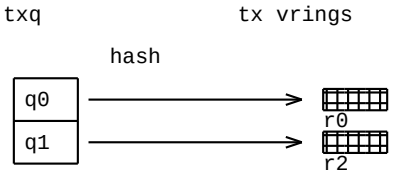
2 activated queues on the guest

```
# ethtool -L eth0 combined 2
```

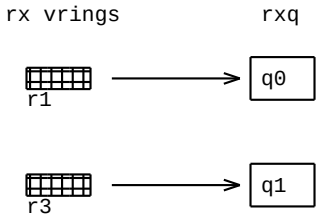
TX mapping, using *default* profile:



TX mapping, using *nfv* profile:



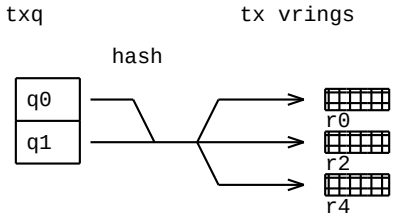
RX mapping:



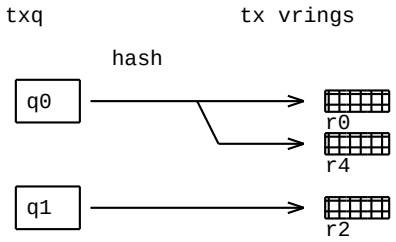
3 activated queues on the guest

```
# ethtool -L eth0 combined 3
```

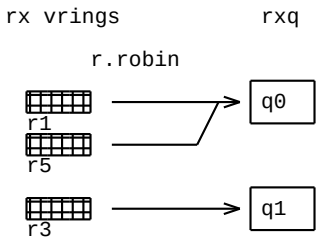
TX mapping, using *default* profile:



TX mapping, using *nfv* profile:



RX mapping:



Static auto mode

In this mode, the numbers of rings and queues are specified and rings are automatically associated to queues. This configuration is static, meaning that the port has to be restarted to modify it.

The option syntax is as follows:

```
auto:<sched>/nb_ring:<nb_ring>[/nb_qmap:<nb_qmap>]
```

sched Mandatory. Select round robin (`rr`) or transmit hash (`hash`).

nb_ring Mandatory. Number of rings to be referenced by `qmaps`.

nb_qmap Optional. Number of queues to be created by the Virtio Host PMD. If it is not specified, the number of queues will be set by the application at Virtio Host PMD initialization. For example, when using the fast path, one TX queue per core is created, and the number of RX queues created depends on the `CORE_PORT_MAPPING` parameter.

The auto mode ensures that all the rings specified by `nb_rings` (`0, 2, 4, ..., 2*(nb_rings-1)` for TX, and `1, 3, 5, ..., 2*(nb_rings-1)+1` for RX) will be polled or filled if all `qmaps` (queues) are used by the application.

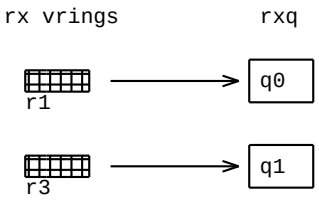
In round-robin mode, if the number of rings is the same as the number of queues, a 1:1 association is applied. For other use cases, see the examples below.

To apply a non-uniform mapping, which can be useful in some specific situations, use the manual mode.

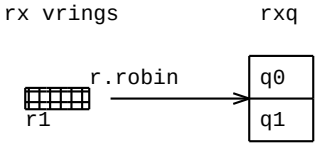
For `rxqmap`, the `sched` argument can only be `rr` (round-robin). For `txqmap`, the `sched` argument can be `rr` (round-robin) or `hash`. In the latter case, each `qmap` points to all virtual rings, and the virtual ring used to transmit packets depends on the flow hash of the packet.

Examples for rxqmap in auto mode

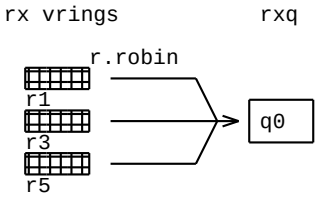
auto:rr/nb_ring:2/nb_qmap:2 The core polling queue 0 polls ring 1, the core polling queue 1 polls ring 3.



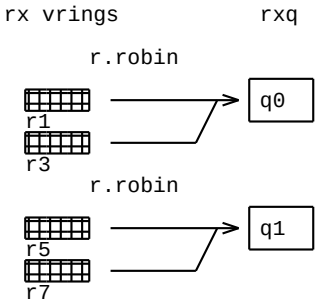
auto:rr/nb_ring:1/nb_qmap:2 The core polling queue 0 polls ring 1, the core polling queue 1 polls ring 1.



auto:rr/nb_ring:3/nb_qmap:1 The core polling queue 0 polls rings 1, 3, and 5.

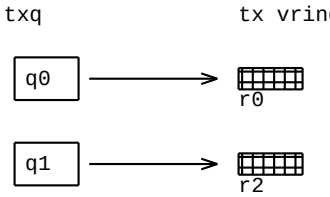


auto:rr/nb_ring:4/nb_qmap:2 The core polling queue 0 polls rings 1, 3 and the core polling queue 1 polls rings 5, and 7.

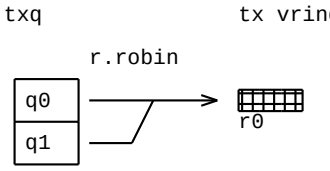


Examples for txqmap in auto mode

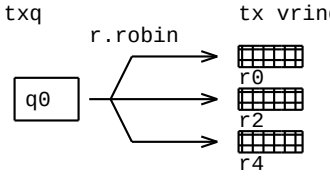
auto:rr/nb_ring:2/nb_qmap:2 The core transmitting on queue 0 transmits on ring 0, and the core transmitting on queue 1 transmits on ring 2.



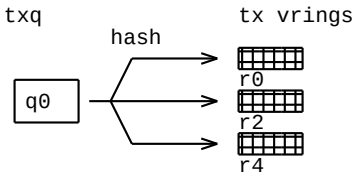
auto:rr/nb_ring:1/nb_qmap:2 The core transmitting on queue 0 transmits on ring 0, and the core transmitting on queue 1 transmits on ring 0.



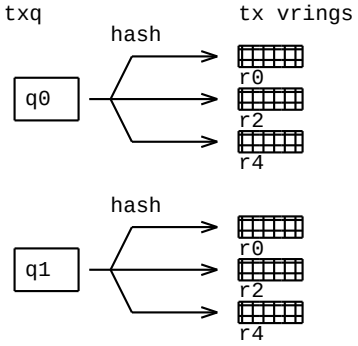
auto:rr/nb_ring:3/nb_qmap:1 The core transmitting on queue 0 transmits on rings 0, 2, or 4, in round-robin mode.



auto:hash/nb_ring:3/nb_qmap:1 The core transmitting on queue 0 transmits on ring 0, 2, or 4, depending on the packet flow.

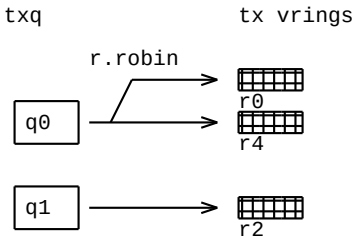


auto:hash/nb_ring:3/nb_qmap:2 When hash is used, each queue references all virtual rings.

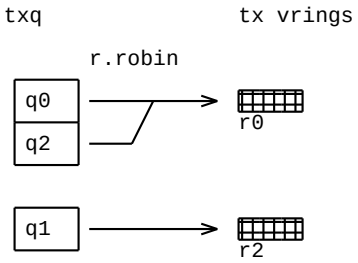


auto:rr/nb_ring:3/nb_qmap:2 If nb_ring and nb_qmap are relatively prime, the mapping will not be uniform. In this case, the core transmitting on queue 0 transmits on rings 0 and 4, and the core transmitting on queue 1 transmits on ring 2.

In this case, to keep a better control, use the manual mode.



auto:rr/nb_ring:2/nb_qmap:3 In this case as well, use the manual mode to avoid a non uniform mapping.



Static manual mode

In manual mode, you can specify the list of rings associated with each queue. This configuration is static, meaning that the port has to be restarted to modify it.

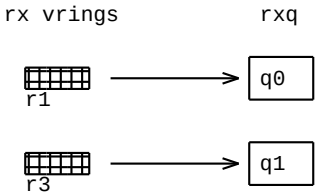
The option syntax is as follows:

```
manual:<sched>/<ring_idx0>:<ring_idx1>.../<ring_idx2>:<ring_idx3>.../...

                \                 /       \                 /
                      qmap 0                    qmap 1          ...
```

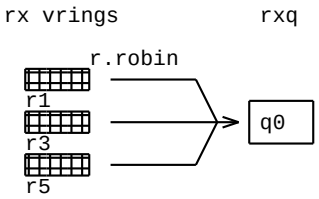
For `rxqmap`, the `sched` argument can only be `rr` (round-robin). For `txqmap`, the `sched` argument can be `rr` (round-robin) or `hash`. In the latter case, the virtual ring used to transmit packets depends on the flow hash of the packet.

Examples for rxqmap in manual mode

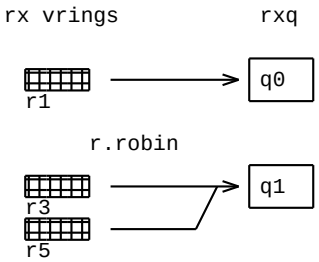
`manual:rr/1/3` The core polling queue 0 polls ring 1, the core polling queue 1 polls ring 3.



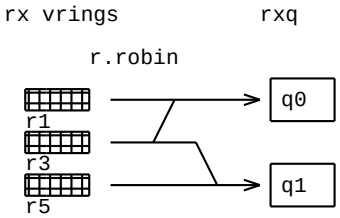
`manual:rr/1:3:5` The core polling queue 0 polls rings 1, 3, and 5.



`manual:rr/1/3:5` The core polling queue 0 polls ring 1, and the core polling queue 1 polls rings 3 and 5.

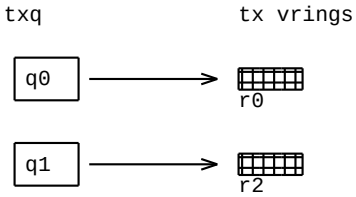


`manual:rr/1:3/3:5` The core polling queue 0 polls rings 1 and 3, and the core polling queue 1 polls rings 3 and 5.

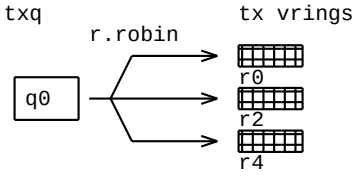


Examples for txqmap in manual mode

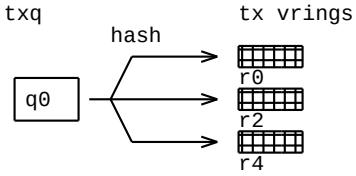
manual:rr/0/2 The core transmitting on queue 0 transmits on ring 0, and the core transmitting on queue 1 transmits on ring 2.



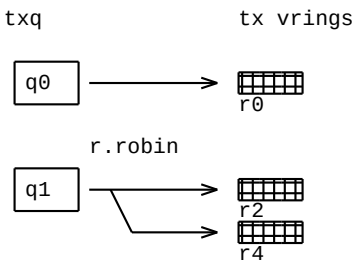
manual:rr/0:2:4 The core transmitting on queue 0 transmits on rings 0, 2, or 4 in round-robin mode.



manual:hash/0:2:4 The core transmitting on queue 0 transmits on ring 0, 2 or 4 depending on the packet flow hash.



manual:rr/0/2:4 The core transmitting on queue 0 transmits on rings 0, and the core transmitting on queue 1 transmits on rings 2 and 4 in round-robin mode.



QEMU configuration

Important:

- A bug in some versions of QEMU prevents mergeable buffers to be correctly negotiated when using a `vhost-user` netdevice. The following patch, backported in QEMU v2.1.3, fixes the bug in QEMU v2.2.0: http://git.qemu.org/?p=qemu.git;a=commitdiff;h=d8e80ae37a7acfea416ad9abbe76b453a73d9cc0
- A bug in some versions of QEMU prevents the `vhost` user interface to properly negotiate supported features, causing the device to fail initialization. The following patch, backported in QEMU v2.1.1, fixes the bug in QEMU v2.2.0: http://git.qemu.org/?p=qemu.git;a=commitdiff;h=b49ae9138d5cadb47fb868297fbcdac8292fb666
- Start the VM with memory allocated from hugepages in shared mode with the following configuration (the `vhost-user` PMD must be able to map the VM's memory to write in the `virtio` ring):

```
-object memory-backend-file,id=mem,size=2G,mem-path=/mnt/huge-2M,share=on \
-numa node,memdev=mem
```

---

1. Start a `vhost-user` netdevice with QEMU:

```
-chardev socket,id=vhostusernet0,path=/tmp/vhost_sock0[,server] \
-netdev type=vhost-user,id=vhostusernet0,chardev=vhostuserchr0,queues=8 \
-device virtio-net-pci,netdev=vhostusernet0,ioeventfd=on,vectors=17,mq=on
```

   `path` Path to the `vhost-user` socket.

   `server` Create the `vhost-user` socket in server mode. This option needs to be set if the Virtio Host PMD socket is configured in client mode.

   `queues=x` Number of queue pairs seen by the driver on the guest. This requires a patch in QEMU, provided in `librte_pmd_vhost/patches/qemu-2.x`.

   `vectors=x` Required for multiqueue; should be `NQ * 2 + 1`.

   `mq=on` Enables multiqueue.

   `ioeventfd=on` Should increase notification performance when the guest driver is a Linux driver

2. **[Optional]** Modify the set of negotiated features.

   By default, QEMU automatically enables all options supported by both sides (*checksum offloading, GSO, mergeable buffers,* and so on). To modify the set of negotiated features, you can edit the following parameters:

   `guest_csum=on/off` Offload l4 checksum from host to guest (default: on)

   `guest_tso4=on/off` Forward TSO4 packets from host to guest (default: on)

   `guest_tso6=on/off` Forward TSO6 packets from host to guest (default: on)

   `csum=on/off` Offload l4 checksum from guest to host (default: on)

host_tso4=on/off Forward TSO4 packets from guest to host (default: on)

host_tso6=on/off Forward TSO6 packets from guest to host (default: on)

mrg_rxbuf=on/off Support mergeable buffers in host (default: on)

3. Display negotiated features *via* the QEMU monitor:

```
(qemu) info qtree
bus: main-system-bus
  type System
  dev: hpet, id ""
    gpio-in "" 2
    gpio-out "" 1
    timers = 3 (0x3)
    msi = false
    hpet-intcap = 4 (0x4)
    irq 32
    mmio 00000000fed00000/0000000000000400
  dev: kvm-ioapic, id ""
    gpio-in "" 24
    gsi_base = 0 (0x0)
    irq 0
    mmio 00000000fec00000/0000000000001000
  dev: i440FX-pcihost, id ""
    pci-hole64-size = 18446744073709551615 (16 EiB)
    short_root_bus = 0 (0x0)
    irq 0
    bus: pci.0
      type PCI
      dev: virtio-net-pci, id ""
        ioeventfd = true
        vectors = 3 (0x3)
        any_layout = false
        csum = true
        guest_csum = true
        gso = false
        guest_tso4 = true
        guest_tso6 = true
        guest_ecn = false
        guest_ufo = false
        guest_announce = true
        host_tso4 = true
        host_tso6 = true
        host_ecn = false
        host_ufo = false
        mrg_rxbuf = true
```

(continues on next page)

(continued from previous page)

```
        status = true
        ctrl_vq = true
        ctrl_rx = true
        ctrl_vlan = false
        ctrl_rx_extra = false
        ctrl_mac_addr = true
        ctrl_guest_offloads = false
        mq = false
        mac = "02:11:22:00:06:00"
        vlan = <null>
        netdev = "vhostusernet0"
        bootindex = -1 (0xffffffffffffffff)
        x-txtimer = 150000 (0x249f0)
        x-txburst = 256 (0x100)
        tx = ""
        indirect_desc = false
        event_idx = false
        addr = 04.0
        romfile = "efi-virtio.rom"
        rombar = 1 (0x1)
        multifunction = false
        command_serr_enable = true
        class Ethernet controller, addr 00:04.0, pci id 1af4:1000 (sub 1af4:0001)
        bar 0: i/o at 0xc100 [0xc11f]
        bar 1: mem at 0xfebd1000 [0xfebd1fff]
        bar 6: mem at 0xffffffffffffffff [0x3fffe]
        bus: virtio-bus
          type virtio-pci-bus
          dev: virtio-net-device, id ""
            mac = "02:11:22:00:06:00"
            vlan = <null>
            netdev = "vhostusernet0"
            bootindex = -1 (0xffffffffffffffff)
            x-txtimer = 150000 (0x249f0)
            x-txburst = 256 (0x100)
            tx = ""
      dev: ne2k_pci, id ""
        mac = "de:ad:de:01:02:03"
        vlan = <null>
        netdev = "user.0"
        bootindex = -1 (0xffffffffffffffff)
        addr = 03.0
        romfile = "efi-ne2k_pci.rom"
        rombar = 1 (0x1)
```

(continues on next page)

(continued from previous page)

```
        multifunction = false
        command_serr_enable = true
        class Ethernet controller, addr 00:03.0, pci id 10ec:8029 (sub 1af4:1100)
        bar 0: i/o at 0xc000 [0xc0ff]
        bar 6: mem at 0xffffffffffffffff [0x3fffe]
        ...
```

`libvirt` configuration

Start a `vhost-user` netdevice with `libvirt` with the following configuration:

```
<interface type='vhostuser'>
  <mac address='52:54:00:ee:96:6d'/>
  <source type='unix' path='/tmp/vhost_sock0' mode='server'/>
  <model type='virtio'/>
  <driver queues='8'/>
</interface>
```

path Path to the `vhost-user` socket.

queues=x Number of queue pairs seen by the driver on the guest. This requires a patch in `libvirt`, provided in `librte_pmd_vhost/patches/libvirt-v1.12.12`.

Important: Start the VM with memory allocated from hugepages in shared mode with the following configuration (the `vhost-user` PMD must be able to map the VM's memory to write in the virtio ring):

```
<cpu>
  <numa>
    <cell id="0" cpus="0" memory="2097152" memAccess="shared"/>
  </numa>
</cpu>
<memoryBacking>
  <hugepages>
    <page size="2048" unit="KiB" nodeset="0"/>
  </hugepages>
</memoryBacking>
```

Live migration

If the host capabilities used by the VM on host 1 are also present on host 2, you can perform a live migration from host 1 to host 2.

Choose carefully all XML tags related to the host architecture (<vcpu>, <cputune>, <cpu>, and so on).

For <cpu> XML tags, use the mode="host-model" option only if all hosts are strictly identical.

Note:

To get common capabilities between hosts, use the virsh capabilities or cpu-baseline commands.

Example

We will get common capabilities between a Sandybridge host and a Westmere host.

1. On the first host (Sandybridge), get the CPU capabilities:

```
$ virsh capabilities > host1.xml
```

2. On the second host (Westmere), get the common capabilities between the two hosts to set in the <cpu> XML tag:

```
$ virsh cpu-baseline host1.xml
<cpu mode='custom' match='exact'>
  <model fallback='forbid'>Westmere</model>
  <vendor>Intel</vendor>
  <feature policy='require' name='invtsc'/>
  <feature policy='require' name='rdtscp'/>
  <feature policy='require' name='pdpe1gb'/>
  <feature policy='require' name='dca'/>
  <feature policy='require' name='pcid'/>
  <feature policy='require' name='pdcm'/>
  <feature policy='require' name='xtpr'/>
  <feature policy='require' name='tm2'/>
  <feature policy='require' name='est'/>
  <feature policy='require' name='smx'/>
  <feature policy='require' name='vmx'/>
  <feature policy='require' name='ds_cpl'/>
  <feature policy='require' name='monitor'/>
  <feature policy='require' name='dtes64'/>
  <feature policy='require' name='pclmuldq'/>
  <feature policy='require' name='pbe'/>
  <feature policy='require' name='tm'/>
```

(continues on next page)

(continued from previous page)

```
  <feature policy='require' name='ht'/>
  <feature policy='require' name='ss'/>
  <feature policy='require' name='acpi'/>
  <feature policy='require' name='ds'/>
  <feature policy='require' name='vme'/>
</cpu>
```

Migration may fail due to restrictions on which flags can be migrated. In the example above, we must remove the `invtsc` feature, since it cannot be migrated.

The following patch solves this issue, but it is not integrated in `libvirt` yet:

http://www.redhat.com/archives/libvir-list/2015-March/msg01521.html

---

You can migrate a VM from host 1 to host 2, *via* `virsh migrate`:

```
$ virsh migrate $domain_name --live \
        qemu+ssh://$name_or_address_of_host2/system \
        tcp://$name_or_address_of_host2
```

2.2 FPN-SDK

2.2.1 FPN-SDK Baseline

Overview

FPN-SDK Baseline is the foundation of the FPN-SDK hardware abstraction layer.

Features

- Portability across multiple processors SDKs: DPDK (Intel, ARMv8), Cavium SDK
- Packet MBUF API
- Crypto API abstraction to leverage crypto processors
- Fast and scalable timer API
- Job API for flexible per core function assignment
- Memory pool and ring API
- Lock and synchronization API
- Atomic operations API

- Shared memory API (userland / kernel / fast path)
- FPVI Linux driver
- CPU usage monitoring
- Function calls tracking for debugging
- Intercore API to distribute packet processing over cores
- Checksum computation
- Software TCP Large Receive Offload
- Hardware TCP Segmentation Offload
- Control Plane Protection
- Hardware flow

Dependencies

6WINDGate modules

- FPN-SDK add-on for your architecture: DPDK (Intel, ARMv8), Cavium.

Usage

Tools

fp-cpu-usage

Description

Display the number of percents of cpu usage spent to process packets.

Packets can come from NIC or from intercore (mainly due to offload of cryptographic operations). Average cycles/packet for these two kinds of packets is provided.

Synopsis

```
# fp-cpu-usage [-q|--quiet] [-d|--delay] [-j|--json] [-h|--help]
```

Parameters

-q, --quiet
Display fast path logical cores usage in quiet mode.

-d, --delay
Duration of CPUs usage polling in microsecond.

-j, --json
Display informations in JSON format.

-h, --help
Display help.

Example

```
# fp-cpu-usage
Fast path CPU usage:
cpu: %busy     cycles   cycles/packet  cycles/ic pkt
  2:   99%  697179716             829             0
  4:   51%  363169408               0          1729
  6:   54%  383451844               0          1825
 16:   51%  362776960               0          1727
 18:   54%  382313120               0          1820
average cycles/packets received from NIC: 2626 (2206989016/840180)
```

```
# fp-cpu-usage -q -d 100000
Fast path CPU usage (quiet):
cpu: status
 19: alive
 20: alive
 27: alive
 34: alive

 100% CPUs alive
```

```
# fp-cpu-usage -j
{
  "average_cycles_per_packet": 2626,
  "total_cycles": 2206989016,
  "cpus": [
    {
      "busy": 99,
      "cpu": 2,
      "cycles": 697179716,
```

(continues on next page)

(continued from previous page)


```
      "cycles_per_packet": 829,
      "cycles_per_ic_pkt": 0
    },
    {
      "busy": 51,
      "cpu": 4,
      "cycles": 363169408,
      "cycles_per_packet": 0,
      "cycles_per_ic_pkt": 1729
    },
    {
      "busy": 54,
      "cpu": 6,
      "cycles": 383451844,
      "cycles_per_packet": 0,
      "cycles_per_ic_pkt": 1825
    },
    {
      "busy": 51,
      "cpu": 16,
      "cycles": 362776960,
      "cycles_per_packet": 0,
      "cycles_per_ic_pkt": 1727
    },
    {
      "busy": 54,
      "cpu": 18,
      "cycles": 382313120,
      "cycles_per_packet": 0,
      "cycles_per_ic_pkt": 1820
    }
  ],
  "total_packets": 840180
}
```


```
# fp-cpu-usage -h
Fastpath CPUs usage:
fp-cpu-usage [-q|--quiet] [-d|--delay] [-h|--help]
  -q, --quiet               Display fastpath CPUs usage in quiet mode.
  -d, --delay               Duration of CPUs usage dump in microsecond
                            (default 200000us)
  -j, --json                Display fast path logical cores usage in json format.
  -h, --help                Display this help.
```

fp-shmem-ports

Description

Display and configure the parameters of detected ports at FPN-SDK level.

Synopsis

```
# fp-shmem-ports <action> <options>
```

Parameters

-d, --dump

Display FPN-SDK port information. The dump contains the following information:

- The core frequency
- The TX offload feature status
- The list of UDP ports considered as vxlan ports by reassembly features
- One block per managed port, displaying the following information:

```
port <port_number>: <port_name> numa <port_numa> bus_info <bus> mac <port_mac>
↪ driver <pmd_driver> GRO <timeout>us
  speed <speed> duplex half|full autoneg on|off rx_pause on|off tx_pause␣
↪on|off autoneg_pause on|off
  <qdir> queues: <n> (max: <m>)
  <feature> on|off
```

  - **<port_number>** Port number
  - **<port_name>** Port name.
  - **<port_numa>** Numa of the port (set to 'no numa' for architecture with no numa or numa independent pci bus).
  - **<bus>** The bus information for this port (typically, pci address).
  - **<port_mac>** Port's MAC address.
  - **<pmd_driver>** Driver thats manages the port in the fast path.
  - **<timeout>** GRO timeout in us.
  - **<speed>** The link speed in Mb/s.
  - **<qdir>** RX or TX.
  - **<n>, <m>** Number and maximum number of RX or TX queues.

- **<feature>** Supported offload feature in:
  - RX vlan strip
  - RX IPv4 checksum
  - RX TCP checksum
  - RX UDP checksum
  - GRO
  - LRO
  - TX vlan insert
  - TX IPv4 checksum
  - TX TCP checksum
  - TX UDP checksum
  - TX SCTP checksum
  - TSO

-g <timeout>, **--gro-timeout**=<timeout>
Set software GRO timeout. timeout is the maximum lapse of time between two coalesced packets. In TCP reassembly, ack only packet timeout is not reloaded to timeout each time an ack is received. This timeout designates instead the maximum lapse of time during which ack only packets are coalesced. A timeout of 10 microseconds gives good reassembly results on 10 Gb links. To be effective, this option must be combined with -K gro on.

-e <eth_port>|all|enabled|disabled, **--eth_port**=<eth_port>|all|enabled|disabled
Select a given FPN-SDK port. all means all ports, enabled means all enabled ports, and disabled means all disabled ports.

--driver <driver_name>
Select FPN-SDK port using a specific driver.

-K, **--features**, **--offload** <feature> on|off

Set or unset offload feature. Supported features:

- rx: rx checksum offloads
- tx: tx checksum offloads
- tso: TCP segmentation offload
- gro: Generic receive offload
- lro: TCP large receive offload
- mpls-ip: GRO reassembly of MPLS IP flows that do not follow RFC3032.

-k, --show-features, --show-offload
Display offload features status

Examples

- Display FPN-SDK port information:

```
# fp-shmem-ports --dump
core freq : 2693482113
offload : enabled
vxlan ports :
   port 4789 (set by user)
   port 8472 (set by user)
port 0: ens1f0-vrf0 numa 0 bus_info 0000:00:03.0 mac 90:e2:00:12:34:56 driver rte_
↪ixgbe_pmd GRO timeout 10us
   RX queues: 2 (max: 128)
   TX queues: 2 (max: 64)
   RX vlan strip off
   RX IPv4 checksum on
   RX TCP checksum on
   RX UDP checksum on
   GRO on
   LRO off
   TX vlan insert on
   TX IPv4 checksum on
   TX TCP checksum on
   TX UDP checksum on
   TX SCTP checksum on
   TSO on
```

- Enable Generic Receive Offload on all enabled ports (reassembly timeout of 10 us):

```
# fp-shmem-ports --gro-timeout=10 --eth_port=enabled --offload gro on
```

fp-shmem-ready

Description

Display the name of the shared memory if it is ready for mapping, or *Not found* if it is not available.

The tool can be used in a script as a sentinel to synchronize multiple applications, because the process of adding a new very large shared memory instance may take a long while.

Synopsis

```
# fp-shmem-ready
```

Example

```
# fp-shmem-ready fp-shared
fp-shared
# fp-shmem-ready unknown-name
Not found
```

fp-track-dump

Description

Display the per core history of function names recorded in your application by the `FPN_RECORD_TRACK()` macro. Can help detect infinite loops.

Synopsis

```
# fp-track-dump
```

Example

```
myfunction()
  while () {
    FPN_RECORD_TRACK();
    ...
}

fp-track-dump
Core 1
       [23] PC=0x4ec59b RA=0x4e793e Func=myfunction:133 cycles=5383286
       [22] PC=0x4ec341 RA=0x4e793e Func=myfunction:133 cycles=1430467202
       [21] PC=0x4ec59b RA=0x4e793e Func=myfunction:133 cycles=5381148
       [20] PC=0x4ec341 RA=0x4e793e Func=myfunction:133 cycles=715474104
...
Core 2
       [31] PC=0x4ec59b RA=0x4e793e Func=myfunction:133 cycles=5383286
```

(continues on next page)

(continued from previous page)

```
      [30] PC=0x4ec341 RA=0x4e793e Func=myfunction:133 cycles=1430467202
...
```

fp-intercore-stats

Description

Display the state of intercore structures.

By default, only cores belonging to the intercore mask are displayed. To display all cores, use the `--all` parameter.

`fp-intercore-stats` can also display the number of cycles spent on packets that went through the pipeline.

Synopsis

```
# fp-intercore-stats
```

Examples

```
# fp-intercore-stats
Intercore information
      mask 0x4004
Core 2
ring <fpn_intercore_2>
  size=512
  ct=0
  ch=0
  pt=0
  ph=0
  used=0
  avail=511
  watermark=0
  bulk_default=1
  no statistics available
Core 14
ring <fpn_intercore_14>
  size=512
  ct=0
  ch=0
  pt=0
  ph=0
```

(continues on next page)

(continued from previous page)

```
  used=0
  avail=511
  watermark=0
  bulk_default=1
  no statistics available
```

```
# fp-intercore-stats --all
Intercore information
     mask 0x4004
Core 0 (NOT IN MASK)
ring <fpn_intercore_0>
 size=512
 ct=0
 ch=0
 pt=0
 ph=0
 used=0
 avail=511
 watermark=0
 bulk_default=1
 no statistics available
Core 1 (NOT IN MASK)
ring <fpn_intercore_1>
 size=512
 ct=0
 ch=0
 pt=0
 ph=0
 used=0
 avail=511
 watermark=0
 bulk_default=1
 no statistics available
Core 2
ring <fpn_intercore_2>
 size=512
 ct=0
 ch=0
 pt=0
 ph=0
 used=0
 avail=511
 watermark=0
```

(continues on next page)

(continued from previous page)

```
 bulk_default=1
 no statistics available
...
```

```
# fp-intercore-stats --cpu
Fast path CPU usage:
cpu: %busy     cycles   cycles/pkt  cycles/ic pkt
  2:   99%  697179716          829              0
  4:   51%  363169408            0           1729
  6:   54%  383451844            0           1825
  8:   <1%    6180544            0              0
 14:   <1%    5683196            0              0
 16:   51%  362776960            0           1727
 18:   54%  382313120            0           1820
 20:   <1%    6234228            0              0
average cycles/packets received from NIC: 2626 (2206989016/840180)
ic pkt: packets that went intercore
```

Control Plane Protection

Overview

This guide describes how to enable and configure the *Control Plane Protection* mechanism. Enabling this feature reduces the risk of dropping control packets when the target is under high load, or when the transmission link is overloaded.

In a network architecture, control packets are critical, since losing some of them has stronger consequences than losing data packets. For instance:

- losing ARP (Address Resolution Protocol) packets can make a gateway unreachable
- losing OSPF/BGP/... packets can make a network unreachable
- losing IKE packets can prevent the setup of IPsec security associations
- losing LACP (Link Aggregation Control Protocol) packets can bring a link down

Control Plane Protection is a mechanism that reduces the risk of dropping these control packets. It has an impact on performance, which can be tuned depending on the required throughput and criticity of losing control packets.

This guide describes how to enable and configure this *Control Plane Protection* mechanism.

Recognized packet types

The parser recognizes ARP, ICMP (Internet Control Message Protocol), ICMPv6, OSPF, VRRP, IKE, DHCP, DHCPv6, BGP, LACP, SSH, OpenFlow, JSON RPC (TCP (Transmission Control Protocol) port 7406), Stats Collector (TCP port 39090), BFD, DPVI packets. All can be encapsulated in VLAN, QinQ or FPTUN (Fast Path Tunneling Protocol).

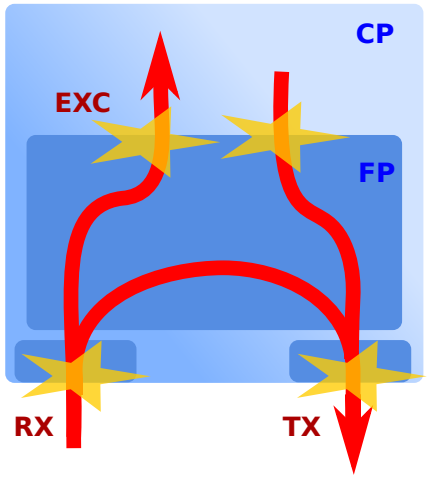
Supported platforms

This feature is only available on products based on DPDK. It requires a specific PMD API (Application Programming Interface) (called RX/TX descriptor status) which is available for the following PMDs: `i40e`, `ixgbe`, `mlx5`, `e1000`.

Design

Packet loss can occur at several places.

- RX: when the router is overloaded, the software may not able to dequeue the incoming packets fast enough. In that case, the hardware RX ring becomes full and the NIC starts to drop packets.
- TX: if the router tries to send more packets than what the network link support, the hardware TX ring becomes full and the software starts to drop packets.
- Exceptions (packets sent to Linux): when the exception rate is too high, the software ring becomes full and the software starts to drop packets.



Control Plane Protection is disabled by default in RX and TX, and always enabled for exceptions.

When enabled, the *Control Plane Protection* mechanism checks the filling level of the rings (RX, TX, or Exception) before enqueuing or dequeuing.

- If the filling level is below a threshold, nothing else is done and the packets are processed normally. Therefore, the additional CPU cost is very low in this situation.
- Else, if the filling level is above the threshold, the retrieved/transmitted packets are filtered: the data plane packets are dropped and the control plane are kept. Since most packets are data plane, the ring is emptied quickly and its filling level falls back below the threshold.

The filtering of packets is done by a parser. This gives more flexibility, but it implies a cost in terms of CPU cycles. To avoid spending all the CPU time to filter and drop without doing any effective processing:

- It is advised to have at least one core per 10Gb link: this ensures that even in the worst case, most CPU power is available for the processing application.
- A maximum CPU budget can be assigned to the *Control Plane Protection* to limit the impact on throughput performance. If the limit is exceeded, only a fraction of the packets will be filtered and the *Control Plane Protection* will be disabled for the other part. Therefore, control plane packets can be dropped in this situation.

Usage

Control Plane Protection is disabled by default. If control plane packets are lost on a given port, enable *Control Plane Protection* using the following command:

```
root@host# fp-cli dpdk-cp-filter-set eth0 rx_mode software-filter tx_mode software-
↪filter
rx cp filter is enabled: rx_mode=software-filter, rxd_thres=256, rxd_count=512
tx cp filter is enabled: tx_mode=software-filter, txd_thres=256, txd_count=512
```

To enable *Control Plane Protection* automatically on start, you can specify FPN-SDK option in fast-path.env:

```
root@host# vi /etc/fast-path.env
  [...]
  FPNSDK_OPTIONS="--rx-cp-filter-mode=phys:software-filter --tx-cp-
↪filter=phys:software-filter"
  [...]
```

Control Plane Protection works according to a maximum CPU budget (default value is 10%). If control plane packets are still dropped after enabling *Control Plane Protection*, it means that this budget has to be increased using the following command:

```
root@host# fp-cli dpdk-cp-filter-budget-set 15
cpu budget is 15%
```

As described in the above section, the *Control Plane Protection* works according to a queue threshold. The threshold can be configured by specifying the following in the FPN-SDK option in fast-path.env:

```
root@host# vi /etc/fast-path.env
  [...]
  FPNSDK_OPTIONS="--rx-cp-filter-threshold=phys:2048,virt:50%,excp:10%"
  [...]
```

Note: The value can be provided with a fixed value or a percentage by using the % keyword. The default value is 50%.

See also:

The 6WINDGate *Fast Path Baseline* documentation for more information about the `fp-cli` commands.

Customizing Packet Detection

tcp/udp services

The behavior of the control plane packet parser can be customized with '–cp-filter-services'. This option needs to be configured in the `FPNSDK_OPTIONS` variable of the `fast-path.env` configuration file.

This option takes as argument the list of tcp or udp services that should be marked as control packets.

Example:

```
: ${FPNSDK_OPTIONS:=--cp-filter-services=default,udp-src-68,udp-dst-67,tcp-srcdst-80}
```

By default, the cp-filter-services value is:

```
udp-dst-500,udp-dst-4500,udp-dst-67,udp-dst-68,udp-dst-546,udp-dst-547,udp-dst-3784,
udp-dst-3785,tcp-srcdst-179,tcp-srcdst-22,tcp-srcdst-6633,tcp-srcdst-7406,tcp-srcdst-
↪39090
```

to recognize ike, DHCP, DHCPv6, BGP, SSH, OpenFlow, JSON RPC, Stats Collector packets.

The `default` keyword can be set in cp-filter-services to still match these packets.

Note: ARP, ICMP, ICMPv6, OSPF, VRRP, DPVI packets are always recognized as control plane packets. L2 or L3 packets detection can not be customized.

vxlan

By default, the *Control Plane Protection* protects control plane packets encapsulated in a vxlan tunnel. Only the outer packets with an udp layer with the dst ports equals to 4789 are considered as VXLAN packets by the *Control Plane Protection*.

The VXLAN udp port can be configured to a different value, with the `--cp-filter-vxlan-port-filter` option of `FPNSDK_OPTIONS` variable in the `fast-path.env` configuration.

To not protect control plane packets encapsulated in a vxlan tunnel, the `--cp-filter-vxlan-port-filter` needs to be set to 0.

RX mode

The *Control Plane Protection* supports three differents mode in RX: software filter, hardware filter and dedicated queue.

The software or hardware filter works like described in the design section by checking the filling level of the RX rings. When the filling is over a threshold, the data plane packets are dropped and the control plane are kept. The only difference between these two modes is how the packets are parsed. In the first mode, the parsing is done in software and in the second mode is done by the hardware NIC. In the hardware filter mode, hardware flows are programmed into the NIC to mark the control plane packets.

The dedicated queue mode works differently. In this mode, the NIC is programmed to enqueue the control plane packets into a specific RX queue. The data plane packets are enqueued by the NIC on the other RX queues.

Thus, the dedicated queue mode impacts how the fast path cores polls a port. When no core/port mapping is provided in fast-path.env (i.e. CORE_PORT_MAPPING=auto), an extra queue is added for dedicated queue mode. If a core/port mapping has been provided, the last queue is used for control plane packets. In this case, be sure to have at least 2 queues configured.

The RX mode of the *Control Plane Protection* can be chosen by setting the `--rx-cp-filter-mode=all|phys|excp|virt:none|software-filter|hardware-filter|dedicated-queue` option in `FPNSDK_OPTIONS` variable of the `fast-path.env` configuration file.

The default RX mode is `none` (i.e. cp filter is disabled). The hardware filter and dedicated queue mode are only supported on Mellanox NIC. Other hardware NICs don't support to offload control plane packet detection. For these NICs, the software-filter mode is always used.

When the fast path has started, it's possible to check which RX mode has been configured on a specific device with the following command `fp-cli dpdk-cp-filter <port_id>`.

Hardware flows configured on the NIC can also be dumped with the the following command `fp-cli dpdk-cp-filter-hw-offload-flows <port_id>`.

It's also possible to change of RX mode dynamically with the following commands:

```
# fp-cli dpdk-cp-filter-set <iface_name> rx_mode hardware-filter
  (or software-filter or dedicated-queue or none to disable it)
```

TX mode

The *Control Plane Protection* supports only one mode in TX: software filter.

The TX mode of the *Control Plane Protection* can be chosen by setting the `--tx-cp-filter-mode=all|phys|excp|virt:none|software-filter` option in `FPNSDK_OPTIONS` variable of the `fast-path.env` configuration file.

The default TX mode is `software-filter` for exception devices, or `none` for other devices (i.e. cp filter is disabled).

When the fast path has started, it is possible to check which TX mode has been configured on a specific device with the following command `fp-cli dpdk-cp-filter <port_id>`.

It is also possible to change of TX mode dynamically with the following commands:

```
# fp-cli dpdk-cp-filter-set <iface_name> tx_mode software-filter
  (or none to disable it)
```

Statistics

Control Plane Protection provides statistics to monitor the number of filtered packets.

RX and TX *Control Plane Protection* statistics

The RX and TX *Control Plane Protection* statistics are available through the `ethtool` command.

```
root@host# ethtool -S eth0
  {...}
  fpn.rx_cp_passthrough: 0
  fpn.rx_cp_kept: 0
  fpn.rx_dp_drop: 0
  fpn.rx_cp_overrun: 0
  fpn.tx_cp_passthrough: 0
  fpn.tx_cp_kept: 0
  fpn.tx_dp_drop: 0
  fpn.tx_cp_overrun: 0
  {...}
```

When RX *Control Plane Protection* is enabled, `fpn.rx_cp_passthrough` is increased for each received packet when machine is not overloaded. These packets are processed normally without being analyzed.

If the machine is loaded (RX ring length exceeds the threshold) and the CPU budget is not reached, `fpn.rx_cp_kept` and `fpn.rx_dp_drop` will increase respectively for each control plane packet (kept) and for each data plane packet (drop).

If the CPU budget is exceeded, `fpn.rx_cp_overrun` is increased for each received packet. These packets are processed normally without being analyzed.

The same applies for TX.

See also:

- The *FPN-SDK Baseline* Usage section for more information about the available statistics.

Exception *Control Plane Protection* statistics

For exceptions *Control Plane Protection*, the statistics are available through `fp-shmem-dpvi`.

```
root@host# fp-shmem-dpvi
rx-ring[00] enq=00000025 deq=00000025 enq_err=00000000 enq_cp_kept=00000000 enq_dp_
↪drop=00000000 deq_err=00000000 deq_copyerr=00000000
tx-ring[00] enq=00000039 deq=00000039 enq_err=00000000 enq_cp_kept=00000000 enq_dp_
↪drop=00000000 deq_err=00000000 deq_copyerr=00000000
```

The *Control Plane Protection* on exceptions is always enabled, without CPU limit:

- `enq` is increased for each packet enqueued in the exception ring
- when the ring is more than half-full, `enq_dp_drop` is increased for each data plane packet dropped
- when the ring is more than half-full, `enq_cp_kept` is increased for each control plane packet enqueued in the ring

See also:

- The *FPN-SDK Baseline* Usage section for more information about the available statistics.

Hardware flow rules

Overview

This guide describes how to enable and configure the hardware-flow.

This feature is currently limited on e810 Intel NIC and only support a subset of the rules capabilities.

Supported platforms

This feature is only available on products based on DPDK. It requires a specific PMD which is available for the following PMDs: `ice`, `iavf`.

API

The API is based on DPDK rte-flow API which describe the matching packet header by header (aka pattern) and apply actions on it.

Usage

Adding a flow:

> fp-cli> dpdk-port-flow add <poort> patterns <item> [/ <items> . . .] / end actions <action> [/ <action> . . .] / end

Example:

```
fp-cli> dpdk-port-flow 0 add patterns eth / ipv4 spec src 192.168.1.2 / udp spec sport␣
↪10 / end
action rss / end
```

Removing the flow:

The same command line must be provide and will be use as a template to match the applied flow.

By using the same example:

```
fp-cli> dpdk-port-flow 0 del patterns eth / ipv4 spec src 192.168.1.2 / udp spec sport␣
↪10 / end
action rss / end
```

The list of flows can be displayed in both format, txt or json by using the following command:

```
fp-cli> dpdk-port-flow 0 show
fp-cli> dpdk-port-flow 0 show json
```


Items

This list is an exhaustive list of the Item supported by the fp-cli.

Ethernet

```
eth [spec [<src <addr>] [<dst <addr>] [type <type>]] [mask [src <addr>] [<dst <addr>]␣
↪[type <type>]]
```

spec [...]: Specify a field value to match, if its equivalent field in mask is 0, the field is ignored. The exact matching result being match = spec & mask.

mask [...]: Specify which bits in the spec field to keep, any bit at 0 is ignored in the matching (can be 0 or 1).

src: Ethernet source address to match in the header.

dst: Ethernet destination to match in the header.

type: Ethernet type to match in the header.

VLAN

```
vlan [spec id <vlanid> mask id <vlanid>]
```

spec [...]: Specify a field value to match, if its equivalent field in mask is 0, the field is ignored. The exact matching result being match = spec & mask.

mask [...]: Specify which bits in the spec field to keep, any bit at 0 is ignored in the matching (can be 0 or 1).

id: VLAN identifier.

IPv4

```
ipv4 [spec [src <ip-src>] [dst <ip-dst>]] [mask [src <ip-src>] [dst <ip-dst>]]
```

spec [...]: Specify a field value to match, if its equivalent field in mask is 0, the field is ignored. The exact matching result being match = spec & mask.

mask [...]: Specify which bits in the spec field to keep, any bit at 0 is ignored in the matching (can be 0 or 1).

src: IPv4 source address to match in the header.

dst: IPV4 destination to match in the header.

IPv6

```
ipv6 [spec [src <ip-src>] [dst <ip-dst>]] [mask [src <ip-src>] [dst <ip-dst>]]
```

spec [...]: Specify a field value to match, if its equivalent field in mask is 0, the field is ignored. The exact matching result being match = spec & mask.

mask [...]: Specify which bits in the spec field to keep, any bit at 0 is ignored in the matching (can be 0 or 1).

src: IPv6 source address to match in the header.

dst: IPV6 destination to match in the header.

UDP

```
udp [spec [sport <src-port>] [dport <dst-port>]] [mask [sport <src-port>] [dport <dst-
↪port>]]
```

spec [...]: Specify a field value to match, if its equivalent field in mask is 0, the field is ignored. The exact matching result being match = spec & mask.

mask [...]: Specify which bits in the spec field to keep, any bit at 0 is ignored in the matching (can be 0 or 1).

sport: source port to match in the header.

dport: destination port to match in the header.

TCP

```
tcp [spec [sport <src-port>] [dport <dst-port>]] [mask [sport <src-port>] [dport <dst-
↪port>]]
```

spec [...]: Specify a field value to match, if its equivalent field in mask is 0, the field is ignored. The exact matching result being match = spec & mask.

mask [...]: Specify which bits in the spec field to keep, any bit at 0 is ignored in the matching (can be 0 or 1).

sport: source port to match in the header.

dport: destination port to match in the header.

GTP-U

```
gtpu [spec [version <version> pt <pt> f-eh <f-eh> f-sn <f-sn> f-npdu <f-npdu>  msg-type
<msg-type> msg-len <msg-len> teid <teid>]] [mask [version <version> pt <pt> f-eh <f-eh>
f-sn <f-sn> f-npdu <f-npdu>  msg-type <msg-type> msg-len <msg-len> teid <teid>]]
```

spec [...]: Specify a field value to match, if its equivalent field in mask is 0, the field is ignored. The exact matching result being match = spec & mask.

mask [...]: Specify which bits in the spec field to keep, any bit at 0 is ignored in the matching (can be 0 or 1).

version: version to match in GTP header.

pt: Protocol Type to match in GTP header.

f-eh: Flag extended header.

f-sn: Flag Sequence Number

f-npdu: Flag N-PDU.

msg-type: Message type.

msg-len: Message length.

teid: Tunnel Endpoint Identifier.

GTP-PSC

```
gtp-psc [spec pdu-type <pdu-type>] [mask pdu-type <pdu-type>]
```

spec [...]: Specify a field value to match, if its equivalent field in mask is 0, the field is ignored. The exact matching result being match = spec & mask.

mask [...]: Specify which bits in the spec field to keep, any bit at 0 is ignored in the matching (can be 0 or 1).

pdu-type: PDU type.

GRE

```
gre [spec protocol <type>] [mask [protocol <type>]
```

spec [...]: Specify a field value to match, if its equivalent field in mask is 0, the field is ignored. The exact matching result being match = spec & mask.

mask [...]: Specify which bits in the spec field to keep, any bit at 0 is ignored in the matching (can be 0 or 1).

protocol: GRE protocol.

SCTP

```
sctp [spec [sport <src-port>] [dport <dst-port>]] [mask [sport <src-port>] [dport <dst-
↪port>]]
```

spec [...]: Specify a field value to match, if its equivalent field in mask is 0, the field is ignored. The exact matching result being match = spec & mask.

mask [...]: Specify which bits in the spec field to keep, any bit at 0 is ignored in the matching (can be 0 or 1).

sport: source port to match in the header.

dport: destination port to match in the header.

ICMP

```
icmp: [spec [type <type>] [code <code>]] [mask [type <type>] [code <code>]]
```

spec [...]: Specify a field value to match, if its equivalent field in mask is 0, the field is ignored. The exact matching result being match = spec & mask.

mask [...]: Specify which bits in the spec field to keep, any bit at 0 is ignored in the matching (can be 0 or 1).

type: ICMP type to match in the header.

code: ICMP code match in the header.

ICMPv6

```
icmp6 [spec [type <type>] [code <code>]] [mask [type <type>] [code <code>]]
```

spec [...]: Specify a field value to match, if its equivalent field in mask is 0, the field is ignored. The exact matching result being match = spec & mask.

mask [...]: Specify which bits in the spec field to keep, any bit at 0 is ignored in the matching (can be 0 or 1).

type: ICMP type to match in the header.

code: ICMP code match in the header.

Actions

RSS

```
rss types <types[, type[,...]]>
```

type: List of possible RSS fields to hash on. DPDK 19.11: geneve, ip, ipv4, ipv4-frag, ipv4-other, ipv4-sctp, ipv4-tcp, ipv4-udp, ipv6, ipv6-ex, ipv6-frag, ipv6-other, ipv6-sctp, ipv6-tcp, ipv6-tcp-ex, ipv6-udp, ipv6-udp-ex, l2-payload, l3-dst-only, l3-src-only, l4-dst-only, l4-src-only, nvgre, port, sctp, tcp, tunnel, udp, vxlan.

DPDK 20.11 additional hash type: ah, c-vlan, esp, eth, gtpu, l2-dst-only, l2-src-only, l2tpv3, l3-pre32, l3-pre40, l3-pre48, l3-pre56, l3-pre64, l3-pre96, pfcp, pppoe, s-vlan, vlan.

2.2.2 FPN-SDK Add-on for DPDK

Overview

FPN-SDK Add-on for DPDK is the DPDK-specific part of the fast path hardware abstraction layer.

Features

- EAL (Environmental Abstraction Layer) / performance tuning runtime configuration
- Integration with DPDK, especially for the `mbuf` API
- Implementation of FPVI using the DPVI Linux driver or TUN/TAP driver.
- Implementation of intercore APIs using `mbuf` headroom to store contexts
- Fast path plugins

Dependencies

6WINDGate modules

- *6WINDGate DPDK*
- *FPN-SDK Baseline*

Linux

- Linux >= 2.6.34 is recommended for correct support of Huge TLB.

Usage

FPN-SDK Add-on for DPDK automatically starts when you start the fast path with the following script:

```
# fast-path.sh start
```

See also:

For more information on how to start the fast path, see the *Fast Path Baseline* documentation.

Providing options

The FPN-SDK can take several arguments on the fast path command line. Most of the usual options are automatically added by the fast path start script while parsing the `fast-path.env` configuration file.

The configuration variables `EAL_OPTIONS` (for DPDK EAL options) and `FPNSDK_OPTIONS` (for FPN-SDK options) of the fast path configuration file may contain additional options that are described here.

Useful DPDK EAL options

Here are the most useful EAL options to set in the `EAL_OPTIONS` variable:

`--log-level`=<LOGLEVEL>
: Available since dpdk-1.8.0. Optional. Set the dpdk log level. When set to n, all dpdk drivers messages with a log level equal or below n are printed. LOGLEVEL can have one of the following values:

  - 1: `EMERG` System is unusable.
  - 2: `ALERT` Action must be taken immediately.
  - 3: `CRIT` Critical conditions.
  - 4: `ERR` Error conditions.
  - 5: `WARNING` Warning conditions.
  - 6: `NOTICE` Normal but significant condition.
  - 7: `INFO` Informational.
  - 8: `DEBUG` Debug-level messages.

Any other EAL option can be passed by modifying the `EAL_OPTIONS` variable.

See also:

For the full list of options, see the DPDK documentation.

FPN-SDK options

Here are the least common FPN-SDK options to set in the `FPNSDK_OPTIONS` variable:

Logmode option

Parameters

`--logmode``=[console|syslog]`
: Optional. Specify where the log of the fast path should be display: console or syslog. By default, the log are displayed with syslog.

Vlan stripping option

Parameters

`--vlan-strip`
: Optional. Enable the VLAN header stripping feature on incoming frames if supported by the hardware. By default, vlan stripping feature is disabled.

  Alternately, you can specify `: ${VLAN_STRIP:=on}` in the configuration file.

Port options

Parameters

`--nb-rxq``=[all|phys|virt]:[number of rx queues]`
: Optional. Specify the number of queues per port for each device type. Useful when using automatic mapping of cores to ports. By default, the maximum of queues is allocated to a port. Cannot be used together with -t.

  The different types of device that can be configured are `phys` (i.e. any physical network device like Intel or Mellanox NICS), `virt` (i.e. any virtual network device like vhost-user ports).

Core / Port binding

`-t [<core number>=<port number>/<core number>=<port number>:<port number>]`

core number Fast path logical core number preceded by `c`.

port number Number of port to poll. For each occurrence, one RX queue is created.

Specify how many RX queues logical cores poll on ports. Unspecified logical cores and ports are idle.

It is preferred to set the core / port binding in the fast path configuration file using the option `: ${CORE_PORT_MAPPING:=<value>}`.

If this parameter is not set, each core polls all the ports on the same numa socket. If there is not enough RX queues, only a subset of these cores will poll the port. If the maximal number of TX queues is smaller than the number of fast path cores, the port is configured with only one shared TX queue. The total number of queues per port must not exceed the NIC's hardware limit, else only shared queue will be used.

The hardware is responsible for distributing flows over queues (for instance, *via* RSS on Intel or Arm platforms).

This parameter overrides the *--nb-rxq* options described above.

Offloads

Description

To support offload features such as TSO or L4 checksum offloading to the NIC, or forwarding offload information from a guest to the NIC through a virtual interface, you must enable offloading in the fast path. You can then tune the offload features more precisely using `ethtool -K <iface> <feature> on|off`.

`--offload`
Enable the offload feature in the fast path.

Example

```
############################
##### FPN-SDK OPTIONS #####
############################

:${FPNSDK_OPTIONS:=--offload}
```

TX scatter

Parameters

`--tx-scatter`
: Optional. Enable the TX scatter feature if it is supported by the hardware. By default, TX scatter is disabled. If offload feature is enabled, TX scatter feature is enabled too.

Cryptography options

Description

You can tune the maximum available cryptographic sessions and the number of buffers allocated for cryptography.

`--crypto-max-sessions`
: Tune maximum available cryptographic sessions. Cryptography code will reserve space to store sessions in pools. If we are doing IPsec, we need one session per sa.

`--crypto-buffers`
: Tune number of buffers allocated for cryptography. Crypto library will allocate buffers from pools to store per buffer crypto parameters. This parameter is the size of the buffer pool.

Example

```
############################
##### FPN-SDK OPTIONS #####
############################

:${FPNSDK_OPTIONS:= --crypto-max-sessions=8192 --crypto-buffers=32768}
```

Cryptographic offloading mask

Description

Specify which fast path cores are able to do cryptographic operations for other cores. This cryptographic offloading is done only between core on the same numa node. By default crypto offloading is done only for decrypt operations and to cores that are not receiving traffic at this moment.

`-i` <crypto_offloading mask>
: Specify the fast path cores available for crypto offloading. By default, all cores specified in `fp_mask` are used. The format of the mask can be a list of cpus or a mask starting with 0x. Two specific strings are also available: none to disable the crypto offloading feature and all (the default value). It is preferred to set the cryptographic offloading mask in the fast path configuration file using the option *: ${CRYPTO_OFFLOAD_MASK:=<value>}*.

Example

```
: ${CRYPTO_OFFLOAD_MASK:=none}
```

- Statistics of offloaded cryptographic operations can be retrieved with:

  ```
  crypto-offload-stats
  ```

- It is possible to enable/disable the cryptographic offload for packets encryption. Be aware that enabling encyption offload for an IPsec tunnel that aggregates many flows managed by several fast path cores can cause IPsec errors due to the anti-replay window.

  ```
  crypto-offload-encrypt-set on|off
  ```

- It is possible to disable the cryptographic offload for small packets by setting the threshold of minimal size of data

  ```
  crypto-offload-threshold-set <min>
  ```

  <min> Minimal size of data to offload cryptographic operations. By default any packet can be offloaded (size set to 0)

- To avoid to send cryptographic offloading to core that receives high traffic, it is possible to disable the cryptographic offload to overloaded cores during a configurable time period.

  ```
  crypto-offload-timer-set <time>
  ```

  <time> Time, in microseconds, during an overloaded core is ignored for cryptographic offloading. Default value is 10 milliseconds

Intercore options

Description

You can tune the size of the intercore rings, used for pipeline implementation.

`--intercore-ring-size`
: One ring is allocated for each core, to store messages sent from other cores. Use this option to change the size of the rings. Default value is `CONFIG_MCORE_INTERCORE_RING_SIZE`, as specified in `/etc/6WINDGate/fpnsdk.config`.

  Alternately, you can specify `: ${INTERCORE_RING_SIZE:=<value>}` in the configuration file.

Intercore implementation

The following functions have been added to the DPDK `mbuf` api:

- `m_set_process_fct()`
- `m_call_process_fct()`

These functions use `mbuf` headroom to store a `fpn_callback` structure.

To avoid updating `mbuf` internal pointers, `m_prepend/m_adj` functions are not used, yet the checks on available lengths in headroom are the same.

Since we don't call `m_prepend/m_adj`, once `m_set_process_fct()` has been called, no operation can be done on `mbuf` until `m_call_process_fct()` is called.

Software scheduling implementation

The software scheduling API is implemented on top of the DPDK `librte_sched` library.

Linux / fast path communication

Parameters

-l <exception mask>
: Specify the fast path cores involved in polling packets coming from Linux over DPVI interfaces. By default, all cores specified in `fp_mask` are used. All packets locally sent by the Linux stack are forwarded to the fast path and processed by the selected cores.

--nb-fpvi-queue=[all|phys|virt|excp]:<number of queues>
: Specify the number of queues per fpvi port. Each port in fast path has a tuntap netdevice used to send/receive packets between fast path and linux kernel. The number of queues of the tuntap device can be configured with a different value in function of the fast path port devtype associated. The different devtypes for the fast path are: configured are `phys` (i.e. any physical network device like Intel or Mellanox NICS), `virt` (i.e. any virtual network device like vhost-user ports), `excp` (i.e. for fpn0).

RX/TX descriptors and thresholds

Parameters

--nb-mbuf=<total mbuf number>|<sock0-mbuf-number,sock1-mbufs-number,...>
: Specify the number of `mbufs` to add in the pool (default is 16384).

  It is preferred to set the number of mbufs in the fast path configuration file using the variable `: ${NB_MBUF:=<value>}`, as it allows to set the value to `auto` which lets the wizard calculate how many

mbufs are required. Indeed, the number of needed mbufs can not be easily computed manually as some features like TCP use additional mbufs.

Optimal performance is reached when there are as few mbufs as possible. However, mbuf allocation failure can lead to unexpected behavior.

--mbuf-rx-size=<size>
Specify the size of Rx data (excluding headroom) inside each `mbuf`. The default value is 2176. Changing this value may affect performance.

The equivalent option in the fast path configuration file is `: ${MBUF_RX_SIZE:=<value>}`.

--nb-rxd=[RX descriptor number]
Optional. Specify the number of RX descriptors allocated to the NIC. It is highly recommended to use a power of 2 to be compliant with any PMD. The minimal value is 64. Default is 128.

Important: For Intel Ethernet Controller XL710, specify 1024 RX descriptors.

--nb-txd=[TX descriptor number]
Optional. Specify the number of TX descriptors allocated to the NIC. It is highly recommended to use a power of 2 to be compliant with any PMD. The minimal value is 64. Default is 512.

--soft-queue=[<port number>=<additional TX desc number>/
default=<additional TX desc number>]
Optional. Add a software queue at FPN-SDK level. This is particularly useful for NICs where it is not possible to configure the number of Tx descriptors. For performance purpose this field must be a power of 2. The maximal value is 32768. Default is 0 (i.e. no software queue).

port number Fast path port number preceded by *p* or *default* to apply the configuration to all ports

additional TX desc number Number of additional TX descriptors allocated at FPN-SDK level.

Alternately, you can specify `: ${SOFT_TXQ_SIZE:=<value>}` in the configuration file. Only the default value can be modified this way.

Control Plane protection options

mode

Parameters

--rx-cp-filter-mode=[all|phys|virt|excp]:[none|software-filter|hardware-filter|dedicated-queue]

--tx-cp-filter-mode=[all|phys|virt|excp]:[none|software-filter]
Optional. Choose control plane protection mode on RX or TX for each device type.

The different types of device that can be configured are `phys` (i.e. any physical network device like Intel or Mellanox NICS), `virt` (i.e. any virtual network device like vhost-user ports), `excp` (i.e. any device in charge

to send packets in exception like fpvi using vhost).

For each device type, the following values are possible:

- phys: hardware-filter or dedicated-queue (only with Mellanox NICs, fallback in software-filter for other NICs) or software-filter or none (default value is none)
- virt: software-filter or none (default value is none)
- excp : software-filter or none in both RX and TX (default value is none in RX and software-filter in TX)

Example

```
############################
##### FPN-SDK OPTIONS #####
############################

${FPNSDK_OPTIONS:=--rx-cp-filter-mode=phys:hardware-filter,virt:software-filter --tx-
↪cp-filter-mode=all:none}
```

threshold

Parameters

--rx-cp-filter-threshold=[all|phys|virt|excp]:[threshold number][%]

--tx-cp-filter-threshold=[all|phys|virt|excp]:[threshold number][%]
Optional. Configure control plane protection threshold on RX or TX for software-filter or hardware-filter mode. The value can be provided with a fixed value or a percentage by using the % keyword. By default, control plane protection threshold is 50% in RX and TX.

Example

```
############################
##### FPN-SDK OPTIONS #####
############################

${FPNSDK_OPTIONS:=--rx-cp-filter-threshold=phys:2048,virt:10% --tx-cp-filter-
↪threshold=all:50%}
```

Managing RETA entries

RETAs (REdirection TAbles) are per-port configurable tables used by the NIC controllers' RSS filtering feature to select the RX queue into which to store an IP input packet. When receiving an IPv4 (Internet Protocol version 4) or an IPv6 (Internet Protocol version 6) packet, the controller computes a 32-bit hash based on:

- the source address and the destination address in the packet's IP header,
- the source port and the destination port in the UDP (User Datagram Protocol)/TCP header, if any.

The controller then uses the RSS hash value to compute a RETA (REdirection TAble) table index to get the number of the RX queue where to store the packet.

The DPDK API includes a function that is exported by PMDs to configure a port's RETA entries.

Note:

- The number or RETA entries depends on your NIC:

NIC	RETA entries
Intel 1GbE	128
Intel 10GbE	128
Intel 40GbE	512

- For test purposes, the `testpmd` application includes the following command to configure RETA entries:

```
port config X rss reta (reta_index,queue_number)[,(hash_index,queue_number)]
```

X Port for which to configure RETA entries.

hash_index Index of a RETA entry.

hash_index RX queue number to be stored in the RETA entry.

- Configure a port's RETA entries:

```
dpdk-rss-reta-set <Pi> index <i>[ <j>[ ...]] queue <Qj>
```

<Pi> Port number

index <i>[<j>[...]] Space-separated or tab-separated list of RETA entries.

queue <Qj> RX queue index to write in RETA entries.

- Display a port's RETA entries:

```
dpdk-rss-reta <Pi> [index <i> [<j>]]
```

<Pi> Port number.

index <i> Optional. RETA entry index.

index <i> [<j>] Optional. RETA entries range indices. By default, all RETA entries are displayed.

- Select the method used to compute the IP input packets RSS hash value:

```
dpdk-rss-hash-func-set <Pi> [<HF>[ <HF>[ ...]]]
      HF := { ipv4|ipv4-frag|ipv4-tcp|ipv4-udp|ipv4-sctp|ipv4-other|
              ipv6|ipv6-frag|ipv6-tcp|ipv6-udp|ipv6-sctp|ipv6-other|
              l2-payload|ipv6-ex|ipv6-tcp-ex|ipv6-udp-ex|
              port|vxlan|geneve|nvgre }
```

<Pi> Port number.

[<HF>[<HF>[...]]] Space-separated or tab-separated list of RSS hash functions among the following: `ipv4`, `ipv4-frag`, `ipv4-tcp`, `ipv4-udp`, `ipv4-sctp`, `ipv4-other`, `ipv6`, `ipv6-frag`, `ipv6-tcp`, `ipv6-udp`, `ipv6-sctp`, `ipv6-other`, `l2-payload`, `ipv6-ex`, `ipv6-tcp-ex`, `ipv6-udp-ex`, `port`, `vxlan`, `geneve`, `nvgre`. Disable RSS filtering is no hash function is supplied.

- Display the set of methods currently used to compute the IP input packets RSS hash value:

```
dpdk-rss-hash-func <Pi>
```

<Pi> Port number.

- Set the 40-byte RSS hash key used to compute the IP input packets RSS hash value:

```
dpdk-rss-hash-key-set <Pi> <key>
```

<Pi> Port number.

<key> 40-bytes key as a contiguous set of 80 hexadecimal digits (2 hexadecimal digits per byte).

- Display the current 40-byte RSS hash key used to compute the IP input packets RSS hash value:

```
dpdk-rss-hash-key <Pi>
```

<Pi> Port number.

Performance Optimization

This section presents some guidelines to get optimal performance from the fast path with FPN-SDK Add-on for DPDK.

Set up your BIOS

To get optimal performance, you may have to edit your BIOS settings.

For instance, the following parameters can enhance performance on an Intel Ivy-Bridge platform:

BIOS CPU Setting	Value
Intel(R) QPI Frequency Select	Auto Max
Intel(R) Turbo Boost Technology	Disabled
Enhanced Intel SpeedStep(R) Tech	Disabled
Processor C3	Disabled
Processor C6	Disabled
Intel(R) Hyper-Threading Tech	Enabled
Active Processor Cores	All
Execute Disable Bit	Enabled
Intel(R) Virtualization Technology	Enabled
Intel(R) VT for Directed I/O	Enabled
• Interrupt Remapping	Enabled
• Coherency Support	Disabled
• ATS Support	Enabled
• Pass-through DMA Support	Enabled
Intel(R) TXT	Disabled
Enhanced Error Containment Mode	Disabled
MLC Streamer	Enabled
MLC Spatial Prefetcher	Enabled
DCU Data Prefetcher	Enabled
DCU Instruction Prefetcher	Enabled
Direct Cache Access (DCA)	Enabled
Extended ATR	0x01

See also:

DPDK documentation (http://dpdk.org/doc/guides/linux_gsg/enable_func.html#enabling-additional-functionality)

Allocate logical cores to the fast path

NICs, PCI slots, ports and cores

NICs are attached to PCI slots, which are attached to sockets. NICs are recognized as fast path ports.

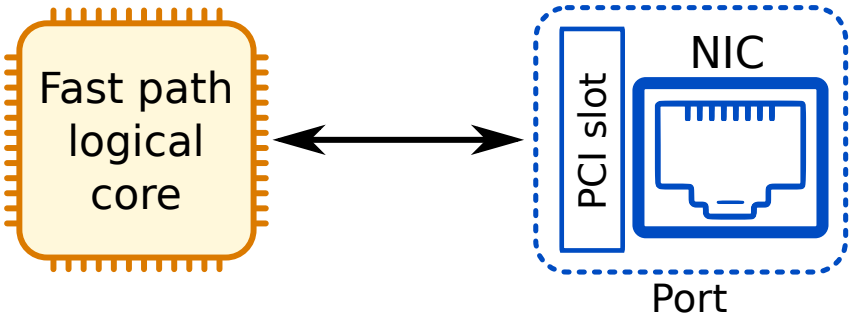


Fig. 1: Core - NIC relation

Cores allocation

When starting the fast path *via* the `fast-path.sh` script, some logical cores are allocated to the fast path, and other logical cores, to the other Linux processes.

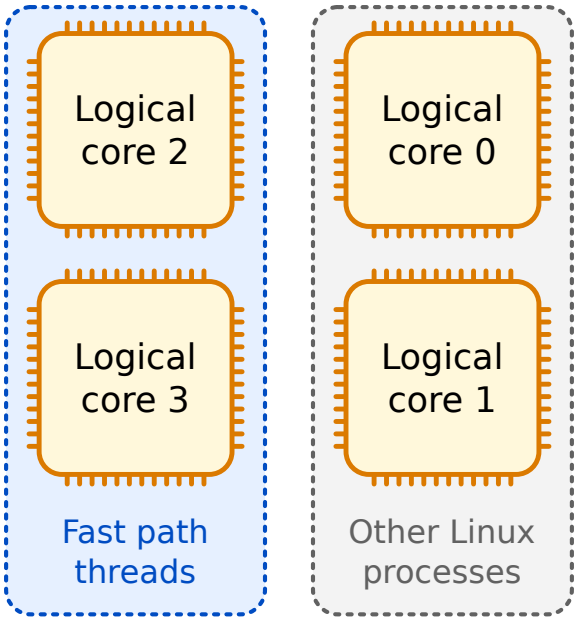


Fig. 2: Cores allocation to the fast path

You can specify which logical cores are allocated to the fast path *via* the `/etc/fast-path.env` fast path configu-

ration file `FP_MASK` parameter.

Important: Never allocate core `0` to the fast path.

Example

Allocate logical cores `2`, `3`, `18` and `19` to the fast path:

```
: ${FP_MASK:=2,3,18,19}
```

Note: Logical core numbering can change between platforms and should be determined beforehand.

Make sure logical cores poll only ports located on the same socket

As the fast path constantly polls the NICs, it is critical that logical cores allocated to the fast path do not poll ports located on a different socket. Otherwise, memory accesses constantly traverse the processor interconnect, which may lead to a bottleneck.

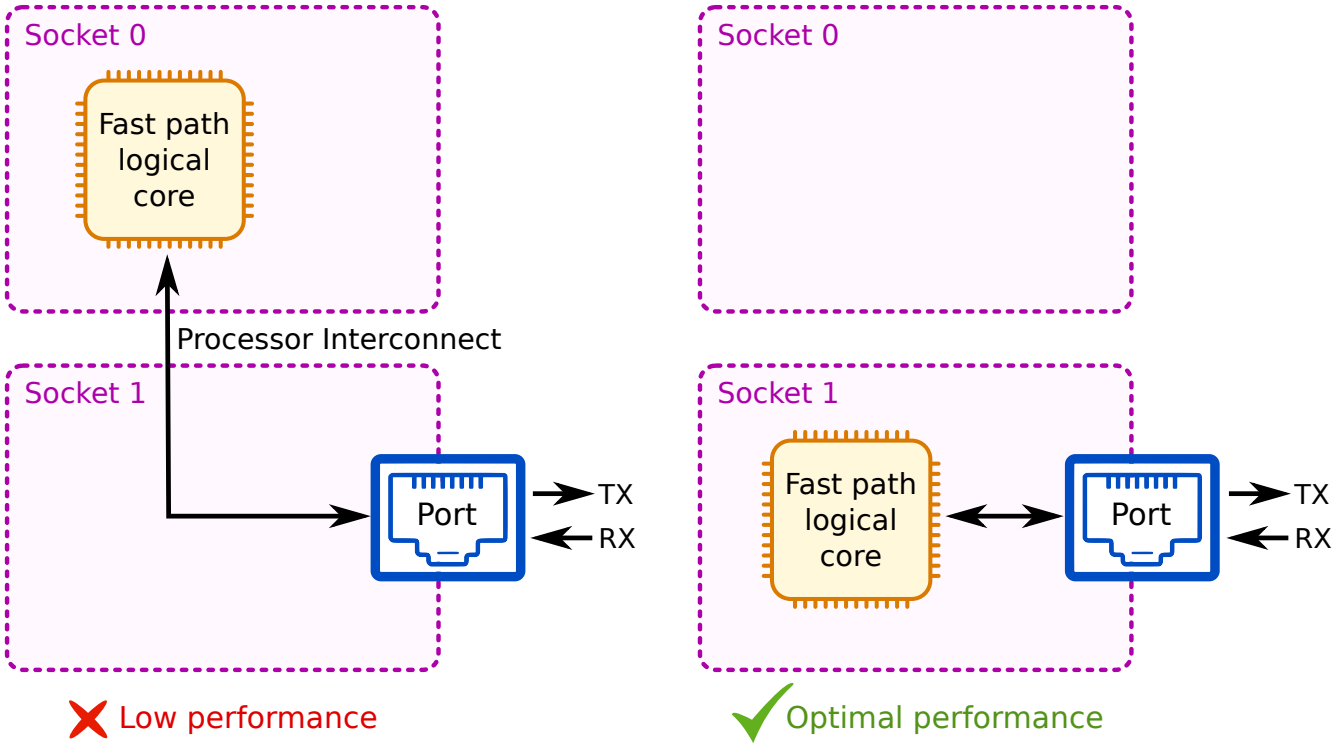


Fig. 3: Cores managing ports on another socket

Map ports and logical cores

If you set `CORE_PORT_MAPPING` manually in the fast path configuration file, you must map ports and logical cores to fit your network architecture, as illustrated in the topologies below. By default (when `CORE_PORT_MAPPING` is set to `auto`), the core/port mapping already matches these constraints.

To specify which fast path logical cores poll which ports, edit the fast path `fast-path.env` configuration file `CORE_PORT_MAPPING` parameter.

Topology 1: as many logical cores as ports on the same socket

When there are as many fast path logical cores as ports, each core should poll one different port:

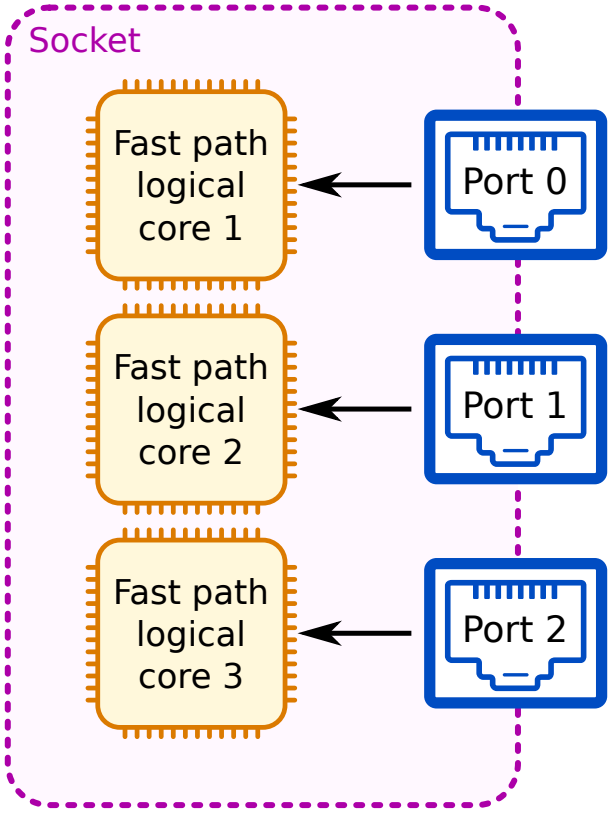


Fig. 4: As many cores as ports

Topology 2: not as many logical cores as ports on the same socket

When there are not as many fast path logical cores as ports, the easiest topology to implement is one of the following, where all fast path logical cores poll all ports. These topologies often offer a good support of borderline cases.

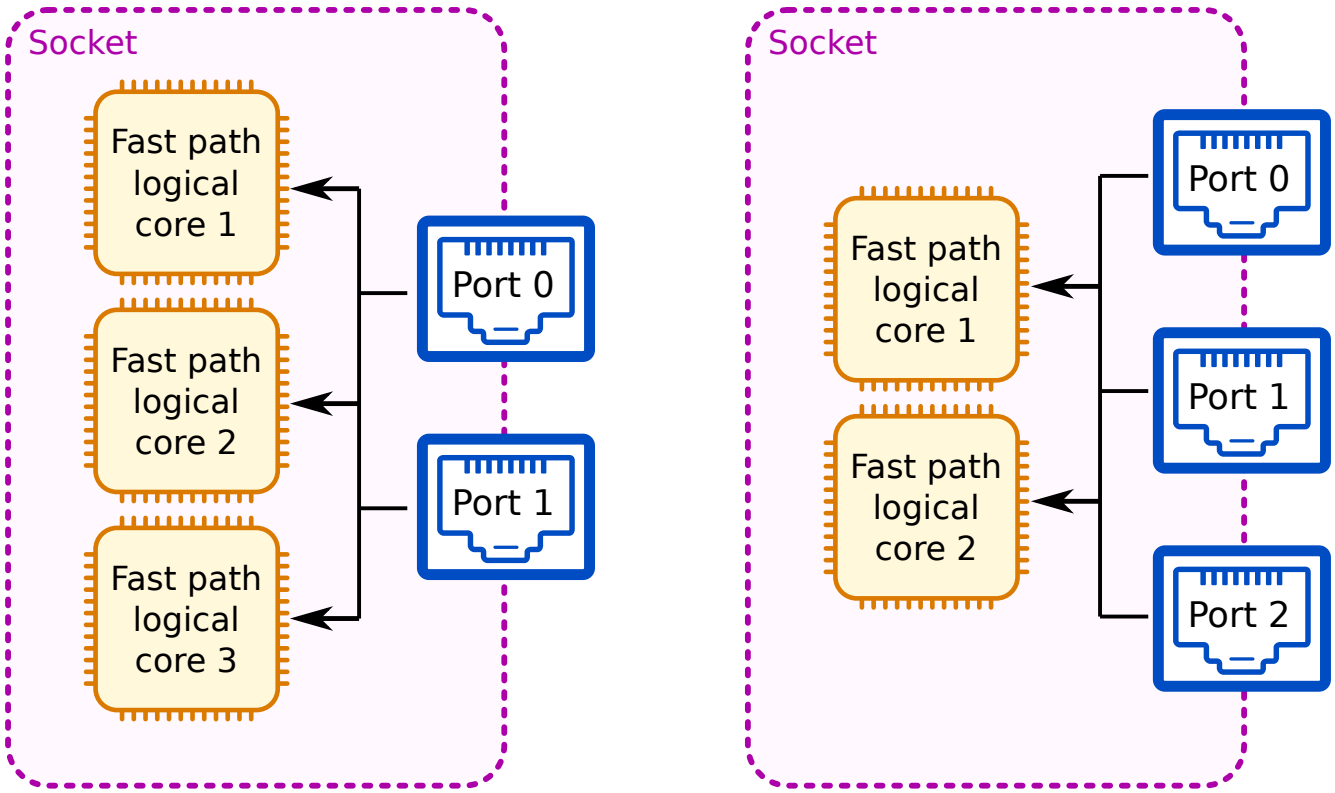


Fig. 5: Different number of cores and ports

You can also test other configurations to better suit your needs.

Prevent packets from being processed on different sockets

For even better performance, try to prevent too many packets from being processed on different sockets. This avoids overloading the processor interconnect.

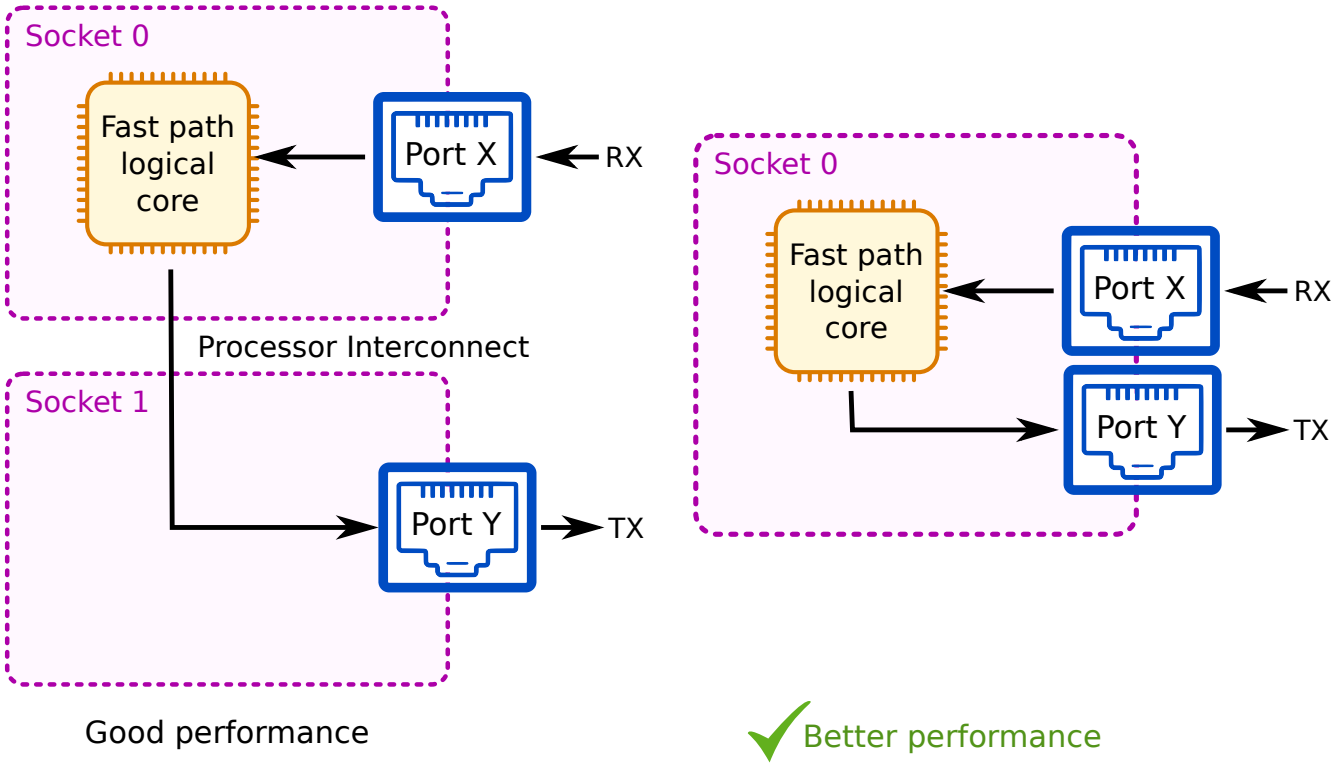


Fig. 6: RX and TX on different sockets

2.3 Fast Path Modules

This lists all the available fast path modules.

2.3.1 Fast Path Baseline

Overview

Fast Path Baseline is the foundation of the fast path packet processing framework.

Features

- Physical port detection
- Logical interface abstraction
- Statistics
- Internal debugging
- Plugins

- MACVLAN (MAC address based Virtual Local Area Network) devices support in the fast path
- Ability to offload linux egress packets to the fast path
- NUMA awareness (e.g. for LAG)

Dependencies

6WINDGate modules

- *FPN-SDK Baseline*

Linux

MACVLAN feature:

- Netlink notifications for MACVLAN mode is a kernel patch (upstream 2.6.33).

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=27c0b1a850cd
- Netlink notifications to put the lower interface in `promiscuous` and `allmulticast` modes is a kernel patch (upstream 3.13).

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=991fb3f74c14
- libnl3 (https://www.infradead.org/~tgr/libnl/files/libnl-3.2.25.tar.gz) >= 3.2.25

Ability to offload linux egress packets to the fast path:

- Support of the `egress clsact` queuing discipline is a kernel patch (upstream 4.5).

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=1f211a1b929c8
- Support of the `match-all` classifier is a kernel patch (upstream 4.8).

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=bf3994d2ed31

Usage

Start-up configuration of the fast path

When starting the fast path without configuration, the default behavior is to use all the supported physical network ports. The number of cores used by the fast path depends on the product:

- on a Turbo Router or 6WINDGate, all cores are enabled except the first one
- on a Virtual Accelerator, one physical core per socket is enabled

Each network port is polled by all logical cores from the same CPU socket.

The `fast-path.sh` script reads the `/etc/fast-path.env` configuration file before starting the fast path and associated modules.

A default `/etc/fast-path.env` configuration file template is provided. It defines the set of available options.

To customize the fast path configuration, you can either:

- use the fast path configuration wizard:

```
$ fast-path.sh config -i
```

- manually edit `/etc/fast-path.env` fast path configuration file.

Note: When changing the configuration, the fast path must be stopped.

The fast path configuration file

Syntax

In the fast path configuration file, you can use one of the following syntaxes:

`PARAMETER=value` Supersedes the corresponding global environment variable, if any. For instance: `FP_MASK=2-5`

`: ${PARAMETER:=value}` Ignored if the corresponding global environment variable exists. For instance: `: ${FP_MASK:=2-5}`

The template configuration file describes the list of available options and their syntax.

```
# Copyright 2015 6WIND S.A.
#
# This is the configuration file template for the fast path.
# Edit it prior to starting the fast path.
#   default location: /etc
#   default file name: fast-path.env

#
# NOTE: variables containing spaces must be quoted.
#

#####################
#### EAL OPTIONS ####
#####################

## FP_MASK defines which logical cores run the fast path.
```

(continues on next page)

(continued from previous page)

```
##
## The value can be:
## - 'min': one physical core per socket is enabled.
## - 'half': half of the cores are enabled, spread across sockets.
## - 'max': all physical cores except one are enabled.
## - 'auto' or commented out: equivalent to 'half' on a Turbo Router,
##   or 'min' on a Virtual Accelerator.
##
## - a list of logical cores ranges.
##   Example: "1-4,6,8" means logical cores 1,2,3,4,6,8.
## - an hexadecimal mask starting with '0x': it represents the mask of logical
##   cores to be enabled.
##   Example: "0x1e" means logical cores 1,2,3,4.
##
## Note: the core 0 is usually reserved for Linux processes and DPVI, so
## it's not advised to use it in FP_MASK.
##
## In expert mode (EXPERT_FPCONF=on), this parameter is mandatory and must not
## be auto.
##
#: ${FP_MASK:=auto}

## FP_MEMORY defines how much memory from the hugepages is reserved for
## the fast path in MegaBytes.
##
## The default architecture hugepage size will be used: 2MB on x86_64,
## 1GB on x86_64 if "hugepagesz=1G default_hugepagesz=1G" is passed to
## the kernel command line.
##
## The value can be:
## - 'auto' or commented out: the proper amount of memory will automatically
##   be reserved at fast path start.
## - an integer: representing the amount of memory accross all NUMA nodes.
##   The memory is automatically spread among nodes according to the
##   configuration.
##   Example: "1024" asks to reserve 1GB of memory for the fast path.
## - an integer, prefixed by '+': allocate the specified amount of memory
##   in addition to the automatically determined amount.
##   Example: "+512" allocates 512MB in addition to the amount required
##   by the fast path.
## - a list of integers, representing the amount of memory in MB to use on each
##   NUMA node.
##   Example: "256,512" asks to reserve 256MB on node 0 and 512MB on node 1.
##   Each element of the list can be prefixed with a "+": in this case,
```

(continues on next page)

(continued from previous page)

```
##   this amount will be added to the auto value for this node.
##   Example: "+512,+512" asks to allocate 512MB on each socket in addition
##   to the automatic value.
##
## In expert mode (EXPERT_FPCONF=on), the parameter is mandatory and its format
## must be a list of integer, one per socket.
##
#: ${FP_MEMORY:=auto}

## HUGEPAGES_DIR defines the hugepages mountpoint (hugetlbfs). The hugepages
## are used by the DPDK to allocate its memory, and they are reserved by the
## fast path startup script. They are available from a specific filesystem
## called hugetlbfs that has to be mounted.
## Refer to the DPDK documentation for details about hugepages.
##
## The value can be:
## - 'auto' or commented out: the default path (/dev/hugepages) is used.
## - a path to a directory in the filesystem.
##
## In expert mode (EXPERT_FPCONF=on), the parameter is mandatory.
##
#: ${HUGEPAGES_DIR:=/dev/hugepages}

## This parameter is deprecated. It is replaced by EXT_MEMORY.
##
## VM_MEMORY defines how much memory from the hugepages to allocate for
## virtual machines.
##
## Note: to properly replace VM_MEMORY by EXT_MEMORY, a '+' should be
## prepended to the value. For instance:
##   VM_MEMORY=123 should be replaced by EXT_MEMORY=+123
##   VM_MEMORY=123,456 should be replaced by EXT_MEMORY=+123,+456
#: ${VM_MEMORY:=auto}

## EXT_MEMORY defines how much memory from the hugepages to allocate
## for DPDK external memory, virtual machines and other applications.
##
## When external memory is used for DPDK mbuf pool, the fast path
## startup script is able to reserve additional memory stored in huge
## pages.
##
## When running the fast path as a host managing VMs, the fast path
## startup script is able to reserve additional memory stored in huge
## pages. This memory can be used by Qemu or libvirt for the virtual
```

(continues on next page)

(continued from previous page)

```
## machines.
##
## The value can be:
## - auto or commented out: if external mbuf pool is enabled, the proper
##   amount of memory will automatically be reserved at fast path start;
##   on a Virtual Accelerator, 4GB per socket will be reserved for VM,
##   on other products no VM memory will be reserved.
## - an integer: it represents the amount of memory in MB to reserve.
##   This amount will be spread equally on all NUMA nodes.
##   Example: "4096" asks to reserve 4GB for the virtual machines, distributed
##   on all the NUMA nodes of the machine (2GB per node if the machine has
##   2 nodes).
## - an integer, prefixed by '+': allocate the specified amount of memory
##   in addition to the automatically determined amount.
##   Example: "+512" allocates 512MB in addition to the amount required
##   by the fast path.
## - a list of integers, representing the amount of memory in MB
##   to reserve on each NUMA node.
##   Example: "2048,2048" asks to reserve 2GB on node0 and 2GB on node1
##   in huge pages.
##   Each element of the list can be prefixed with a "+": in this case,
##   this amount will be added to the auto value for this node.
##   Example: "+512,+512" asks to allocate 512MB on each socket in addition
##   to the automatic value.
##
## It is mandatory to stop or pause all VMs managed by fastpath to change this
## parameter otherwise the old value is kept.
## In expert mode (EXPERT_FPCONF=on), the parameter is mandatory and its format
## must be a list of integer, one per socket.
##
#: ${EXT_MEMORY:=auto}

## RESERVE_FP_HUGEPAGES enables or disables the allocation of
## huge pages used for the fast path by the startup script.
##
## When enabled (set to 'on' or commented out), the memory required to
## run the fast path is automatically reserved in the hugetlbfs at
## fast path start.
## Otherwise (set to 'off'), it is assumed that there are enough free huge
## pages reserved by the user to start the fast path. The user can
## allocate hugepages at boot time or dynamically (which is not supported
## on all kernels for 1G hugepages).
##
## In expert mode (EXPERT_FPCONF=on), the parameter is mandatory.
```

(continues on next page)

(continued from previous page)

```
##
#: ${RESERVE_FP_HUGEPAGES:=on}

## FP_PORTS defines the list of ports enabled in the fast path.
##
## The value can be:
## - 'all' or commented out: all supported physical ports are enabled.
## - a space-separated list of keys, defining the list of ports. A key
##   adds one or several ports to the current list, or removes them if
##   it is prefixed by '-'. The valid keys are: 'all', a pci identifier,
##   a linux interface name.
##
##   Example: "" means no port
##   Example: "all -mgmt0" means all ports except the one associated to
##   the interface called mgmt0 in Linux.
##   Example: "0000:03:00.0 0000:03:00.1" means the ports whose pci bus
##   addresses match.
##
##   A PCI bus can be suffixed by driver-specific arguments. For instance:
##   "0000:03:00.0,role=right,verbose=1".
##
##   The list is evaluated from left to right, so that
##   "eth0 0000:03:00.0 -all" means no port are enabled.
##
##   Note: be careful when using Linux interface names in configuration,
##   as they are subject to changes, depending on system configuration.
##
## This parameter is evaluated at fast path start and is converted into
## a whitelist or blacklist of PCI devices, that is passed to the fast
## path command line. Therefore, it is not possible to enable only some
## ports of a PCI device.
##
## In expert mode (EXPERT_FPCONF=on), this parameter is mandatory and
## must contain a list of PCI devices only.
##
#: ${FP_PORTS:=all}

## FP_VDEVS defines the list of static virtual devices enabled in the
## fast path.
##
## The value can be:
## - "" or commented out: no static virtual device is configured
## - a space-separated list of virtual devices, as in the *--vdev* DPDK EAL
##   option.
```

(continues on next page)

(continued from previous page)

```
##   The format is <driver-name><id>,cfg1=val1,cfg2=val2,...
##   Example: "pmd-vhost0,sockname=/tmp/pmd-vhost0,verbose=1".
##
## Refer to DPDK documentation for details about creation of virtual devices.
##
## Note: virtual devices created using hotplug once the fast path is running do
## not appear in the configuration.
##
#: ${FP_VDEVS:=''}

## EAL_ADDONS defines the list of EAL addons loaded by the fast path.
## An addon is a shared library that is loaded at fast path start. These
## addons usually provide new ethernet or crypto drivers.
##
## The value can be:
## - 'auto' or commented out: the required addons are automatically
##   loaded.
## - a string: the given list of addons are loaded.
##   Example: "librte_pmd_vhost.so"
## - a string prefixed by '+': the list is appended to the automatic
##   value.
##   Example: "+myaddon1.so myaddon2.so"
##
## In expert mode (EXPERT_FPCONF=on), this parameter is mandatory and
## should contain the list of EAL addons to be loaded at fast path start.
##
#: ${EAL_ADDONS:=auto}

## NB_MEM_CHANNELS defines the total number of memory channels on the machine.
## It is used by the fast path to spread the pool objects (like mbufs) among
## the different memory channels, thus increasing performance.
##
## The value can be:
## - 'auto' or commented out: the number of memory channels is determined
##   automatically when the fast path starts.
## - an integer between 1 and 32, representing the number of memory channels.
##
## This argument corresponds to the '-n' EAL option which is described
## in DPDK documentation.
##
## In expert mode (EXPERT_FPCONF=on), the parameter is mandatory.
##
#: ${NB_MEM_CHANNELS:=auto}
```

(continues on next page)

(continued from previous page)

```
## CRYPTO_TYPE defines the type of crypto driver to use.
##
## The value can be:
## - 'auto' or commented out: use hardware crypto if available, else use
##   fastest available software crypto
## - the driver name: "Generic Crypto", "Intel Multi-Buffer",
##   "Intel QuickAssist", "Cavium Octeon TX crypto", "Cavium Octeon TX2 crypto"
## - the driver config name: software, multibuffer, quickassist, octeontxcpt,
##   octeontx2cpt
##
## In expert mode (EXPERT_FPCONF=on), this parameter is unused. The
## crypto driver is selected from the addons list (EAL_ADDONS).
##
#: ${CRYPTO_TYPE:=auto}

## MACHINE_MEMORY is a hint used to limit all memory used by the fast path
## (hugepages, shm, mallocs...), so it can run on a machine with this amount
## of physical memory (in megabytes).
## The wizard tries to scale down the runtime parameters according to this
## value to decrease its memory consumption, in order to keep enough free memory
## for other purposes, like VM memory.
## The value can be:
## - 'auto' or commented out: the fast path memory will be limited by available␣
↪physical memory.
## - an integer: the fast path memory will be limited to the specified value.
##
#: ${MACHINE_MEMORY:=auto}

## EAL_OPTIONS specifies additional EAL options that are passed as-is
## to the fast path command line. They should not conflict with the rest
## of the configuration.
#: ${EAL_OPTIONS:='--log-level 4'}

#########################
#### FPN-SDK OPTIONS ####
#########################

## CORE_PORT_MAPPING maps fast path cores with network ports,
## specifying which logical cores poll which ports.
##
## The value can be:
## - 'auto' or commented out: each port is polled by all the logical
##   cores of the same socket.
## - a mapping string: it associates each logical core to a list
```

(continues on next page)

(continued from previous page)

```
##   of ports.
##   Example: "c1=0:1/c2=2/c3=0:1:2" means the logical core 1 polls
##   the port 0 and 1, the core 2 polls the port 2, and the core 3
##   polls the ports 0, 1, and 2.
##
## Note: the port identifiers are assigned in the following order:
## - static virtual devices (FP_VDEVS), in the order they appear.
## - physical ports, in the PCI bus order.
## - hotplug ports created once fast path is running.
##
#: ${CORE_PORT_MAPPING:=auto}

## NB_MBUF defines the total number of mbufs to add in the mbufs pools.
## The mbufs store the network packets inside the fast path. The optimal
## performance is reached when there are as few mbufs as possible. However,
## mbuf allocation failure can lead to unexpected behavior.
##
## The value can be:
## - 'auto' or commented out: the optimal number of mbufs is determined
##   automatically at fast path start.
## - an integer: in this case, it specifies the number of mbufs to allocate
##   accross all nodes. The minimum value is 16384, the maximum value is 16777215.
##   Example: NB_MBUF=16384
## - an integer prefixed with '+': in this case, it specifies an additional
##   number of mbufs which is added to the automatic value.
##   Example: NB_MBUF=+16384
## - a list of integers: it specifies the amount of mbufs to allocate on
##   each NUMA node. An entry can be prefixed by '+', meaning this number
##   is added to the auto value for this node.
##   Example: NB_MBUF=20000,50000
##
## In expert mode (EXPERT_FPCONF=on), the parameter is mandatory and its format
## must be a list of integer, one per socket.
##
#: ${NB_MBUF:=auto}

## MBUF_RX_SIZE specifies the size of Rx data (excluding headroom) inside
## each mbuf.
##
## The value is an integer between 512 and 64000. The default value is 2176.
##
## In expert mode (EXPERT_FPCONF=on), the parameter is mandatory.
##
#: ${MBUF_RX_SIZE:=2176}
```

(continues on next page)

(continued from previous page)

```
## MAINLOOP_DELAY_US defines the mainloop sleeping period in microseconds.
#: ${MAINLOOP_DELAY_US:=0}

## LINUX2FP_MASK defines the fast path cores that can receive packets
## from Linux. It must be included in FP_MASK.
##
## The value can be:
## - 'auto' or commented out: all fast path cores can receive packets from Linux.
## - a list of logical cores ranges.
##   Example: "2-4,6" means logical cores 2,3,4,6
## - an hexadecimal mask starting with '0x': it represents the mask of logical
##   cores to be enabled.
##   Example: "0x18" means logical cores 3,4
##
#: ${LINUX2FP_MASK:=auto}

## QOS_SCHEDULER_MASK defines the fast path cores dedicated for qos schedulers.
## It must be included in FP_MASK.
##
## The value can be:
## - 'none' or commented out: No fast path cores are reserved.
##   Example: "2-4,6" means logical cores 2,3,4,6
## - an hexadecimal mask starting with '0x': it represents the mask of logical
##   cores to be enabled.
##   Example: "0x18" means logical cores 3,4
##
#: ${QOS_SCHEDULER_MASK:=none}

## FP_OFFLOAD enables or disables the offload support in the
## fast path.
##
## To support offload features such as TSO or L4 checksum offloading to
## the NIC, or forwarding offload information from a guest to the NIC
## through a virtual interface, you must enable offloading in the fast
## path. You can then tune the offload features more precisely using
## "ethtool -K <iface> <feature> on|off".
##
## If set to 'auto', the effective configuration depends on the
## product: it is enabled on a Virtual Accelerator and disabled
## on Turbo Router and 6WINDGate.
## Else, it can be forced to "on" or "off".
##
## In expert mode (EXPERT_FPCONF=on), the parameter is mandatory
```

(continues on next page)

(continued from previous page)

```
## and must be set to "on" or "off".
##
#: ${FP_OFFLOAD:=auto}

## CRYPTO_OFFLOAD_MASK defines the fast path cores that can do crypto operations
## for other fast path cores. It must be included in FP_MASK.
## NB: the crypto offloading is always done on cores in the same numa node.
##
## The value can be:
## - 'all' or commented out: all fast path cores can do crypto operations for
##   other fast path cores.
## - 'none': no fast path cores can do crypto operations for other fast path
##   cores
## - a list of logical cores ranges.
##   Example: "2-4,6" means logical cores 2,3,4,6
## - an hexadecimal mask starting with '0x': it represents the mask of logical
##   cores to be enabled.
##   Example: "0x18" means logical cores 3,4
##
#: ${CRYPTO_OFFLOAD_MASK:=all}

## NB_RX_DESCRIPTORS defines the default number of descriptors configured
## for each Rx queue of Ethernet ports.
##
## If set to 'auto', the effective configuration is determined automatically
## depending on the devices.
##
#: ${NB_RX_DESCRIPTORS:=auto}

## NB_TX_DESCRIPTORS defines the default number of descriptors configured
## for each Tx queue of Ethernet ports.
##
## If set to 'auto', the effective configuration is determined automatically
## depending on the devices.
##
#: ${NB_TX_DESCRIPTORS:=auto}

## SOFT_TXQ_SIZE defines the size of the software Tx queue for each
## Ethernet ports. The value must be a power of 2.
##
#: ${SOFT_TXQ_SIZE:=0}

## NB_CRYPTO_SESSION the maximum number of cryptographic sessions.
##
```

(continues on next page)

(continued from previous page)

```
## The value can be:
## - an integer
## - unset or 'auto' to use the default value.
##
#: ${NB_CRYPTO_SESSION:=auto}

## NB_CRYPTO_BUFFER the maximum number of cryptographic buffers.
##
## Maximum number of 'in flight' operations, either being processed by the
## asynchronous crypto engine, or waiting in crypto device queues.
##
## The value can be:
## - an integer
## - unset or 'auto' to use the default value.
##
#: ${NB_CRYPTO_BUFFER:=auto}

## INTERCORE_RING_SIZE sets the size of intercore ring.
##
## The intercore ring isused by dataplane cores to send messages to
## another dataplane core.
## The default value is CONFIG_MCORE_INTERCORE_RING_SIZE, as specified in
## /etc/6WINDGate/fpnsdk.config.
##
## The value can be:
## - an integer
## - unset or 'auto' to use the default value.
##
#: ${INTERCORE_RING_SIZE:=auto}

## VLAN_STRIP forces to strip the VLAN header on incoming frames, if supported
## by the hardware.
##
## The value can be:
## - 'on' to enable the VLAN stripping.
## - unset or 'auto' to disable the VLAN stripping.
##
#: ${VLAN_STRIP:=off}

## FPNSDK_OPTIONS specifies additional FPNSDK options.
#: ${FPNSDK_OPTIONS:=--logmode=console}

####################
#### FP OPTIONS ####
```

(continues on next page)

(continued from previous page)

```
####################

## FP_OPTIONS specifies additional fast path options.
#: ${FP_OPTIONS:=''}

######################
#### FPVI OPTIONS ####
######################

## DPVI_MASK defines the cores allocated to exception packets processing.
##
## The value can be:
## - 'auto' or commented out: use the first non fast path core
## - a list of logical cores ranges.
##   Example: "0,9-11" means logical cores 0,9,10,11
## - an hexadecimal mask starting with '0x': it represents the mask of logical
##   cores to be enabled.
##   Example: "0xc0" means logical cores 6,7
##
#: ${DPVI_MASK:=auto}

## DPVI_RING_ENTRIES defines the number of entries in each DPVI ring.
##
## Default value is 4096. The value must be a power of 2.
#: ${DPVI_RING_ENTRIES:=4096}

## DPVI_OPTIONS specifies additional DPVI options.
#: ${DPVI_OPTIONS:=''}

####################
#### ADV OPTIONS ####
####################

## ADV_OPTIONS specifies advanced options specified through the wizard.
#: ${ADV_OPTIONS:=''}

## If EXPERT_FPCONF is set to 'on', the automatic calculation of some parameters
## is disabled. While not recommended, this can be needed if the configuration
## script is not able to parse the machine specifications and generates a
## wrong configuration.
##
#: ${EXPERT_FPCONF:=off}

## If DEBUG_FPCONF is set to 'on', additional debug logs are issued when the
```

(continues on next page)

(continued from previous page)

```
## configuration is parsed at fast path start.
##
#: ${DEBUG_FPCONF:=off}

## If FP_NAMESPACE is set to 'off', the creation of a netns to store fast-path
## interfaces is disabled. This can be needed for some specific interface types.
##
#: ${FP_NAMESPACE:=on}
```

fast path capabilities

You can change some fast path capabilities to tune the fast path according to your requirements in terms of scalability, for example, the maximum number of VRs (Virtual Routers) or the maximum number of netfilter rules managed by the fast path.

These parameters are set in the `fast-path.env` configuration file through the `FP_OPTIONS` variable and cannot be changed during a graceful restart.

See also:

- *Start-up configuration of the fast path*
- *Restarting the fast path with a different list of ports*

Configurable fast path capabilities are described in *Fast path capabilities* section.

Starting the fast path

1. Isolate the fast path from other system tasks using `cpusets` to avoid disrupting the fast path process.
2. Do one of the following:
   - Start the fast path with the default configuration file:

     ```
     # fast-path.sh start
     ```

   Or

   - Start the fast path with a custom configuration file:

     ```
     # CONF_FILE_fast_path=/path/to/conf/conf_file fast-path.sh start
     ```

> **Warning:** If you are connected to the target using the network, take care of removing this interface from the selected fast path ports, as any previous network configuration (IP addresses, MTU, routes, etc) is destroyed on interfaces assigned to the fast path.

See also:

- *Configuring cpusets*.

Stopping the fast path

Stop the fast path:

```
# fast-path.sh stop
```

If *Linux - Fast Path Synchronization* is active, it is stopped before the fast path.

Unloading the fast path

The fast path sometimes requires to remove some existing kernel modules before inserting up-to-date versions, or need some modules to be installed to work properly. These modules are automatically set up by the start procedure, but are not removed by the stop command, to avoid disturbing some other parts that uses the same kernel modules (management interface, for instance). Issue the unload command to completely uninstall the fast path:

```
# fast-path.sh unload
```

This commands first issue a stop command, then unload all modules that may have been installed by the fast path.

Displaying the fast path status

- Display the current status of running fast path threads:

  ```
  # fast-path.sh status
  ```

- Display the current status of running fast path threads and of the current installation (inserted `.ko`, for example):

  ```
  # fast-path.sh status complete
  ```

Applying a new fast path configuration

You can edit the `.env` fast path configuration file and restart the fast path to apply the new configuration.

Important: If you use *Control Plane OVS*, you must restart Open vSwitch after restarting the fast path.

See also:

The *Control Plane OVS* documentation.

Restarting the fast path with Linux - Fast Path Synchronization

If *Linux - Fast Path Synchronization* is active, you can configure your network via standard Linux commands. If you edit the list of ports handled by the fast path, you must re-apply your network configuration after restarting the fast path with the new configuration.

See also:

Start-up configuration of the fast path

Restarting the fast path with the same list of ports

If *Linux - Fast Path Synchronization* is active, and if you have not edited the list of ports handled by the fast path in the fast path configuration file, the fast path restarts with the new fast path configuration, and the current Linux network configuration is preserved.

1. Restart the fast path with the new fast path configuration:

```
# fast-path.sh restart
```

Restarting the fast path with a different list of ports

If *Linux - Fast Path Synchronization* is active, and if you have edited the list of ports handled by the fast path in the fast path configuration file, the Linux network configuration for these ports is erased when the fast path restarts. You must re-apply the Linux network configuration when the fast path has restarted.

1. Restart the fast path with the new fast path configuration:

```
# linux-fp-sync.sh stop
# fast-path.sh restart
# linux-fp-sync.sh start
```

   The Linux network configuration of ports handled by the fast path is erased.

2. Re-apply the Linux network configuration for the list of ports handled by the fast path.

Restarting the fast path without Linux - Fast Path Synchronization

If *Linux - Fast Path Synchronization* is not active, you can configure your network via the fast path CLI (Command Line Interface).

You must then force the fast path to restart with the current network configuration.

1. Restart the fast path with the new fast path configuration:

```
# fast-path.sh restart graceful
```

Saving/restoring linux networking configuration

If nothing is done, the linux networking configuration is lost after a fast path start or stop. To solve this problem, a tool named `networking-config` is available to save the linux networking configuration, diff it and restore it, using the netlink python-pyroute2 package.

The objects that can be restored are:

- interface state and mtu
- virtual interfaces (vlan, vxlan, macvlan, gre, gretap, ip6gre, ip6gretap, ipip, sit, ip6tnl, vti)
- IPv4 and IPv6 addresses
- IPv4 and IPv6 routes
- permanent neighbours
- bridge and bonding interfaces

Unsupported features:

- net namespaces
- policy based routing rules

To use the tool, you should save the configuration before starting or stopping the fast path, and restore it after.

Note: If you are using iptables, you should use `iptables-save` / `iptables-restore` to save and restore those rules in addition to `networking-config`.

Example of use:

```
# networking-config --help
usage: networking-config [-h] {save,diff,restore} ...

Manage the linux network configuration

positional arguments:
  {save,diff,restore}  sub-command help
    save               Save current configuration
    diff               Diff configuration with saved one
    restore            Restore saved configuration

optional arguments:
  -h, --help           show this help message and exit
```

(continues on next page)

(continued from previous page)

```
# networking-config save
# head /tmp/networking-config.json
{
  "routes": {
    "v4": {
      "192.168.0.0/24": [
        {
          "RTA_DST": "192.168.0.0",
          "family": 2,
          "dst_len": 24,
          "proto": 2,
          "ifaces": [
# ip addr add 192.168.0.1/24 dev eth1
# ip link set eth1 up
# networking-config diff
eth1|IFLA_CARRIER                                    0 => 1
eth1|IFLA_CARRIER_CHANGES                            1 => 2
eth1|IFLA_OPERSTATE                                  DOWN => UP
eth1|IFLA_QDISC                                      noop => pfifo_fast
eth1|flags                                           0x1002 => 0x11043
eth1|ipaddr|192.168.0.1/24|IFA_ADDRESS               {} => 192.168.0.1
eth1|ipaddr|192.168.0.1/24|IFA_FLAGS                 {} => 128
eth1|ipaddr|192.168.0.1/24|IFA_LABEL                 {} => eth1
eth1|ipaddr|192.168.0.1/24|IFA_LOCAL                 {} => 192.168.0.1
eth1|ipaddr|192.168.0.1/24|address                   {} => 192.168.0.1
eth1|ipaddr|192.168.0.1/24|prefixlen                 {} => 24
eth1|ipaddr|192.168.0.1/24|scope                     {} => 0
eth1|ipaddr|fe80::dced:1ff:fe41:4ca3/64|IFA_ADDRESS  {} => fe80::dced:1ff:fe41:4ca3
eth1|ipaddr|fe80::dced:1ff:fe41:4ca3/64|IFA_FLAGS    {} => 128
eth1|ipaddr|fe80::dced:1ff:fe41:4ca3/64|address      {} => fe80::dced:1ff:fe41:4ca3
eth1|ipaddr|fe80::dced:1ff:fe41:4ca3/64|prefixlen    {} => 64
eth1|ipaddr|fe80::dced:1ff:fe41:4ca3/64|scope        {} => 253
# networking-config save
# ip add del 192.168.0.1/24 dev eth1
# ip addr show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group␣
↪default qlen 1000
    link/ether de:ed:01:41:4c:a3 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::dced:1ff:fe41:4ca3/64 scope link
       valid_lft forever preferred_lft forever
# networking-config restore eth1  -v
networking-config:   INFO  starting system networking restoration
networking-config:   INFO  #### starting interfaces ####
networking-config:   INFO  ==== eth1 ====
```

(continues on next page)

(continued from previous page)

```
networking-config:  DEBUG  + state DOWN
networking-config:  DEBUG  + fe80::dced:1ff:fe41:4ca3/64
networking-config:  DEBUG  + mtu 1500, state UP
networking-config:   INFO  ==== eth1 restored ====
networking-config:   INFO  #### finished interfaces ####
networking-config:   INFO  #### starting routes ####
networking-config:   INFO  ==== eth1 routes ====
networking-config:   INFO  ==== eth1 routes restored ====
networking-config:   INFO  #### finished routes ####
networking-config:   INFO  system networking restoration ended
```

Configuring NUMA awareness

Some protocols can cause packets to be transmitted on a processor socket different from the one where they were received, which has an impact on performance (see *Performance optimization of FPN-SDK Add-on for DPDK* for more details).

For some protocols, it is possible to configure the fast path to avoid processor socket crossing and therefore avoid performance loss. This feature is called NUMA awareness.

As NUMA awareness may break compatibility with Linux and/or RFCs, it is disabled by default.

NUMA awareness can be enabled with the fast path CLI:

```
# fp-cli numa-aware-set on
```

For more details on the CLI commands to manage NUMA awareness, see *detailed NUMA awareness configuration*.

Configuring cpusets

`cpuset` is a kernel mechanism which is used to control the processor and memory placement of processes. It is accessible from userspace through the cpuset pseudo filesystem.

The fast path starting scripts support the isolation of the fast path process on DPDK architectures. It is implemented in `cpuset.sh`, which is called by `fast-path.sh`.

After the fast path is started, a new `cpuset` "system" is created. It includes all CPUs but fast path ones. All processes except the fast path are moved from the `root cpuset` to the `cpuset` "system".

On fast path stop, this `cpuset` is removed and all processes are moved back to the `root cpuset`.

You can configure or disable the cpuset feature in `/etc/cpuset.env`.

1. Display the status of cpusets:

```
$ cpuset.sh status
```

See also:

cpuset(7) (http://man7.org/linux/man-pages/man7/cpuset.7.html)

Managing virtual devices

Adding a new virtual port

You can dynamically create a new virtual device with the `fp-vdev` script.

```
fp-vdev add [-d pmd-vhost] [-f {txt,json}] [driver options] IFNAME
```

IFNAME Interface name

Optional arguments

-d vhost-pmd, **--driver** vhost-pmd
Driver to use (default: `vhost-pmd`).

-f {txt,json}, **--format** {txt,json}
Which format to use to display the new virtual device (default: `txt`).

Driver options for Virtio Host PMD

Default behavior is specified in the `/etc/fp-vdev.ini` file under the [pmd-vhost] section.

```
# Copyright 2018 6WIND S.A.

# Default configuration for pmd-vhost ports created with fp-vdev.
# available options are:
# profile = ['endpoint', 'nfv']
# sockmode = ['client', 'server']
# txhash = ['l3', 'l3l4']
# verbose = <debug level>
# sockfolder = /tmp
[pmd-vhost]
profile = endpoint
verbose = 0
sockmode = client
txhash = l3l4
sockfolder = /tmp
```

This behavior can be changed on the `fp-vdev` command line by providing one of the following options:

--sockpath SOCKPATH
: Set the UNIX socket path. This socket is used to negotiate features with QEMU and configure *Virtio Host PMD* (default: `/tmp/<IFNAME>.sock`).

--sockmode {client,server}
: Set the UNIX socket mode. This socket is used to negotiate features with QEMU and configure *Virtio Host PMD* (default: `client`). This default can be overwritten by the sockmode option in the `/etc/fp-vdev.ini` file.

--nb_rings NB_RINGS
: Set the number of rings to use. The default value is 1 (min=1, max=32 for *Virtio Host PMD*). This option is deprecated and will be removed in the next release.

--profile {endpoint,nfv}
: Define the usage profile (default: `endpoint`). In nfv mode, mergeable buffers are disabled, and auto txhash mapping tries to lower the lock contention, at the price of deteriorating the spreading among guest cores. This default can be overwritten by the profile option in the `/etc/fp-vdev.ini` file.

--txhash {l3l4,l3}
: Select which layers are used to calculate the Tx hash (default is `l3l4`). This default can be overwritten by the txhash option in the `/etc/fp-vdev.ini` file.

--verbose
: Enable debug log (default is `0`). This default can be overwritten by the verbose option in the `/etc/fp-vdev.ini` file.

Example

```
$ fp-vdev add tap1
 devargs:
   profile: endpoint
   sockmode: client
   sockname: /tmp/tap1.sock
   txhash: l3l4
   verbose: 0
 driver: pmd-vhost
 ifname: tap1
 rx_cores: all

$ fp-vdev add tap2 --profile nfv --verbose 1
 devargs:
   profile: nfv
   sockmode: client
   sockname: /tmp/tap2.sock
   txhash: l3l4
```

(continues on next page)

(continued from previous page)

```
  verbose: 1
driver: pmd-vhost
ifname: tap2
rx_cores: all
```

Deleting a virtual port

You can delete a virtual device with the `fp-vdev` script.

```
fp-vdev del IFNAME
```

IFNAME Interface name

Example

```
$ fp-vdev del tap1
```

The fast path configuration wizard

The configuration wizard simplifies the fast path configuration through simplified menus. It allows interactively selecting network ports, CPU cores, cryptographic hardware accelerators, etc.

Interactively configure the fast path

```
# fast-path.sh config -i

Fast path configuration
=======================

1 - Select fast path ports and polling cores
2 - Select a hardware crypto accelerator
3 - Advanced configuration
4 - Advanced plugin configuration
5 - Display configuration

S - Save configuration and exit
Q - Quit

Enter selection [S]:
```

The option “1 - Select fast path ports and polling cores” allows assigning physical ports to the fast path.

```
Fast path port selection
========================

Core/port mapping mode is auto

== ============ == ========== ============== ====================================
#  PCI or Name  Id Interface  Selected cores NIC full name
== ============ == ========== ============== ====================================
1  0000:01:00.0 0  mgmt0                     Intel Corporation 82575EB Gigabit Netw
2  0000:01:00.1 0  eth0                      Intel Corporation 82575EB Gigabit Netw
3  0000:03:00.0 0  eth1       auto           Intel Corporation 82599ES 10-Gigabit S
4  0000:03:00.1 0  eth2       auto           Intel Corporation 82599ES 10-Gigabit S
5  0000:06:00.0 0  eth3       auto           Intel Corporation 82599ES 10-Gigabit S
6  0000:06:00.1 0  eth4       auto           Intel Corporation 82599ES 10-Gigabit S
== ============ == ========== ============== ====================================

A - Add a virtual device
C - Switch to manual core/port mapping mode
M - Set the fast path core mask (current: auto)
E - Manually configure FP_PORTS (current: all -mgmt0 -eth0)

B - Back

Enter selection [B]:
```

Dump the fast path user configuration

```
# fast-path.sh config --dump --strip-comments
Configuring Fast Path...

: ${FP_MEMORY:=+512}
: ${FP_PORTS:='all -mgmt0 -eth0'}
```

The output may be empty if all parameters are set to their default values.

Dump the fast path fully-evaluated configuration

```
# fast-path.sh config --dump --strip-comments --full
Configuring Fast Path...

FP_MASK=1-23
FP_MEMORY=509,621
HUGEPAGES_DIR=/mnt/huge
VM_MEMORY=0
RESERVE_FP_HUGEPAGES=off
FP_PORTS='0000:03:00.0 0000:03:00.1 0000:06:00.0 0000:06:00.1'
FP_VDEVS=''
EAL_ADDONS=''
NB_MEM_CHANNELS=3
CRYPTO_TYPE=software
EAL_OPTIONS=''
CORE_PORT_MAPPING=auto
NB_MBUF=32768,49152
LINUX2FP_MASK=auto
FPNSDK_OPTIONS=''
DPVI_MASK=auto
EXPERT_FPCONF=off
DEBUG_FPCONF=off
```

This shows the effective configuration, which depends on:

- the user configuration
- the system (number of CPUs, NICs, ...)

Most automatic parameters are evaluated to their value (ex: amount of memory, fast path cores mask, ...). It corresponds to the configuration passed to the fast path.

This is useful to understand for instance how much memory or how many mbufs are reserved for the fast path.

Check the configuration validity:

```
# fast-path.sh config --check
Configuring Fast Path...

Check configuration
===================

WARNING - FP_MEMORY user value 500 (176 on this socket) is lower than auto value (240)␣
↪for socket 0
WARNING - FP_MEMORY user value 500 (323 on this socket) is lower than auto value (438)␣
↪for socket 1
```

(continues on next page)

(continued from previous page)

```
Configuration check done.
```

This mode displays warnings if the user configuration seems to be invalid.

Display the selected fast path ports and their polling cores

```
# fast-path.sh config --display
Configuring Fast Path...

Fast path configuration info
============================

  Selected ethernet card
  ----------------------

  Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01) PCI card␣
↪mounted on eth1 with cores 2,4,6,8,10,12,14,16,18,20,22
  Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01) PCI card␣
↪mounted on eth3 with cores 2,4,6,8,10,12,14,16,18,20,22
  Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01) PCI card␣
↪mounted on eth4 with cores 2,4,6,8,10,12,14,16,18,20,22
  Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01) PCI card␣
↪mounted on eth5 with cores 2,4,6,8,10,12,14,16,18,20,22
```

Display system information

```
# fast-path.sh config --machinedisplay
Configuring Fast Path...

General information
-------------------

  Processor name: Intel(R) Xeon(R) CPU           X5680  @ 3.33GHz
  Sockets 2, Cores 12, Hyperthreads 24
  Multi channel memory architecture: 3 channels

socket 0
--------

CPU information
  Core   0 = [  0- 12]  Core   1 = [  2- 14]  Core   2 = [ 16-  4]
```

(continues on next page)

(continued from previous page)

```
  Core   8 = [ 18-  6]  Core   9 = [  8- 20]  Core  10 = [ 10- 22]

Ethernet PCI information:
  Id           Interface  NIC full name
  ============ ========== ======================================================
  0000:03:00.0 eth1       Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network
  0000:03:00.1 eth3       Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network
  0000:06:00.0 eth4       Mellanox
  0000:06:00.0 eth5       Mellanox
  0000:0a:00.0 eth6       Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network
  0000:0a:00.1 eth7       Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network

Available crypto:
  Intel Multi-Buffer
  Cavium Networks Device 0011 (on bus 0000:0b:00.0)

socket 1
--------

CPU information
  Core   0 = [  1- 13]  Core   1 = [  3- 15]  Core   2 = [ 17-  5]
  Core   8 = [ 19-  7]  Core   9 = [  9- 21]  Core  10 = [ 11- 23]

Available crypto:
  Intel Multi-Buffer
```

Update an old configuration

```
# fast-path.sh config --update
```

or

```
# fast-path.sh config --update --file /path/to/myfile
```

The wizard parses the configuration and update it, removing deprecated options and adding comments. This may be useful when upgrading your version of 6WINDGate.

Unsupported options in the wizard

The following options cannot be configured in the wizard:

- `FP_OPTIONS`
- specific `FPNSDK_OPTIONS`, especially those related to the mapping of cores to crypto devices
- `EAL_OPTIONS`
- `MAINLOOP_DELAY_US`
- `DPVI_OPTIONS`

These options should be updated manually in the `fast-path.env` configuration file.

See also:

For details about what can be set in `EAL_OPTIONS` and `FPNSDK_OPTIONS`, refer to:

- The *FPN-SDK Baseline* and *FPN-SDK Add-on for DPDK* documentation

Fast path capabilities

Fast path capabilities can be tuned according to your requirements in terms of scalability. This is done through fast path runtime parameters that enable to configure, for example, the maximum number of VRs or the maximum number of Netfilter rules managed by the fast path.

Runtime parameters are set in the `fast-path.env` configuration file through the `FP_OPTIONS` variable. Changing them requires to stop and start the fast path.

For example to allow the fast path to manage up to 512 VRs:

```
$ fast-path.sh stop
$ vi /etc/fast-path.env
  [...]
  FP_OPTIONS="--max-vr=512"
  [...]
$ fast-path.sh start
```

For example to allow the fast path to manage up to 100K IPsec tunnels (IPV4 or IPV6):

```
$ fast-path.sh stop
$ vi /etc/fast-path.env
  [...]
  FP_OPTIONS="--mod-opt=ipsec:--max-sp=100000 --mod-opt=ipsec:--max-sa=200000"
  FPNSDK_OPTIONS="--crypto-max-sessions=400000"
  [...]
$ fast-path.sh start
```

Important: Changing the fast path capabilities through runtime parameters has an impact on the memory footprint.

The default value for configurable fast path capabilities indicated in the documentation will be reduced if the available memory is lower than 6GB. The actual value can be retrieved by the following fp-cli command:

```
<fp-0> conf runtime
Global runtime values are:
        max-ifnet    :512
        max-vr       :16
[snip]
vlan runtime values are:
        ifaces      :31
        hash-order  :5
macvlan runtime values are:
        ifaces       :31
        hash-order   :5
[snip]
```

See also:

- *Start-up configuration of the fast path*
- *Restarting the fast path with a different list of ports*

Main runtime parameters

- *Generic capabilities*
- *Routing capabilities*
- PBR *capabilities*
- *Filtering capabilities*
- *Bridge filtering capabilities*
- VLAN *capabilities*
- MACVLAN *capabilities*
- VXLAN *capabilities*
- GRE *capabilities*
- IPsec *capabilities*
- LAG *capabilities*

- *bridge capabilities*
- *Security*

For convenience, this section gathers the most important runtime parameters. The default values are indicated for Virtual Accelerator and may differ for other 6WIND products.

Generic capabilities

`--max-ifnet`
: Maximum number of logical interfaces. The Linux logical interfaces are mirrored in the fast path as ifnet. It includes physical ports and all virtual ports like ethgrp, VRRP, GRE, VLAN, vti etc.

  It must be at least greater than or equal to the maximum number of physical ports plus the number of VRs.

  Default value 2048

  Memory footprint per ifnet 3 KB

  Range 16 .. 50K

`--max-vr`
: Maximum number of VRs. Linux VRs based on network namespaces are mirrored as VR (Virtual Router) objects in the fast path.

  Be careful if the number of VRs is increased it can be necessary to increase some other capabilities too:

  - Number of PBR rules. See –mod-opt=pbr:–max-rules for details.
  - Number of IPv4 Netfilter rules. See –mod-opt=filter:–max-rules for details.
  - Number of IPv6 Netfilter rules. –mod-opt=filter6:–max-rules for details.

  Default value 16

  Memory footprint per vr 4.6 MB

  This memory footprint is mainly dependent of ipset. See –mod-opt=filter:–max-ipsets for details.

  Range 1 .. 2048

Routing capabilities

See *Fast Path Forwarding IPv4* See *Fast Path Forwarding IPv6*

PBR capabilities

See *Fast Path Policy-Based Routing*

Filtering capabilities

See *Fast Path Filtering IPv4* See *Fast Path Filtering IPv6*

Bridge filtering capabilities

See *Fast Path Filtering Ethernet Bridge*

VLAN capabilities

See *Fast Path VLAN*

MACVLAN capabilities

See *Fast Path MACVLAN*

VXLAN capabilities

See *Fast Path VXLAN*

GRE capabilities

See *Fast Path GRE*

IPsec capabilities

See *Fast Path IPsec IPv4*

LAG capabilities

See *Fast Path LAG*

bridge capabilities

See *Fast Path Ethernet Bridge*

Security

export FP_UNSECURE_MODE=yes

By default, the fast path will try to execute with the least privileges possible. That means it will only keep linux capabilities that it actually uses, and will also change from user "root" to user "fastpath" once initialization is finished.

Export this option in your `fast-path.env` configuration in order to disable this behavior and run fast path as root with full capabilities.

See also:

For more details about linux capabilities, see capabilities(7) (http://man7.org/linux/man-pages/man7/capabilities.7.html)

Advanced runtime parameters

- *Routing capabilities*
- *Filtering capabilities*
- VLAN *capabilities*
- MACVLAN *capabilities*
- VXLAN *capabilities*
- GRE *capabilities*
- IPsec *capabilities*
- LAG *capabilities*
- *bridge capabilities*

The following options are less commonly used, but may be useful from time to time.

Routing capabilities

See *Fast Path Forwarding IPv4*

Filtering capabilities

See *Fast Path Filtering IPv4* See *Fast Path Filtering IPv6*

VLAN capabilities

See *Fast Path VLAN*

MACVLAN capabilities

See *Fast Path MACVLAN*

VXLAN capabilities

See *Fast Path VXLAN*

GRE capabilities

See *Fast Path GRE*

IPsec capabilities

See *Fast Path IPsec IPv4* See *Fast Path IPsec IPv6*

LAG capabilities

See *Fast Path LAG*

bridge capabilities

See *Fast Path Ethernet Bridge*

Fast path plugins runtime parameters

You can also provide options to fast path plugins (or modules) using the following syntax:

```
--mod-opt <module_name>:<argument_value>
```

<module_name> Name of the module as specified in the module's `fp_mod object` structure.

<argument_value> Argument used in the module's `fp_mod object` structure `argv` field.

Several options may be provided to the same module in a single –mod-opt option:

```
--mod-opt <module_name>:<argument_value1>:<argument_value2>
```

Example

```
FP_OPTIONS="--mod-opt=my-module:-a --mod-opt=my-module:1 --mod-opt=my-module:--
↪lopt=8192"
```

or:

```
FP_OPTIONS="--mod-opt=my-module:-a:1:--lopt=8192"
```

The `my-module fp-mod` structure is updated with the following content:

- `argc` field is set to 4
- `argv` field is set to {"my-module", "-a", "1", "–lopt=8192"}

Then, `my-module` can use the `getopt` functions to perform as expected with the provided arguments.

See also:

For more information about creating plugins, see the *Fast Path Baseline - Plugins* documentation.

Maximum parameter values

Default maximum parameters values for fast path features are set in the `fp.config` file, generated once you built from sources, or delivered along with the executable files.

There are several ways to modify these values to suit your needs:

1. At compilation time:
   - Edit manually the `fp.config` file, or
   - Use the menuconfig textual interface to change values (refer to the *6WINDGate Build Framework* documentation).

   Use this `fp-cli` command to display parameters used at compile time:

   ```
   <fp-0> conf compiled
   ```

2. At runtime (values set at runtime will override compilation-time values).

   Use this `fp-cli` command to display current runtime values:

   ```
   <fp-0> conf runtime
   ```

   Note: Changing parameters at runtime is only available for DPDK and Octeon architectures.

See also:

- *Fast Path Command Line Interface*
- *Fast path capabilities*

The following table sums up what each parameter refers to (the parameters are all prefixed with `CONFIG_MCORE_`):

Maximum number of	Parameter	Module
FPN logical cores allocated to the fast path	FPN_MAX_CORES	*FPN-SDK Baseline*
Crypto sessions	CRYPTO_MAX_SESSIONS	*FPN-SDK Baseline*
IPv4 addresses	MAX_NB_ADDR4	Fast Path Baseline
IPv6 addresses	MAX_NB_ADDR6	Fast Path Baseline
Ports	MAX_PORT	Fast Path Baseline
VRs	MAX_VR	Fast Path Baseline
VNB (Virtual Networking Blocks) nodes	VNB_MAX_NS	Fast Path Baseline
PBR rules	PBR_MAX_RULES	*Fast Path Policy-Based*
Ebtables rules	EBT_MAXRULES	*Fast Path Filtering Ethe*
GRE tunnels	GRE_MAX	*Fast Path GRE*
ECMP maximum path	MAX_MPATH	*Fast Path Forwarding IP*
LAG master interfaces	LAG_MAX	*Fast Path LAG*

continues on

Table 1 – continued from previous page

Maximum number of	Parameter	Module
LAG slaves	LAG_SLAVES_MAX	*Fast Path LAG*
LAG slaves by interface	LAG_SLAVES_MAX_PER_IFACE	*Fast Path LAG*
Bridge FDB (Forwarding Database) entries	BRIDGE_FDB_MAX	*Fast Path Ethernet Bridg*
Bridge interfaces	BRIDGE_IFACE_MAX	*Fast Path Ethernet Bridg*
Bridge ports	BRIDGE_PORT_MAX	*Fast Path Ethernet Bridg*
IPsec SA (Security Association) entries	IPSEC_MAX_SA_ENTRIES	*Fast Path IPsec IPv4*
IPsec SP (Security Policy) entries	IPSEC_MAX_SP_ENTRIES	*Fast Path IPsec IPv4*
Trie input threshold	IPSEC_TRIE_IN_DEFAULT_THRESH_MAX	*Fast Path IPsec IPv4*
Trie rules	IPSEC_TRIE_MAXRULES	*Fast Path IPsec IPv4*
Trie output threshold	IPSEC_TRIE_OUT_DEFAULT_THRESH_MAX	*Fast Path IPsec IPv4*
SVTI tunnels	IPSEC_MAX_SVTI	*Fast Path IPsec SVTI - V*
XFRMI (Virtual XFRM Interfaces) interfaces	IPSEC_MAX_XFRMI	*Fast Path IPsec SVTI - X*
IPv4 conntracks	NF_CT_MAX	*Fast Path Filtering IPv4*
IPv4 NF cache size order	NF_MAX_CACHE_ORDER	*Fast Path Filtering IPv4*
IPv4 rules	NF_MAXRULES	*Fast Path Filtering IPv4*
IPv6 conntracks	NF6_CT_MAX	*Fast Path Filtering IPv6*
IPv6 NF cache size order	NF6_MAX_CACHE_ORDER	*Fast Path Filtering IPv6*
IPv6 rules	NF6_MAXRULES	*Fast Path Filtering IPv6*
MACVLAN interfaces	MACVLAN_IFACE_MAX	*Fast Path MACVLAN*
MACVLAN link interfaces	MACVLAN_LINKIFACE_MAX	*Fast Path MACVLAN*
Multicast IPv4 GRP	MULTICAST4_GRP_MAX	*Fast Path Multicast IPv4*
Multicast IPv4 MFC	MULTICAST4_MFC_MAX	*Fast Path Multicast IPv4*
Multicast IPv6 GRP	MULTICAST6_GRP_MAX	*Fast Path Multicast IPv6*
Multicast IPv6 MFC	MULTICAST6_MFC_MAX	*Fast Path Multicast IPv6*
Reassembly IPv4 inter-fragment time (ms)	MAX_IPV4_INTERFRAG_MS	*Fast Path Reassembly IP*
Reassembly IPv4 time in queue (ms)	MAX_IPV4_REASS_MS	*Fast Path Reassembly IP*
Reassembly IPv6 inter-fragment time (ms)	MAX_IPV6_INTERFRAG_MS	*Fast Path Reassembly IP*
Reassembly IPv6 time in queue (ms)	MAX_IPV6_REASS_MS	*Fast Path Reassembly IP*
VLAN interfaces	VLAN_IFACE_MAX	*Fast Path VLAN*
VLAN lower interfaces	VLAN_LOWER_MAX	*Fast Path VLAN*
VXLAN FBD entries	VXLAN_FDB_MAX	*Fast Path VXLAN*
VXLAN interfaces	VXLAN_IFACE_MAX	*Fast Path VXLAN*
VXLAN ports	VXLAN_PORT_MAX	*Fast Path VXLAN*

Fast Path CLI

The fast path CLI is a tool to manage fast path configuration while it is running.

Fast Path Command Line Interface

Starting the CLI

```
# fp-cli
<fp-0>
```

You can launch `fp-cli` with *Linux - Fast Path Synchronization* enabled or disabled:

Linux - Fast Path Synchronization **enabled** You can launch `fp-cli` right away.

Linux - Fast Path Synchronization **disabled** You must launch `fp-init`, then `fp-cli`.

Example

This example assumes that *Linux - Fast Path Synchronization* is enabled. It illustrates how to:

- enable the `eth1` interface,
- launch `fp-cli`,
- dump the logical interfaces table.

```
# ip link set up dev eth1
# fp-cli
<fp-0> iface
1:lo [VR-0] ifid=1 (virtual) <UP|RUNNING|FWD4|FWD6> (0x63)
          type=loop mac=00:00:00:00:00:00 mtu=16436 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
2:eth3 [VR-0] ifid=2 (port 2) <FWD4|FWD6> (0x60)
          type=ether mac=00:1b:21:c5:7f:76 mtu=1500 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
3:eth2 [VR-0] ifid=3 (port 1) <FWD4|FWD6> (0x63)
          type=ether mac=00:1b:21:c5:7f:75 mtu=1500 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
4:eth1 [VR-0] ifid=4 (port 0) <UP|RUNNING|FWD4|FWD6> (0x63)
          type=ether mac=00:1b:21:c5:7f:74 mtu=1500 tcp4mss=0 tcp6mss=0
          IPv4 routes=2  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
5:eth0 [VR-0] ifid=5 (virtual) <FWD4|FWD6> (0x60)
          type=ether mac=00:21:85:c1:82:58 mtu=1500 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
6:eth4 [VR-0] ifid=6 (port 3) <FWD4|FWD6> (0x60)
          type=ether mac=00:1b:21:c5:7f:77 mtu=1500 tcp4mss=0 tcp6mss=0
```

(continues on next page)

(continued from previous page)

```
        IPv4 routes=0  IPv6 routes=0
        if_ops: rx_dev=none tx_dev=none ip_output=none
```

Initialization

Detecting Linux devices and configuring interface names and MAC addresses

- Detect Linux devices previously created that represent physical ports
- Automatically configure interface names and MAC addresses

```
fp-init
```

Example

```
<fp-0> fp-init
Adding interface eth0 (ifuid 7) to port 0
Adding interface eth1 (ifuid 8) to port 1
Adding interface eth2 (ifuid 9) to port 2
Adding interface eth3 (ifuid 10) to port 3
```

How to find a command

1. List available commands name sort by fp-cli module:

   ```
   list [<module name>]
   list-deprecated [<module name>]
   ```

   <module name> Name or part of a fp-cli module name.

   Note: It's not recommended to use `deprecated` commands. See 'help <deprecated command>' to find the command(s) to use as replacement.

Example

```
<fp-0> list bridge

Available commands for fp-bridge:
  bridge
  bridge-fdb
  bridge-port-set
  bridge-fdb-hitflags-set
<fp-0> list-deprecated bridge

Available commands for fp-bridge:
  bridge-dump
  bridge-dump-fdb
  dump-l2-stats
```

2. List available commands with their help:

```
help [<command name>]
```

<command name> A command name.

Note: The word `help` can be placed anywhere in the command line.

Example

```
<fp-0> help bridge-fdb
bridge-fdb                : Dump fp-bridge FDB
                bridge-fdb <ifname>
<fp-0> help bridge-dump-fdb
bridge-dump-fdb           : Dump fp-bridge fdb
                bridge-dump-fdb <ifname>
This command is deprecated, use new command(s):
                bridge-fdb <ifname>
```

3. Search for commands from a keyword.

```
find <pattern>
```

<parttern> Word contained in a command name.

Example

```
<fp-0> find fdb
bridge-fdb: Dump fp-bridge FDB
                bridge-fdb <ifname>
bridge-fdb-hitflags-set: Set bridge FDB hitflags parameters
                bridge-fdb-hitflags-set <period_in_seconds> <max_scanned> <max_
↪sent>
vxlan-fdb: Display VXLAN FDB
                vxlan-fdb <ifname>
```

VRF support

The default VR to which all commands apply is displayed in the fp-cli prompt: <fp-X> indicates that commands apply to VRF X.

1. To execute a command in a specific VRF, use `vrf-exec` command:

```
vrf-exec <vrfid>|all <command name> [<command arguments>]
```

<vrfid>|all Execute the command only in the VRF `vrfid` or in all VRF.

<command name> [<command arguments>] Command to execute, with its arguments.

Note: `vrf-exec` <cmd> returns an error when:

- <cmd> is an invalid `fp-cli` command

Example

```
<fp-0> vrf-exec 0 route4
vrf0:
# - Preferred, * - Active, > - selected
0.0.0.0/0  [02]  NEIGH gw 10.0.2.2 via eth0-vr0 (8)

<fp-0> vrf-exec all route4
vrf0:
# - Preferred, * - Active, > - selected
0.0.0.0/0  [02]  NEIGH gw 10.0.2.2 via eth0-vr0 (8)

vrf1:
# - Preferred, * - Active, > - selected
```

(continues on next page)

(continued from previous page)

```
vrf3:
# - Preferred, * - Active, > - selected

<fp-0>
```

2. To change the current VRF, use the `vrf-set` command:

```
vrf-set <vrfid>

<vrfid>
   The new |vrf| id.
```

Example

```
<fp-0> route4
# - Preferred, * - Active, > - selected
0.0.0.0/0  [02]  NEIGH gw 10.0.2.2 via eth0-vr0 (8)
<fp-0> vrf-set 1
 New reference for VRF: 1
<fp-1> route4
# - Preferred, * - Active, > - selected
<fp-1> so-portset
portset tcp vrfid 1 16384-32767
portset udp vrfid 1 16384-32767
<fp-1>
```

JSON support

Some `fp-cli` commands can also write their output in JSON format. This is done by suffixing the command name with "-json".

Note: You can display the help of all JSON commands with `fp-cli find json`.

Example

```
<fp-0> iface
2:eth0 [VR-0] ifid=2 (virtual) <UP|RUNNING|FWD4|FWD6> (0x1b)
        type=ether mac=de:ad:de:01:02:03 mtu=1500 numa=0 tcp4mss=0 tcp6mss=0
        IPv4 routes=4  IPv6 routes=0
        if_ops: rx_dev=none tx_dev=none ip_output=none
<fp-0> iface-json
[
  {
    "vrfid": 0,
    "ifaces": [
      {
        "tcp6_mss": 0,
        "flags": [
          "up",
          "running",
          "fwd4",
          "fwd6"
        ],
        "name": "eth0",
        "ipv6_routes": 0,
        "ifid": 2,
        "ipv4_routes": 4,
        "mac": "de:ad:de:01:02:03",
        "tcp4_mss": 0,
        "vrfid": 0,
        "numa": 0,
        "mtu": 1500,
        "master": null,
        "port": "virtual",
        "type": "ether"
      },
    ]
  }
]
```

Displaying the command status in JSON format

This command displays the status returned by the last executed command.

Note: The command execution is regard as failed if the result is not `0`.

```
cmd-status-json
```

Example

```
<fp-0> route4 1.0.0.1
Invalid argument(s)
Usage: route4 [dst <addr dst>|<addr dst/prefix> [src <addr src>]]|[type TYPE]
                        TYPE := [all|fpm|local|neigh|connected|black]
<fp-0> cmd-status-json
{
  "result": -1
}
<fp-0> route4
# - Preferred, * - Active, > - selected
0.0.0.0/0  [03]  NEIGH gw 10.0.2.2 via eth0-vr0 (8)
<fp-0> cmd-status-json
{
  "result": 0
}
```

Displaying statistics in JSON format

This command displays all network statistics.

```
stats-json [agg[regated]]
```

agg[regated] Display sum of Linux and fast path statistics.

NUMA awareness configuration

To enable or disable NUMA awareness:

```
numa-aware-set [<protocol>] on|off
```

Example

```
<fp-0> numa-aware-set lag on
protocol lag numa awareness is on (was off)
```

The list of protocols that can be NUMA aware and their state can be displayed with the following command:

```
numa-aware
```

Example

```
<fp-0> numa-aware
Numa aware configuration:
     protocol lag is off
```

This same list can be displayed in JSON format:

```
numa-aware-json
```

Example

```
<fp-0> numa-aware-json
[
   {
      "protocol_name": "lag",
      "numa_aware": "off"
   }
]
```

Interface management

Displaying the logical interfaces table

```
iface [<name>]
```

<name> Only display information about this interface. If unset, all the interfaces are displayed.

Example

```
<fp-0> iface
8:eth1 [VR-0] ifid=8 (port 0) <UP|RUNNING|FWD4> (0x23)
        type=ether mac=00:1b:21:c5:7f:74 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=2  IPv6 routes=0
9:eth3 [VR-0] ifid=9 (port 2) <UP|RUNNING|FWD4> (0x23)
        type=ether mac=00:1b:21:c5:7f:76 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=3  IPv6 routes=0

<fp-0> iface eth1
8:eth1 [VR-0] ifid=8 (port 0) <UP|RUNNING|FWD4> (0x23)
        type=ether mac=00:1b:21:c5:7f:74 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=2  IPv6 routes=0
```

Displaying MACVLAN devices

```
macvlan
```

Example

```
<fp-0> macvlan
eth2-vr0:
        eth2.mv0-vr0: mode: private
```

Port management

Displaying the physical ports table

```
port
```

```
<fp-0> port
0: eth1-vr0 cached ifp=0x7f2d7f497640
2: eth3-vr0 cached ifp=0x7f2d7f497f00
```

Changing the number of fast path cores polling a port dynamically

Define which fast path cores poll the port. Default value is all.

If the port configuration is changed with this command, the port must be disabled and re-enabled to apply the configuration.

```
dpdk-port-rxcores-set <Pi>|<iface> <Ci>[:<Cj>[...]]|all|nb=<number of cores>
```

<Pi> Port number.

<iface> Interface name of the port.

<Ci>[:<Cj>[...]] A list of fast path cores. Example: 2:3

all All fast path cores. If the fast path has been started with the –nb-rxq=all:N option, the port is configured to be polled by N cores.

nb=<number of cores> Number of fast path cores polling the port. Cores are selected depending on the number of RX queues already polled (i.e. cores with the least number of RX queues are selected).

Example

```
<fp-0> dpdk-port-rxcores-set 0 nb=1
```

See also:

- *Enabling or disabling a port dynamically*
- The *Virtio Host PMD* documentation.

Enabling or disabling a port dynamically

- An enabled port is polled by fast path cores,
- A disabled port is no longer polled and its interface is down.

Important: Before disabling a port, bring its network interface down.

```
dpdk-port-set <Pi>|<iface> enable|disable
```

<Pi> Port number.

<iface> Interface name of the port.

enable or disable Enable or disable the port.

Example

```
<fp-0> dpdk-port-set 0 enable
Port 0 enabled.
```

Displaying the fast path core/port mapping

Display the current polling configuration of the fast path.

```
dpdk-core-port-mapping [<Pi>|<iface>]
```

<Pi> or <iface> Port number. Display all ports if not specified.

Example

```
<fp-0> dpdk-core-port-mapping
port 0: eth1 (rte_em_pmd)
  nb_rxq=1 nb_txq=1
  rxq0=c1
  txq0=c1,c2,c3
port 1: eth2 (rte_em_pmd)
  nb_rxq=1 nb_txq=1
  rxq0=c1
  txq0=c1,c2,c3
```

Displaying the RX/TX port's configuration

Display the configuration of a port. If the TX queue flag is non-zero, the name of the active flag is printed.

```
dpdk-port-rxtx <Pi>|<iface> [default] [raw]
```

<Pi> or <iface> Port number.

default If this token is provided and if the `default` configuration exists, it is displayed instead. The `default` configuration is usually the one applied when the fast path starts, excepted the TX queue flags.

raw If this token is provided, the configuration is displayed as an hexadecimal dump instead.

Example

```
<fp-0> dpdk-port-rxtx 0 raw
current: rx
 0000   00 00 00 00 00 00 00 00  ........
current: tx
 0000   00 00 00 00 00 00 00 00  ........
 0008   00 00 00 00 00 00 00 00  ........
```

Setting the RX/TX port's configuration

Important: After modifying a configuration, disable, then re-enable the interface to update the configuration on the adapter.

```
dpdk-port-rxtx-set <Pi>|<iface>
      [rx_pthresh <u8>] [rx_hthresh <u8>] [rx_wthresh <u8>]
      [rx_free_thresh <u16>] [rx_drop_en <u8>]
      [tx_pthresh <u8>] [tx_hthresh <u8>] [tx_wthresh <u8>]
      [tx_rs_thresh <u8>] [tx_free_thresh <u16>]
      [raw rx|tx <offset u32> <size u32> <val size>]
```

(r|t)x_pthresh <u8> Set the prefetch threshold register of the RX/TX rings. From 0 to 255 included.

(r|t)x_hthresh <u8> Set the host threshold register of the RX/TX rings. From 0 to 255 included.

(r|t)x_wthresh <u8> Set the write-back threshold register of the RX/TX rings. From 0 to 255 included.

(r|t)x_free_thresh <u16> Set the transmit free threshold of the RX/TX rings. From 0 to `rxd` or `txd` included, where `rxd` or `txd` is the number of descriptors in the RX or TX rings.

tx_rs_thresh <u8> Set the transmit RS bit threshold of the TX rings. From 0 to `txd` included, where `txd` is the number of descriptors in the TX rings.

rx_drop_en <u8> Set packet drop for packets with no descriptor. If set to `off`, such a packet will be cached.

raw Write directly to the configuration structure.

rx|tx Set the RX or TX rings configuration.

<offset u32> <size u32> <val size> The value `val` will be written at `offset` on `size` bytes. For example, a `uint8_t` is one byte wide, its corresponding size would be 1. Any `size` greater than 4 will be brought down to 4.

Example

```
<fp-0> dpdk-port-rxtx-set 0 rx_pthresh 1 tx_pthresh 2 rx_drop_en 1
current   RX                          TX
          thresh:                     thresh:
             pthresh:            1       pthresh:            2
             hthresh:            0       hthresh:            0
             wthresh:            0       wthresh:            0
          rx_free_thresh:        0    tx_free_thresh:        0
          rx_drop_en:            1    tx_rs_thresh:          0
rx:
 0000   01 00 00 00 00 00 01 00  ........
tx:
 0000   02 00 00 00 00 00 00 00  ........
 0008   00 00 00 00 00 00 00 00  ........
<fp-0>dpdk-port-rxtx-set 0 raw rx 1 1 5
current   RX                          TX
          thresh:                     thresh:
             pthresh:            1       pthresh:            2
             hthresh:            5       hthresh:            0
             wthresh:            0       wthresh:            0
          rx_free_thresh:        0    tx_free_thresh:        0
          rx_drop_en:            1    tx_rs_thresh:          0
rx:
 0000   01 05 00 00 00 00 01 00  ........
tx:
 0000   02 00 00 00 00 00 00 00  ........
 0008   00 00 00 00 00 00 00 00  ........
```

Configuring the Control Plane Protection

Configure the Control Plane Protection. Default is disabled.

When the machine running the fast path is overloaded, some received packets may be dropped by the hardware. The Control Plane Protection is a software mechanism that anticipates this situation, dropping the data plane packets and keeping control plane packets before the hardware drops them.

It can be also be enabled for TX, dropping in priority the data plane packets if the network link is overloaded.

```
dpdk-cp-filter-set <Pi>|<iface>
   [rx_mode|tx_mode none|software-filter|hardware-filter|dedicated-queue]
   [rx_threshold|tx_threshold <threshold>[%
```

<Pi> or <iface> Port number or interface name.

enable|disable For rx, tx: Enable or disable the Control Plane Protection with default thresholds.

none|software-filter|hardware-filter|dedicated-queue For tx.rx_mode, choose the Control Plane Protection. Default is none.

<thres> Control Plane Protection threshold: correspond to a number or a percentage of descriptors in the hardware Rx/Tx ring.

The thresholds control the data plane drop policy. When the number of packets in a Rx/Tx ring is over the threshold, the packets are analyzed and the data plane packets are dropped.

For standard uses, the default thresholds value should be used. Some drivers limitations prevent to use too high thresholds. Setting the thresholds manually requires to understand these driver limitations.

Example

```
<fp-0> dpdk-cp-filter-set 0 rx_mode none
rx cp filter: rx_mode=none, rxd_count=512
tx cp filter: tx_mode=none, txd_count=512

<fp-0> dpdk-cp-filter-set 0 rx_mode software-filter
rx cp filter: rx_mode=software-filter, rxd_thres=256, rxd_count=512
tx cp filter: tx_mode=none, txd_count=512

<fp-0> dpdk-cp-filter-set 0 tx_mode software-filter tx_threshold 78%
rx cp filter: rx_mode=software-filter, rxd_thres=256, rxd_count=512
tx cp filter: tx_mode=software-filter, txd_thres=399, txd_count=512
```

Set the default Control Plane Configuration for a device type and reconfigure all the port of this type.

It can also used to configure Control plane protection for exception path, to drop in priority the data plane packets if exception path is overloaded.


```
dpdk-cp-filter-devtype-set all|phys|excp|virt
[rx_mode|tx_mode none|software-filter|hardware-filter|dedicated-queue]
[rx_threshold|tx_threshold <threshold>[%]
```

none|software-filter|hardware-filter|dedicated-queue For mode, choose the Control Plane Protection mode.

<threshold> Control Plane Protection threshold: correspond to a number or a percentage of descriptors of the Rx/Tx ring.

The thresholds control the data plane drop policy. When the number of packets in a Rx/Tx ring is over the threshold, the packets are analyzed and the data plane packets are dropped.

Example

```
<fp-0> dpdk-cp-filter-devtype-set excp tx_mode none
cp filter excp device: rx_mode=none, tx_mode=none

<fp-0> dpdk-cp-filter-devtype-set excp tx_mode software-filter
cp filter excp device: rx_mode=none, tx_mode=software-filter, txd_thres=50%

<fp-0> dpdk-cp-filter-devtype-set excp tx_threshold 75%
cp filter excp device: rx_mode=none, tx_mode=software-filter, txd_thres=75%
```

Configure the Control Plane Protection CPU budget.

Set the CPU usage limit that must not be exceeded for the parsing of packets which is part of Control Plane Protection. This limit is per core and applies to Rx and Tx.

If the limit is exceeded on a core, only a fraction of the packets are analyzed and filtered. Therefore, control plane packets can be dropped in this situation.

```
dpdk-cp-filter-set <cpu_budget>
```

<cpu_budget> The maximum cpu budget allocated to the Control Plane Protection.

Example

```
<fp-0> dpdk-cp-filter-budget-set 5
cpu budget is 5%
```


Displaying the Control Plane Protection status

For both Rx and Tx, show the mode of Control Plane Protection (none or software-filter or hardware-filter or dedicated-queue), the value of thresholds, and the number of descriptors for a given port.

```
dpdk-cp-filter <Pi>|<iface>
```

<Pi> or <iface> Port number or interface name.

Example

```
<fp-0> dpdk-cp-filter 0
rx cp filter: rx_mode=software-filter, rxd_thres=256, rxd_count=512
tx cp filter: tx_mode=software-filter, txd_thres=256, txd_count=512
```

Statistics can be displayed using `ethtool -S ifname` and `fp-shmem-dpvi`. Refer to *FPN-SDK Baseline* for details.

Show the default Control Plane Protection mode in Rx and Tx used for a device type (i.e. phys, excp, virt), the value of thresholds.

```
dpdk-cp-filter-devtype
```

Example

```
<fp-0> dpdk-cp-filter-devtype all
cp filter phys device: rx_mode=hardware-filter, rxd_thres=50%, tx_mode=none
cp filter excp device: rx_mode=none, tx_mode=software-filter, txd_thres=50%
cp filter virt device: rx_mode=software-filter, rxd_thres=50%, tx_mode=none
```

Show the CPU budget limit allowed for Control Plane Protection.

```
dpdk-cp-filter-budget
```

Example

```
<fp-0> dpdk-cp-filter-budget
cpu budget is 10%
```

Configuring tx soft queue size of a port

For some PMD it is not possible to configure the number of tx descriptors. For these PMD it can be useful to provide an additional soft queue to allow configuration of the number of tx descriptors

Configure size of a tx soft queue for a specified port

```
dpdk-softqueue-set <Pi>|<iface> <queue_size>
```

<Pi> or <iface> Port number or interface name.

<queue_size> Size of the additional tx soft queue.

Configure default size of tx soft queue applied for any new or reconfigured port. By default the tx soft queue size for new ports is 0.

```
dpdk-softqueue-default-set <queue_size>
```

<queue_size> Size of the tx soft queue applied for any new or reconfigured port.

Statistics can be displayed using `ethtool -S ifname` Refer to *FPN-SDK Baseline* for details.

Configuring Queue Threshold Statistics

Configure the Queue Threshold Statistics. Default is disabled.

When the fast path takes too long to process some packets or is interrupted by another task (usually kernel threads), the other incoming packets are queued in the hardware. Similarly, when a packet is transmitted on a port whose link is overloaded, the packets are queued in the Tx queue. This increases the latency and the risk of dropping packets.

To monitor this, a per-queue statistic counter can be enabled when the filling level of a queue is above a threshold when it is accessed.

```
dpdk-qthres-stats-set <Pi>|<iface> rx|tx enable|disable|<thres>
```

<Pi> or <iface> Port number or interface name.

enable|disable Enable or disable the Queue Threshold Statistics with default thresholds.

<thres> Queue Threshold Statistics threshold: correspond to a number of descriptors in the hardware Rx/Tx ring.

For standard uses, the default thresholds value should be used. Some drivers limitations prevent to use too high thresholds. Setting the thresholds manually requires to understand these driver limitations.

Example

```
<fp-0> dpdk-qthres-stats-set 0 rx disable
rx qthres stats is disabled: rxd_count=512
tx qthres stats is disabled: txd_count=512

<fp-0> dpdk-qthres-stats-set 0 rx enable
rx qthres stats is enabled: rxd_thres=256, rxd_count=512
tx qthres stats is disabled: txd_count=512

<fp-0> dpdk-qthres-stats-set 0 tx 400
rx qthres stats is enabled: rxd_thres=256, rxd_count=512
tx qthres stats is enabled: txd_thres=400, txd_count=512
```

Displaying the Queue Threshold Statistics status

For both Rx and Tx, show the status of Queue Threshold Statistics (enabled or disabled), the value of thresholds, and the number of descriptors for a given port.

```
dpdk-qthres-stats <Pi>|<iface>
```

<Pi> or <iface> Port number or interface name.

Example

```
<fp-0> dpdk-qthres-stats 0
rx qthres stats is enabled: rxd_thres=256, rxd_count=512
tx qthres stats is enabled: txd_thres=256, txd_count=512
```

Statistics can be displayed using `ethtool -S ifname`. Refer to *FPN-SDK Baseline* for details.

Configuring The Link Flow Control

The reception and the transmission of link flow control XOFF/XON pause frames is disabled by default on the port controlled by the fast path. The command `dpdk-link-flow-ctrl-set` enables configuring dynamically the link flow control of a port.

```
dpdk-link-flow-ctrl-set <Pi>|<iface> [async] [autoneg on|off] [rx on|off]
                        [tx (on [highth <h> lowth <l> pause <t>] [send-xon])|off]
```

<Pi> or <iface> Port number or interface name.

async Send message to fast path control thread instead of executing synchronously.

autoneg on|off Enable/disable auto-negotiation of pause frames.

rx on|off Enable/disable reception of pause frames.

tx on|off Enable/disable transmission of pause frames.

highth <h> The amount `h` of allocated bytes in the NIC memory that triggers the transmission of a XOFF frame. This option is available only if `tx` is `on`.

lowth <l> The amount `l` of allocated bytes in the NIC memory that triggers the transmission of a XON frame. This option is available only if `tx` is `on`.

pause <t> The number `t` of 512 bit time for the pause duration, that is also used to determine the refresh period for the re-transmission of the pause frame. This option is available only if `tx` is `on`.

send-xon Enable the transmission of XON frames. This option is available only if `tx` is `on`.

Example

```
<fp-0> dpdk-link-flow-ctrl-set 3 rx on tx off
<fp-0> dpdk-link-flow-ctrl-set 4 rx on tx on highth 256 lowth 128 pause 1664
```

Displaying The Link Flow Control status

The link flow control state can be displayed with the command `dpdk-link-flow-ctrl`:

```
dpdk-link-flow-ctrl <Pi>|<iface>
```

<Pi> or <iface> Port number or interface name.

Example

```
<fp-0> dpdk-link-flow-ctrl 4
Send XON frame: off
Auto-negotiation: off
Receiving MAC control frames: off
Mode:
   RX pause frame: on
   TX pause frame: on
High threshold value to trigger XOFF: 256
Low threshold value to trigger XON: 128
Pause quota in the Pause frame: 1664
```

Restarting Auto-negotiation

The auto-negociation can be restarted with the `dpdk-port-reset` command:

```
dpdk-port-reset <Pi>|<iface>
```

<Pi> or <iface> Port number or interface name.

Example

```
<fp-0> dpdk-port-reset eth0
```

Displaying The Eeprom

The `dpdk-port-modeeprom` command enables displaying the entire Eeprom or a part:

```
dpdk-port-modeeprom <Pi>|<iface> [offset <offset>] [length <len>]
```

<Pi> or <iface> Port number or interface name.

offset <offset> Offset from which to read the Eeprom.

length <len> Number of bytes to read.

Example

```
<fp-0> dpdk-port-modeeprom eth0 length 128
Offset          Values
------          ------
0x0000:         03 04 07 10 00 00 01 00 00 00 00 06 67 02 00 00
0x0010:         08 03 00 1e 49 6e 74 65 6c 20 43 6f 72 70 20 20
0x0020:         20 20 20 20 00 00 1b 21 46 54 4c 58 38 35 37 31
0x0030:         44 33 42 43 56 2d 49 54 41 20 20 20 03 52 00 88
0x0040:         00 3a 00 00 41 47 38 30 35 53 47 20 20 20 20 20
0x0050:         20 20 20 20 30 39 30 38 32 36 20 20 68 fa 03 f7
0x0060:         00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0070:         00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Filling

The filling of fast path tables can be displayed. It takes each VRF into account.

```
filling
```

Examples

```
<fp-0> filling
interfaces: 6/1344
neigh: 4/5000
routes: 9/50000
vrf: 1/16
lpm-tables: 128/256
lpm-mem: 232448/8388608
pbr-rules: 5/1024
neigh6: 3/5000
routes6: 5/50000
ipsec-sa: 0/8192
ipsec-sp: 0/8192
svti-ifaces: 0/128
vxlan-port: 0/15
vxlan-ifaces: 0/127
vxlan-fdb: 0/5000
vlan-ifaces: 0/127
macvlan-ifaces: 0/127
bridge-fdb: 0/5000
bridge-ifaces: 0/127
bridge-ports: 0/511
lag-ifaces: 0/32
lag-slave-ifaces: 0/65
gre-ifaces: 0/255
ebtables-rules: 32/3072
nfct: 5/1024
nfrules: 94/3072
nf-ipsets: 0/64
nf6ct: 0/1024
nf6rules: 74/2048
fp-vswitch-ports: 0/256
fp-vswitch-flows: 0/65536
fp-vswitch-masks: 0/32768
```


Statistics

Displaying all statistics

Display all non null statistics.

```
stats [percore|agg[regated]] [all]
```

percore Display all statistics per core running the fast path.

agg[regated] Display sum of Linux and fast path statistics.

all Display all statistics (even those that are null).

Examples

```
<fp-0> stats all
==== interface stats:
eth1-vr0 port:0
  ifs_ipackets:0
  ifs_ierrors:0
  ifs_ilasterror:0
  ifs_ibytes:0
  ifs_imcasts:0
  ifs_opackets:0
  ifs_oerrors:0
  ifs_obytes:0
eth3-vr0 port:2
  ifs_ipackets:0
  ifs_ierrors:0
  ifs_ilasterror:0
  ifs_ibytes:0
  ifs_imcasts:0
  ifs_opackets:0
  ifs_oerrors:0
  ifs_obytes:0
==== IPv4 stats:
  IpForwDatagrams:682
  IpInReceives:682
  IpInDelivers:447
  IpInHdrErrors:0
  IpInTruncatedPkts:0
  IpInAddrErrors:0
  IpDroppedNoArp:0
  IpDroppedNoMemory:0
```

(continues on next page)

(continued from previous page)

```
  IpDroppedForwarding:0
  IPDroppedOutOperative:0
  IpDroppedIPsec:0
  IpDroppedBlackhole:0
  IpDroppedInvalidInterface:0
  IpDroppedNetfilter:0
  IpDroppedRouteException:0
  IpReasmTimeout:0
  IpReasmReqds:0
  IpReasmOKs:0
  IpReasmFails:0
  IpReasmExceptions:0
  IpFragOKs:0
  IpFragFails:0
  IpFragCreates:0
  IpCsumErrors:0
  IpNhrpPacket:0
  IpNhrpErrorSend:0
==== arp stats:
  arp_errors:0
  arp_unhandled:0
  arp_not_found:0
  arp_replied:0
==== global stats:
  fp_dropped:0
  fp_dropped_excp:0
  fp_dropped_excloop:0
  fp_dropped_ether:0
  fp_dropped_vlan:0
  fp_dropped_macvlan:0
  fp_dropped_bridge:0
  fp_dropped_ebtables:0
  fp_dropped_bonding:0
  fp_dropped_arp:0
  fp_dropped_mcast:0
  fp_dropped_mcast6:0
  fp_dropped_ip:0
  fp_dropped_ipv6:0
  fp_dropped_reasm:0
  fp_dropped_reasm6:0
  fp_dropped_netfilter:0
  fp_dropped_netfilter6:0
  fp_dropped_tunnel:0
  fp_dropped_gre:0
```

(continues on next page)

(continued from previous page)

```
  fp_dropped_vxlan:0
  fp_dropped_ipsec:0
  fp_dropped_ipsec6:0
  fp_dropped_vnb:0
  fp_dropped_ovs:0
  fp_dropped_system:0
  fp_dropped_plugins:0
==== bridge stats:
  L2ForwFrames:0
  BridgeDroppedInvalidOutPort:0
  BridgeDroppedUnknownIface:0
  BridgeDroppedInputLookupError:0
  BridgeDroppedOutputLookupError:0
  BridgeDroppedFwdInvalid:0
  BridgeDroppedOutputUnknown:0
  BridgeDroppedMtuExceeded:0
  BridgeDroppedNoOutputPort:0
  BridgeDroppedLearning:0
  BridgeDroppedInvalidSrc:0
  BridgeDroppedPauseFrame:0
  BridgeDroppedInvalidState:0
  BridgeDroppedOutOperative:0
==== vlan stats:
  VlanDroppedInvalidTag:0
  VlanUnknownTag:0
  VlanDroppedInputUnknownIf:0
  VlanOutputUnknownIf:0
  VlanDroppedPrependFailure:0
  VlanDroppedInOperative:0
  VlanDroppedOutOperative:0
==== ebtables stats:
  L2FilterDroppedVerdict:0
  L2FilterDroppedHeaderTooShort:0
  L2FilterDroppedPrependFailure:0
  L2FilterDroppedIpInvalid:0
  L2FilterDroppedIpv6Invalid:0
  L2FilterDroppedFragFailure:0
==== exception stats:
  LocalBasicExceptions:506
  LocalFPTunExceptions:0
  ExceptionByModule:
    fp_exception_netfpc:57
    fp_exception_ether:0
    fp_exception_ifnet:0
```

(continues on next page)

(continued from previous page)

```
    fp_exception_unknown_ifnet:0
    fp_exception_syslog:0
    fp_exception_reass:0
    fp_exception_tap:0
    fp_exception_tunnel:75
    fp_exception_mcast:0
    fp_exception_mcast6:0
    fp_exception_bonding:0
    fp_exception_bridge:25
    fp_exception_ebtables:0
    fp_exception_netfilter:25
    fp_exception_netfilter6:10
    fp_exception_npf:0
    fp_exception_gre:112
    fp_exception_ip:94
    fp_exception_ipv6:65
    fp_exception_ipsec:0
    fp_exception_ipsec6:0
    fp_exception_macvlan:0
    fp_exception_mpls:0
    fp_exception_packet_steer:0
    fp_exception_vlan:0
    fp_exception_vnb:0
    fp_exception_vxlan:43
    fp_exception_ovs:0
  LocalExceptionClass:
    FPTUN_EXC_UNDEF:0
    FPTUN_EXC_SP_FUNC:449
    FPTUN_EXC_ETHER_DST:0
    FPTUN_EXC_IP_DST:0
    FPTUN_EXC_ICMP_NEEDED:0
    FPTUN_EXC_NDISC_NEEDED:0
    FPTUN_EXC_IKE_NEEDED:0
    FPTUN_EXC_FPC:0
    FPTUN_EXC_NF_FUNC:0
    FPTUN_EXC_TAP:0
    FPTUN_EXC_REPLAYWIN:0
    FPTUN_EXC_ECMP_NDISC_NEEDED:0
    FPTUN_EXC_VNB_TO_VNB:0
    FPTUN_EXC_SOCKET:0
    FPTUN_EXC_IP_PMTU:0
  LocalExceptionType:
     FPTUN_BASIC_EXCEPT:449
     FPTUN_IPV4_NATDONE_INPUT_EXCEPT:0
```

(continues on next page)

(continued from previous page)

```
    FPTUN_IPV4_IPSECDONE_OUTPUT_EXCEPT:0
    FPTUN_IPV6_IPSECDONE_OUTPUT_EXCEPT:0
    FPTUN_IPV4_OUTPUT_EXCEPT:0
    FPTUN_IPV6_OUTPUT_EXCEPT:0
    FPTUN_IPV4_IPSECDONE_INPUT_EXCEPT:0
    FPTUN_IPV6_IPSECDONE_INPUT_EXCEPT:0
    FPTUN_ETH_INPUT_EXCEPT:0
    FPTUN_ETH_NORXOPS_INPUT_EXCEPT:0
    FPTUN_IFACE_INPUT_EXCEPT:0
    FPTUN_OUTPUT_EXCEPT:0
    FPTUN_ETH_SP_OUTPUT_REQ:0
    FPTUN_IPSEC_SP_OUTPUT_REQ:0
    FPTUN_TAP:0
    FPTUN_RFPS_UPDATE:0
    FPTUN_VNB2VNB_FP_TO_LINUX_EXCEPT:0
    FPTUN_VNB2VNB_LINUX_TO_FP_EXCEPT:0
    FPTUN_TRAFFIC_GEN_MSG:0
  ExcpDroppedFpToLinuxFptunFailure:0
  ExcpDroppedFpToLinuxRestoreFailure:0
  ExcpDroppedFpToLinuxEthFptunPrependFailure:0
  ExcpDroppedFpToLinuxEcmpPrependFailure:0
  ExcpDroppedFpToLinuxEcmp6PrependFailure:0
  ExcpDroppedFpToLinuxIPsecPrependFailure:0
  ExcpDroppedFpToLinuxEthPrependFailure:0
  ExcpDroppedFpToLinuxNoIPv4RouteLocal:0
  ExcpDroppedFpToLinuxNoIPv6RouteLocal:0
  ExcpDroppedFpToLinuxAddMarkFailure:0
  ExcpDroppedInvalidMtag:0
  ExcpDroppedLinuxToFpOtherHost:0
  ExcpDroppedLinuxToFpMsgTooShort:0
  ExcpDroppedLinuxToFpInvalidVersion:0
  ExcpDroppedLinuxToFpUnknownIfUid:0
  ExcpDroppedLinuxToFpNoOutputFunction:0
  ExcpDroppedLinuxToFpIPv4PullupFailure:0
  ExcpDroppedLinuxToFpIPv6PullupFailure:0
  ExcpDroppedLinuxToFpUnknownCommand:0
  ExcpDroppedLinuxToFpInvalidPortId:0
  ExcpDroppedLinuxToFpTproxyFailure:0
  ExcpDroppedLinuxToFpGenericCommandFailure:0
```

When you invoke `stats` with the `percore` parameter, the value between square brackets ([]) is the cpu id to which the statistics belong.

```
<fp-0> stats percore
```

(continues on next page)

(continued from previous page)

```
==== interface stats:
eth1-vr0 port:0
eth3-vr0 port:2
==== IPv4 stats:
    IpForwDatagrams:
      IpForwDatagrams[1]:682
      Total:682
    IpInReceives:
       IpInReceives[1]:682
       Total:682
    IpInDelivers:
       IpInDelivers[1]:447
       Total:447
==== arp stats:
==== global stats:
==== exception stats:
    LocalBasicExceptions[1]:527
    Total:527
  ExceptionByModule:
    fp_exception_netfpc[1]:78
    Total:57
    fp_exception_tunnel[1]:75
    Total:75
    fp_exception_bridge[1]:25
    Total:25
    fp_exception_netfilter[1]:25
    Total:25
    fp_exception_netfilter6[1]:10
    Total:10
    fp_exception_gre[1]:112
    Total:112
    fp_exception_ip[1]:94
    Total:94
    fp_exception_ipv6{1]:65
    Total:65
    fp_exception_vxlan[1]:43
    Total:43
  LocalExceptionClass:
    FPTUN_EXC_SP_FUNC[1]:449
    Total:449
  LocalExceptionType:
    FPTUN_BASIC_EXCEPT[1]:449
    Total:449
==== IPsec stats:
```


Displaying network interface statistics

```
stats-iface [percore|agg[regated]] [all]
```

percore Display all statistics per core running the fast path.

aggregated Display sum of Linux and fast path statistics.

all Display all statistics (even those that are null).

Example

```
<fp-0> stats-iface all
lo-vr0 port:254
  ifs_ipackets:0
  ifs_ierrors:0
  ifs_ilasterror:0
  ifs_ibytes:0
  ifs_imcasts:0
  ifs_opackets:0
  ifs_oerrors:0
  ifs_obytes:0
```

Displaying network port statistics

```
stats-port [percore] [all]
```

percore Display all statistics per core running the fast path.

all Display all statistics (even those that are null).

Example

```
<fp-0> stats-port all
eth1-vr0 port:0
  ifs_ipackets:0
  ifs_ierrors:0
  ifs_ilasterror:0
  ifs_ibytes:0
  ifs_imcasts:0
  ifs_opackets:0
  ifs_oerrors:0
```

(continues on next page)

(continued from previous page)

```
  ifs_obytes:0
eth3-vr0 port:2
  ifs_ipackets:0
  ifs_ierrors:0
  ifs_ilasterror:0
  ifs_ibytes:0
  ifs_imcasts:0
  ifs_opackets:0
  ifs_oerrors:0
  ifs_obytes:0
```

Displaying global fast path statistics

```
stats-global [percore] [all]
```

percore Display all statistics per core running the fast path.

all Display all statistics (even those that are null).

Example

```
<fp-0> stats-global all
  fp_dropped:0
  fp_dropped_excp:0
  fp_dropped_excloop:0
  fp_dropped_ether:0
  fp_dropped_vlan:0
  fp_dropped_macvlan:0
  fp_dropped_bridge:0
  fp_dropped_ebtables:0
  fp_dropped_bonding:0
  fp_dropped_arp:0
  fp_dropped_mcast:0
  fp_dropped_mcast6:0
  fp_dropped_ip:0
  fp_dropped_ipv6:0
  fp_dropped_reasm:0
  fp_dropped_reasm6:0
  fp_dropped_netfilter:0
  fp_dropped_netfilter6:0
  fp_dropped_tunnel:0
  fp_dropped_gre:0
```

(continues on next page)

(continued from previous page)

```
  fp_dropped_vxlan:0
  fp_dropped_ipsec:0
  fp_dropped_ipsec6:0
  fp_dropped_vnb:0
  fp_dropped_ovs:0
  fp_dropped_system:0
  fp_dropped_plugins:0
```

Cryptographic statistics

Displaying list of available cryptographic libraries

```
crypto-lib
```

List the available cryptographic libraries

Example

```
<fp-0> crypto-lib
  Available crypto libraries:
     multibuffer
     generic
```

Displaying cryptographic statistics

```
stats-crypto [percore] [lib_name]
```

percore Display all statistics per core running the fast path.

lib_name Display only statistics of this cryptographic library.

Example

```
<fp-0> stats-crypto
  Cumulative statistics
  Nb crypto                 :        86
  Nb asymmetric operations  :         0
  Nb random operations      :         0
  Out of space in queue     :         0
```

(continues on next page)

(continued from previous page)

```
  Out of mbufs              :         0
  Internal errors           :         0
  Authentication errors     :         0
  Dst output size errors    :         0
  Nb polls                  :       172
  Dummy polls               :        86
  Timeout flushs            :         0
  Bulk flushs               :        86

  Global statistics
  Nb sessions               :         2
  Out of sessions           :         0

<fp-0> stats-crypto multibuffer percore
  Core 1 statistics
  Nb crypto                 :        86
  Nb asymmetric operations  :         0
  Nb random operations      :         0
  Out of space in queue     :         0
  Out of mbufs              :         0
  Internal errors           :         0
  Authentication errors     :         0
  Dst output size errors    :         0
  Nb polls                  :       172
  Dummy polls               :        86
  Timeout flushs            :         0
  Bulk flushs               :        86

  Global statistics
  Nb sessions               :         2
  Out of sessions           :         0
```

Displaying bridge statistics

Synopsis

```
stats-bridge [percore] [all]
```

percore Display bridge statistics per core.

all Display all statistics (even those that are null).

Example

```
<fp-0> stats-bridge all
  L2ForwFrames:0
  BridgeDroppedUnknownIface:0
  BridgeDroppedInputLookupError:0
  BridgeDroppedOutputLookupError:0
  BridgeDroppedFwdInvalid:0
  BridgeDroppedOutputUnknown:0
  BridgeDroppedMtuExceeded:0
  BridgeDroppedNoOutputPort:0
  BridgeDroppedLearning:0
  BridgeDroppedInvalidSrc:0
  BridgeDroppedPauseFrame:0
  BridgeDroppedInvalidState:0
  BridgeDroppedOutOperative:0
```

Displaying filter-bridge statistics

Synopsis

```
stats-filter-bridge [percore] [all]
```

percore Display filter-bridge statistics per core.

all Display all statistics (even those that are null).

Example

```
<fp-0> stats-ebtables all
  L2FilterDroppedVerdict:0
  L2FilterDroppedHeaderTooShort:0
  L2FilterDroppedPrependFailure:0
  L2FilterDroppedIpInvalid:0
  L2FilterDroppedIpv6Invalid:0
  L2FilterDroppedFragFailure:0
```

Displaying vlan statistics

Synopsis

```
stats-vlan [percore] [all]
```

percore Display vlan statistics per core.

all Display all statistics (even those that are null).

Example

```
<fp-0> stats-vlan all
  VlanDroppedInvalidTag:0
  VlanUnknownTag:0
  VlanDroppedInputUnknownIf:0
  VlanOutputUnknownIf:0
  VlanDroppedPrependFailure:0
  VlanDroppedInOperative:0
  VlanDroppedOutOperative:0
```

Resetting all fast path statistics

```
stats-reset
```

Example

```
<fp-0> stats-reset
```

```
<fp-0> stats
==== interface stats:
eth1-vr0 port:0
eth3-vr0 port:2
==== IPv4 stats:
==== arp stats:
==== global stats:
==== exception stats:
   LocalBasicExceptions:2
   ExceptionByModule:
     fp_exception_netfpc:2
   LocalExceptionClass:
```

(continues on next page)

(continued from previous page)

```
   LocalExceptionType:
==== IPsec stats:
```

Internal debugging

This section lists all commands that can force fast path behavior for debugging purposes.

Managing debug log

Displaying log level

```
log
```

Example

```
<fp-0> log
Log mode is syslog
Log level is 3

log MAIN_PROC is on
log EXC is on
log IP is on
log FRAG is on
log IPSEC_IN is on
log IPSEC_OUT is on
log IPSEC_REPL is on
[ snip ]
```

Setting log level

```
log-level-set LEVEL
```

LEVEL

LEVEL	LOG
0	Emergency: system is unusable
1	Alert: action must be taken immediately
2	Critical level messages
3	Error level messages
4	Warning level messages
5	Notice: normal but significant condition
6	Information level messages
7	Debug level messages

Example

```
<fp-0> log-level-set 2
Log level is 2
```

Setting log mode

```
log-mode-set console|syslog
```

console Display logs directly on the console.

syslog Send logs to syslog.

Example

```
<fp-0> log-mode-set syslog
Log mode is syslog
```

Setting log type

```
log-type-set <type>|all on|off
```

<type> log type (exec 'fp-cli log' command to know the available type list).

all Set all log type.

on or off Enable or disable log of the selected type.

Example

```
<fp-0> log-type-set GRE on
log GRE is on
```

Displaying the fast path configuration flags

```
fp-state
```

Example

```
<fp-0> fp-state
FP is started

IPv4 Netfilter: off
IPv6 Netfilter: off
IPv4 Netfilter cache: off
IPv6 Netfilter cache: off
IPv4 IPsec output: off
IPv6 IPsec output: off
IPv4 IPsec input: off
IPv6 IPsec input: off
Do IPsec only once: off
Bridge filtering: off
Forced reassembly: off
Tap: off (local) (exceptions)
ARP reply: off
NPF packet filter: off
Fast forward: on
Transparent socket mode: off
IPv4 options processing: on
IPv6 options processing: on
```

The line “Fast forward: on” means the stack takes a shortcut to forward IP packets as long as other features are off (typically it skips filter and IPsec rules).

Displaying the list of fast path plugins

Displaying the list of fast path plugins for the fast path, the fast path manager, or fast path CLI modules.

```
module-list [fp|fpm|fpcli|all]
```

fp List fast path loaded modules.

fpm List FPM loaded modules.

fpcli List fp-cli loaded modules.

all List fast path, FPM and fp-cli loaded modules.

Example

```
<fp-0> module-list all
FP loaded modules:
     /usr/lib/fastpath/libfp-vswitch.so
FPM loaded modules:
     /usr/lib/fpm/libfpm-fp-vswitch.so
FPCLI loaded modules:
     /usr/lib/fp-cli/libfpd-vswitch.so
```

Displaying configuration

```
conf runtime|compiled
```

runtime Display runtime arguments set when the fast path is launched (if no value has been set, the default values are displayed).

compiled Display fast path configuration set at compilation time.

Example

```
<fp-0> conf runtime
Runtime values are:
     max-ifnet    :1025
     max-vr       :16
```

Manage Flow Director configuration

Flow Director functionality is available on some network adapters (at the moment, ixgbe and i40e are the most popular adapters implementing it, but others do as well).

Displaying Flow Director configuration

Display the Flow Director configuration for a specific port.

```
dpdk-fdir-config <Pi>
```

<Pi> Port number of the interface.

Example

```
<fp-0> dpdk-fdir-config 0
mode: none
memory: 64k
fdirhash: noreport
dropq: 0
```

Configuring Flow Director

Enable / disable Flow Director and change its parameters for a specific port.

Important: After modifying this configuration, then re-enable the interface to update the configuration on the adapter.

```
dpdk-fdir-config-set <Pi>
         [mode { perfect | signature | none }]
         [memory { 64k | 128k | 256k }]
         [fdirhash { noreport | report | always }]
         [dropq <u8>]
```

<Pi> Port number of the interface.

mode 'perfect' is the only mode supported at the moment. 'none' disables Flow Director.

memory Possible values are '64k', '128k', '256k'. This value has a direct impact on the number of Flow Director filters available.

fdirhash Possible values are 'noreport', 'report', 'always'. Instruct hardware to report the hash value that it computed.

dropqueue If using Flow Director filters with drop flag set, set the queue where packets will be received (and if queue is not enabled, then packets will finally be dropped).

Example

```
<fp-0> dpdk-fdir-config-set 0 mode perfect memory 64k fdirhash noreport dropq 127
```

Manage a flow director mask

The mask is used to determine the part on which the flow director filter will be applied (i.e. for a specific field, bits set to 1 define the relevant bits to compare with the filter).

Displaying a flow director mask

Display the configured mask for a specific port.

```
dpdk-fdir-mask <Pi>
```

<Pi> Port number of the interface.

Example

```
<fp-0> dpdk-fdir-mask 0
mask: vlan_tci: 0x002a
      ipv4 src_ip: 0xffffffff
      ipv4 dst_ip: 0x00000000
      ipv6 src_ip: 0x00000000 0x00000000 0x00000000 0x00000000
      ipv6 dst_ip: 0x00000000 0x00000000 0x00000000 0x00000000
      sport: 0x0000
      dport: 0xffff
```

Setting a flow director mask

Set the configured mask for a specific port.

Important: After modifying a mask, disable, then re-enable the interface to update the configuration on the adapter.

```
dpdk-fdir-mask-set <Pi>
         [vlan_tci           <u16>]
         [src_port           <u16>]
         [dst_port           <u16>]
         [src_ip   <@ipv4>|<@ipv6>]
         [dst_ip   <@ipv4>|<@ipv6>]
```

<Pi> Port number of the interface.

vlan_tci Bits set to 1 define which bits will be matched in the `vlan_tci` field.

dst_ip Bits set to 1 define the relevant bits to use in the destination address of an IP packet. You can provide either a v4, or a v6 address; the relevant field will be set.

src_ip Bits set to 1 define the relevant bits to use in the source address of an IP packet. You can provide either a v4, or a v6 address; the relevant field will be set.

dst_port Bits set to 1 define the relevant bits to use in the destination port of selected LAYER 4 protocol (TCP or UDP).

src_port Bits set to 1 define the relevant bits to use in the source port of selected LAYER 4 protocol (TCP or UDP).

Example

```
<fp-0> dpdk-fdir-mask-set 0 src_ip 255.255.255.255 src_ip ffff::cff dst_port 65535
```

Manage a port's filters

A filter is composed of two main parts: an input flow, and a programmable action. The input flow serves to characterize which flow should be matched, while the action defines what should be done when a flow is matched.

Only the `perfect` filter type has been tested and is currently supported. The `signature` filter type will have a different behavior, and is not covered in this documentation.

Adding a flow director filter

The filter provided will be added to the filter pool of the specified interface. How the interface responds to various situations is up to the PMD. On IXGBE, if two input flows collide, the new programmable action is discarded. Otherwise, if there is enough space, the new filter is added. If a filter is successfully defined (but before it is added by the driver), it is displayed. Make sure that the provided filter is compatible with your interface, however.

```
<fp-0> dpdk-fdir-filter-add <Pi> FILTER
               FILTER := (in any order)
                       FLOW_TYPE [flex_off <u8>] [flexbytes <u8>[ ...]]
```

(continues on next page)

(continued from previous page)

```
                    [src_port <u16>] [dst_port <u16>] [rx_queue <u16>]
                    [vlan_tci <u16>] [verify_tag <u32>] [soft_id <u32>]
                    [REPORT_TYPE] [src_ip <@ipv4>|<@ipv6>]
                    [dst_ip <@ipv4>|<@ipv6>] [ACTION]
```

<Pi> Port number of the interface.

FLOW_TYPE The flow type that will be targeted by the filter. Available flow types are defined by the PMD of your interface. You can either use the full name or its shorthand alias to specify it. If no *flow_type* is provided, it is assumed to be *none*. If several flow types are provided, the last one will overwrite all previous values. The *flow_type* is part of the input flow.

full name	alias
none	N/A
raw	N/A
ipv4	i4
ipv4_frag	f4
ipv4_tcp	t4
ipv4_udp	u4
ipv4_sctp	s4
ipv4_other	o4
ipv6	i6
ipv6_frag	f6
ipv6_udp	u6
ipv6_tcp	t6
ipv6_sctp	s6
ipv6_other	o6
l2_payload	l2pl
ipv6_ex	i6x
ipv6_tcp_ex	t6x
ipv6_udp_ex	u6x

<flex_off> An offset from which the flexbytes will be read for reporting purposes. It is part of the programmable action.

<flexbytes> When using a flex configuration, bytes to be matched within a flow. Although they should be provided as unsigned 8 bits values, a flexbyte matching will only be performed on 16 bits wide words. If *0x* is prepended to a value, it will be read as hexadecimal.

<src_port>, <dst_port> Which source or destination port to match in a flow. It is part of the input flow. It will conflict with any *verify_tag* parameter.

<rx_queue> Define which queue the packet should be sent to. It is part of the programmable action.

<vlan_tci> A VLAN identifier that can be matched.

<verify_tag> A *verify_tag* that can be matched in a SCTP (Stream Control Transmission Protocol) stream. It is

part of the input flow. It will conflict with both source and destination port.

<soft_id> A value that can be used to identify a filter and possibly differentiate how a match is reported.

REPORT_TYPE Which kind of information to report when a packet is matched.

no_report Do not fill the matching packet with any extra information (any counters would still be incremented).

report_id The matching packet will contain the given *soft_id* of the matching filter.

report_flex Set the relevant fields in the packet with the matched flex bytes.

report_id_flex Try to fit both the *id* and the *flex*. However, only part of each will be written due to size constraints. It is part of the programmable action.

<src_ip>, <dst_ip> IPv4 or IPv6 source and destination addresses. The relevant field will be set. It is part of the input flow.

Important: Do not provide mix IPv4 and IPv6 addresses.

ACTION

accept | reject Whether to forward (accept) or drop (reject) the matched packet. It is part of the programmable action.

Examples

```
<fp-0> dpdk-fdir-filter-add 5 o4 rx_queue 3 dst_ip 11.0.1.1
ADD operation on filter:

filter: soft_id: 00000000
        input: flow_type: 0007 (ipv4_other)
               flow: ip4_flow: src_ip: 0.0.0.0
                               dst_ip: 11.0.1.1
               flow_ext: vlan_tci: 0000
                         flexbytes: 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0␣
↪0x0 0x0 0x0 0x0
        action: rx_queue: 0003
                flex_off: 00
                behavior: ACCEPT
                report_status: no_report
```

Setting a flow director filter

Allow to update a filter, which can be targeted by specifying its input flow. On IXGBE, if such a filter already exists, its programmable action is overwritten with the new one. If there is no filter matching the input flow, this command behaves like `dpdk-fdir-filter-add`.

```
<fp-0> dpdk-fdir-filter-set <Pi> FILTER
               FILTER := (in any order)
                       FLOW_TYPE [flex_off <u8>] [flexbytes <u8>[ ...]]
                       [src_port <u16>] [dst_port <u16>] [rx_queue <u16>]
                       [vlan_tci <u16>] [verify_tag <u32>] [soft_id <u32>]
                       [REPORT_TYPE] [src_ip <@ipv4>|<@ipv6>]
                       [dst_ip <@ipv4>|<@ipv6>] [ACTION]
```

<Pi> Port number of the interface.

FLOW_TYPE The flow type that will be targeted by the filter. Available flow types are defined by the PMD of your interface. You can either use the full name or its shorthand alias to specify it. If no *flow_type* is provided, it is assumed to be *none*. If several flow types are provided, the last one will overwrite all previous values. The *flow_type* is part of the input flow.

full name	alias
none	N/A
raw	N/A
ipv4	i4
ipv4_frag	f4
ipv4_tcp	t4
ipv4_udp	u4
ipv4_sctp	s4
ipv4_other	o4
ipv6	i6
ipv6_frag	f6
ipv6_udp	u6
ipv6_tcp	t6
ipv6_sctp	s6
ipv6_other	o6
l2_payload	l2pl
ipv6_ex	i6x
ipv6_tcp_ex	t6x
ipv6_udp_ex	u6x

<flex_off> An offset from which the flexbytes will be read for reporting purposes. It is part of the programmable action.

<flexbytes> When using a flex configuration, bytes to be matched within a flow. Although they should be provided as unsigned 8 bits values, a flexbyte matching will only be performed on 16 bits wide words. If *0x* is prepended

to a value, it will be read as hexadecimal.

<src_port>, <dst_port> Which source or destination port to match in a flow. It is part of the input flow. It will conflict with any *verify_tag* parameter.

<rx_queue> Define which queue the packet should be sent to. It is part of the programmable action.

<vlan_tci> A VLAN identifier that can be matched.

<verify_tag> A *verify_tag* that can be matched in a SCTP stream. It is part of the input flow. It will conflict with both source and destination port.

<soft_id> A value that can be used to identify a filter and possibly differentiate how a match is reported.

REPORT_TYPE Which kind of information to report when a packet is matched.

no_report Do not fill the matching packet with any extra information (any counters would still be incremented).

report_id The matching packet will contain the given *soft_id* of the matching filter.

report_flex Set the relevant fields in the packet with the matched flex bytes.

report_id_flex Try to fit both the *id* and the *flex*. However, only part of each will be written due to size constraints. It is part of the programmable action.

<src_ip>, <dst_ip> IPv4 or IPv6 source and destination addresses. The relevant field will be set. It is part of the input flow.

Important: Do not provide mix IPv4 and IPv6 addresses.

ACTION

accept | reject Whether to forward (accept) or drop (reject) the matched packet. It is part of the programmable action.

Examples

```
<fp-0> dpdk-fdir-filter-set 5 s4 report_id reject
UPDATE operation on filter:

filter: soft_id: 00000000
        input: flow_type: 0006 (ipv4_sctp)
               flow: ip4_flow: src_ip: 0.0.0.0
                               dst_ip: 0.0.0.0
                     sctp4_flow: verify_tag: 00000000
               flow_ext: vlan_tci: 0000
                         flexbytes: 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0␣
↪0x0 0x0 0x0 0x0
```

(continues on next page)

(continued from previous page)

```
        action: rx_queue: 0000
                flex_off: 00
                behavior: REJECT
                report_status: report_id
```

Deleting a flow director filter

Provided the specified input flow is used by a filter, such a filter is deleted from the filter pool. If no filter exists, nothing happens.

```
<fp-0> dpdk-fdir-filter-del <Pi> FILTER
               FILTER := (in any order)
                       FLOW_TYPE [flex_off <u8>] [flexbytes <u8>[ ...]]
                       [src_port <u16>] [dst_port <u16>] [rx_queue <u16>]
                       [vlan_tci <u16>] [verify_tag <u32>] [soft_id <u32>]
                       [REPORT_TYPE] [src_ip <@ipv4>|<@ipv6>]
                       [dst_ip <@ipv4>|<@ipv6>] [ACTION]
```

<Pi> Port number of the interface.

FLOW_TYPE The flow type that will be targeted by the filter. Available flow types are defined by the PMD of your interface. You can either use the full name or its shorthand alias to specify it. If no *flow_type* is provided, it is assumed to be *none*. If several flow types are provided, the last one will overwrite all previous values. The *flow_type* is part of the input flow.

full name	alias
none	N/A
raw	N/A
ipv4	i4
ipv4_frag	f4
ipv4_tcp	t4
ipv4_udp	u4
ipv4_sctp	s4
ipv4_other	o4
ipv6	i6
ipv6_frag	f6
ipv6_udp	u6
ipv6_tcp	t6
ipv6_sctp	s6
ipv6_other	o6
l2_payload	l2pl
ipv6_ex	i6x
ipv6_tcp_ex	t6x
ipv6_udp_ex	u6x

<flex_off> An offset from which the flexbytes will be read for reporting purposes. It is part of the programmable action.

<flexbytes> When using a flex configuration, bytes to be matched within a flow. Although they should be provided as unsigned 8 bits values, a flexbyte matching will only be performed on 16 bits wide words. If *0x* is prepended to a value, it will be read as hexadecimal.

<src_port>, <dst_port> Which source or destination port to match in a flow. It is part of the input flow. It will conflict with any *verify_tag* parameter.

<rx_queue> Define which queue the packet should be sent to. It is part of the programmable action.

<vlan_tci> A VLAN identifier that can be matched.

<verify_tag> A *verify_tag* that can be matched in a SCTP stream. It is part of the input flow. It will conflict with both source and destination port.

<soft_id> A value that can be used to identify a filter and possibly differentiate how a match is reported.

REPORT_TYPE Which kind of information to report when a packet is matched.

no_report Do not fill the matching packet with any extra information (any counters would still be incremented).

report_id The matching packet will contain the given *soft_id* of the matching filter.

report_flex Set the relevant fields in the packet with the matched flex bytes.

report_id_flex Try to fit both the *id* and the *flex*. However, only part of each will be written due to size constraints. It is part of the programmable action.

<src_ip>, <dst_ip> IPv4 or IPv6 source and destination addresses. The relevant field will be set. It is part of the input flow.

Important: Do not provide mix IPv4 and IPv6 addresses.

ACTION

accept | reject Whether to forward (accept) or drop (reject) the matched packet. It is part of the programmable action.

Examples

```
<fp-0> dpdk-fdir-filter-del 5 o4 rx_queue 3 dst_ip 11.0.1.1
DELETE operation on filter:

filter: soft_id: 00000000
        input: flow_type: 0007 (ipv4_other)
               flow: ip4_flow: src_ip: 0.0.0.0
                               dst_ip: 11.0.1.1
               flow_ext: vlan_tci: 0000
                         flexbytes: 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0␣
↪0x0 0x0 0x0 0x0
        action: rx_queue: 0003
                flex_off: 00
                behavior: ACCEPT
                report_status: no_report
```

Flushing a flow director filter

Erase every filter from the specified interface filter pool.

```
<fp-0> dpdk-fdir-filter-flush <Pi>
```

<Pi> Port number of the interface.

Examples

```
<fp-0> dpdk-fdir-filter-flush 5
Flushing filters
```

Manage a flex configuration

You must disable, then re-enable the interface to take your changes into account.

Displaying flex configuration

Display the flex configuration of a given interface. If no other parameter is provided, only display the non-zero sets in decimal notation for the payload offsets, and in hexadecimal notation for the masks.

```
<fp-0> dpdk-fdir-flex <Pi> [<base=10>] [full]
```

<Pi> Port number of the interface.

<base> Base in which the values are displayed.

full Display every field.

Examples

```
<fp-0> dpdk-fdir-flex 1
      payload   0 raw [58 59 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
      flexmask  0 none [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Setting flex payload

Command to write a flex-set. A payload set is a set of 16 offsets that will be matched against a flexbyte series provided in a filter.

```
<fp-0> dpdk-fdir-flex-set <Pi> <pid>|all PAYLOAD <u16>[ <u16>[ ...]]
```

<Pi> Port number of the interface.

<pid> | all Payload ID. Defines which flex-set to write to (from 0 to 7 included). Use `all` to set every payload at the same time.

PAYLOAD The payload type.

l2 The flexible payload is started from the beginning of the data link layer (unsupported by ixgbe driver).

l3 The flexible payload is started from the beginning of the network layer (unsupported by ixgbe driver).

l4 The flexible payload is started from the beginning of the transport layer (unsupported by ixgbe driver).

raw The flexible payload is started from the beginning of the packet.

<u16>[<u16>[...]] The payload offset (zero if not set).

Examples

```
<fp-0> dpdk-fdir-flex-set 1 all raw 0xff 0xff
<fp-0> dpdk-fdir-flex 1 16
      payload   0 raw [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
      payload   1 raw [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
      payload   2 raw [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
      payload   3 raw [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
      payload   4 raw [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
      payload   5 raw [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
      payload   6 raw [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
      payload   7 raw [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Setting flex mask

Define a flex-mask. The first bit marks the bits to be matched from the offsets defined by the flex-sets.

```
<fp-0> dpdk-fdir-flex-mask-set <Pi> <fid>|all FLOW_TYPE <u8>[ <u8>[ ...]]
```

<Pi> Port number of the interface.

<fid> | all Flow ID. Which flow to match (from 0 to 17 included). Some PMDs will infer the flow type from the `fid`, while others will allow to use a specific flow type on a specific `fid`. Use `all` to set every mask at the same time.

FLOW_TYPE The flow type that will be targeted by the filter. Available flow types are defined by the PMD of your interface. You can either use the full name or its shorthand alias to specify it. If no *flow_type* is provided, it is assumed to be *none*. If several flow types are provided, the last one will overwrite all previous values. The *flow_type* is part of the input flow.

full name	alias
none	N/A
raw	N/A
ipv4	i4
ipv4_frag	f4
ipv4_tcp	t4
ipv4_udp	u4
ipv4_sctp	s4
ipv4_other	o4
ipv6	i6
ipv6_frag	f6
ipv6_udp	u6
ipv6_tcp	t6
ipv6_sctp	s6
ipv6_other	o6
l2_payload	l2pl
ipv6_ex	i6x
ipv6_tcp_ex	t6x
ipv6_udp_ex	u6x

<u8>[<u8>[...]] The mask offset (zero if not set).

Examples

```
<fp-0> dpdk-fdir-flex-mask-set 1 0 raw 0xff 0xff
<fp-0> dpdk-fdir-flex 1
      flexmask  0 raw [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Copying a flex configuration

Copy a flex configuration from another interface. The flex configuration of the <Pi> interface will be a copy of the *src Pi* flex configuration.

```
<fp-0> dpdk-fdir-flex-copy <Pi> <src Pi>
```

<Pi> Port number of the destination interface.

<src Pi> Port number of the source interface.

Examples

```
<fp-0> dpdk-fdir-flex-set 1 0 raw 58 59
<fp-0> dpdk-fdir-flex-mask-set 1 0 none 0xff 0xff
<fp-0> dpdk-fdir-flex-copy 2 1
<fp-0> dpdk-fdir-flex 1
      payload   0 raw [58 59 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
      flexmask  0 none [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
<fp-0> dpdk-fdir-flex 2
      payload   0 raw [58 59 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
      flexmask  0 none [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Flushing flex payload

Flush a flex configuration. If no specific ID is provided, every payload field will be flushed. If a specific ID is provided, only this entry will be flushed.

```
<fp-0> dpdk-fdir-flex-flush <Pi> [<pid>]
```

<Pi> Port number of the interface.

<pid> Payload ID. Defines which flex-set to flush to (from 0 to 7 included).

Examples

```
<fp-0> dpdk-fdir-flex-set 1 0 raw 58 59
<fp-0> dpdk-fdir-flex-mask-set 1 0 none 0xff 0xff
<fp-0> dpdk-fdir-flex 1
      payload   0 raw [58 59 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
      flexmask  0 none [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
<fp-0> dpdk-fdir-flex-flush 1
<fp-0> dpdk-fdir-flex 1
      flexmask  0 none [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Flushing flex mask

Flush a flex configuration. If no specific ID is provided, every mask field will be flushed. If a specific ID is provided, only this mask will be flushed.

```
<fp-0> dpdk-fdir-flex-mask-flush <Pi> [<fid>]
```

<Pi> Port number of the interface.

<fid> Flow ID. Which flow to flush (from 0 to 17 included).

Examples

```
<fp-0> dpdk-fdir-flex-set 1 0 raw 58 59
<fp-0> dpdk-fdir-flex-mask-set 1 0 none 0xff 0xff
<fp-0> dpdk-fdir-flex 1
      payload   0 raw [58 59 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
      flexmask  0 none [ff ff 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
<fp-0> dpdk-fdir-flex-mask-flush 1
<fp-0> dpdk-fdir-flex 1
      payload   0 raw [58 59 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Fast Path Statistics

fast path statistics transparently provides to applications a view of network statistics which takes fast path statistics in account.

Fast Path Statistics library

The fast path statistics library can be preloaded (using `LD_PRELOAD`) when starting an application that accesses to network statistics. It hooks some *glibc* functions to inspect and to modify network statistics retrieved from the operating system.

It includes:

- Interface statistics retrieved through *netlink* (*RTM_NEWLINK* and *RTM_NEWSTATS* messages)
- Interface statistics retrieved through `/proc/net/dev`
- Interface statistics retrieved through `/sys/device/.../net/.../statistics`
- IP statistics retrieved through `/proc/net/snmp`
- IPv6 statistics retrieved through `/proc/net/snmp6`

Usage

An application can be started with the fast path statistics library preloaded using the `fps` helper script.

Example

```
# cat /proc/net/dev
<linux-only statistics>

# fps cat /proc/net/dev
<fast path aware statistics>
```

--log-level=<level>
: Set the log level. Valid range is from 0 (no log) to 7 (debug). Default is 4 (warning). The LIBFPS_LOG_LEVEL environment variable can be used for the same purpose.

--use-syslog=0|1
: Set this option to 1 to redirect fast path statistics library logs to *syslog*. By default (0), logs are issued on *stderr*. The LIBFPS_USE_SYSLOG can be used for the same purpose.

Limitations

The fast path statistics library retrieves the fast path statistics from the fast path shared memories. If the fast path is updated with a new shared memory format, the applications using the fast path statistics library should be restarted.

Fast Path Longest Prefix Match Library

IPv4 and IPv6 routers use an LPM (Longest Prefix Match) algorithm to determine where to forward the packet to.

The fast path LPM library stores IPv4 and IPv6 routes in a tree, and provides an API to add and remove routes, and a fast lookup API to be used from data path.

Features

- IPv4 and IPv6 support
- Lookup from destination (LPM)
- Exact lookup (prefix, prefix length and metric must match)
- Add and delete routes
- Flush tree
- Iterate route tree

2.3.2 Fast Path Filtering Ethernet Bridge

Overview

Fast Path Filtering Ethernet Bridge provides bridge (layer 2) filtering in the fast path.

Rules defined via the Linux tool `ebtables` are automatically applied in the fast path.

See also:

For more information, see the `ebtables` manual pages.

Features

- `filter` and `broute` tables
- BROUTING, INPUT, FORWARD and OUTPUT hooks
- User-defined chains
- Standard targets: ACCEPT, DROP, CONTINUE, RETURN
- Matches (with or without '!' flag):
  - Input/output (logical or not) interface
  - MAC source/destination
  - IP(v4/v6) source/destination address
  - IP(v4/v6) source/destination port
  - IPv4 type of service and protocol
  - IPv6 traffic class and protocol

Dependencies

Linux

- Kernel compiled with `Netfilter` and `ebtables` support

6WINDGate modules

- *Fast Path Baseline*
- *Fast Path Ethernet Bridge*

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

Managing `ebtables` rules

Use the `ebtables` Linux program to define bridge filtering rules.

Note: Only a subset of targets and matches is supported.

See also:

For more information on the `ebtables` syntax, see the `ebtables` manual.

Viewing `ebtables` rules from the fast path

The `fp-cli` command below allow you to view current `ebtables` rules.

1. To start `fp-cli`, enter:

```
$ fp-cli
```

Displaying `ebtables` rules

Synopsis

```
filter-bridge [broute|filter [all]]
```

filter

Display `filter` table rules.

broute

Display `broute` table rules.

all

Display all rules instead of a simple summary.

Example

```
<fp-0> filter-bridge filter all
filter bridge is on

EBTable: filter
              pre   in  fwd  out post  brt
Valid hooks:         x    x    x
Hooks:          0    0    2    4    0    0
Underflows:     0    1    3    4    0    0

#   0:   -p IPv6 --ip6-dst 3ffe:0002:0010:0000:0000:0000:0000:0001/
↪ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
  Target: STANDARD, verdict: FP_EBT_ACCEPT
#   1:   policy (INPUT)
  Target: STANDARD, verdict: FP_EBT_ACCEPT
#   2:   -s 03:03:03:03:03:03/ff:ff:ff:ff:ff:ff
  Target: STANDARD, verdict: 6
#   3:   policy (FORWARD)
  Target: STANDARD, verdict: FP_EBT_ACCEPT
#   4:   policy (OUTPUT)
  Target: STANDARD, verdict: FP_EBT_ACCEPT
#   5:   ERROR
#   6:   -s 04:04:04:04:04:04/ff:ff:ff:ff:ff:ff
  Target: STANDARD, verdict: FP_EBT_ACCEPT
#   7:   policy (user-defined-chain)
  Target: STANDARD, verdict: FP_EBT_ACCEPT
#   8:   ERROR
```

Enabling or disabling `ebtables`

Synopsis

```
filter-bridge-set on|off
```

Example

```
<fp-0> filter-bridge-set on
filter-bridge is on
```

ebtables cache status

Synopsis

```
ebt-cache
```

Example

```
<fp-0> ebt-cache
filter bridge cache is on
filter bridge drop cache is on
```

Enabling or disabling ebtables cache

Synopsis

```
ebt-cache-set on|off
```

Example

```
<fp-0> ebt-cache-set on
ebt-cache is on
```

Enabling or disabling ebtables cache for drop flows

Synopsis

```
ebt-cache-drop-set on|off
```

Example

```
<fp-0> ebt-cache-drop-set on
ebt-cache-drop is on
```

Providing options

--max-rules
Maximum number of bridge filter rules

Default value 3072

Memory footprint per IPv4 Netfilter rule 35 KB

Range 0 .. 40K

Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

---

2.3.3 Fast Path Ethernet Bridge

Overview

Fast Path Ethernet Bridge provides Ethernet bridging in the fast path.

Features

Hairpin Controls whether or not traffic may be sent back from the port on which it was received.

Learning Controls whether or not a given port will learn MAC (Medium Access Control) addresses from received traffic.

Flooding Controls whether or not a given port will flood unicast traffic for which there is no FDB entry.

Dependencies

6WINDGate modules

- *Fast Path Baseline*

Linux

- Bridge synchronization is a kernel patch (upstream 3.0).

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=bb900b27a2f4

- Promiscuity status synchronization (for bridge ports) is a kernel patch (upstream 3.13). Without this patch, user has to manually set the bridge ports in promiscuous mode.

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=991fb3f74c14

- Bridge FDB synchonization is a kernel patch (upstream 3.0).

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=b078f0df6762

- Synchronization of permanent FDB entries is a kernel patch (upstream 3.3).

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=f58ee4e1a28b

Usage

A bridge interface is a virtual interface that can be created to perform data link layer forwarding on physical ethernet ports connected to multiple network segments. The ethernet ports become slaves to the bridge interface.

You can manage bridges from Linux or from the fast path.

Managing bridges from Linux

The following Linux commands allow to manage bridge devices (assuming *Linux - Fast Path Synchronization* is started).

Loading required modules

1. Load required modules:

```
# modprobe bridge
```

Creating a bridge device

```
# brctl addbr br0
```

This creates a new device (`br0`).

Displaying information about a bridge device

```
# brctl show br0
```

Deleting a bridge device

```
# brctl delbr br0
```

Adding a port to the bridge

```
# brctl addif br0 eth0
```

Forcing promiscuity in the fast path

This feature is meant for Linux < 3.13.

```
# ip link set eth0 promisc on
```

Deleting a port from a bridge

```
# brctl delif br0 eth0
```

Managing bridges from the fast path

The `fp-cli` commands below allow you to manage Fast Path Ethernet Bridge instances.

Setting port features from a bridge

Synopsis

```
bridge-port-set IFNAME [hairpin|learning|flooding <on|off>]
```

IFNAME Name of the port interface.

Example

```
<fp-0> bridge-port-set eth0 hairpin on
eth0-vr0: master br0-vr0
        state: forwarding
        features: hairpin learning flooding
```

Displaying bridge interfaces

Synopsis

```
bridge
```

Example

```
<fp-0> bridge
Bridge interfaces:
br0-vr0:
        eth0-vr0: master br0-vr0
                state: forwarding
                features: learning flooding
        eth1-vr0: master br0-vr0
                state: forwarding
                features: learning flooding
```

Displaying a bridge interface's FDB entry

Synopsis

```
bridge-fdb IFNAME
```

IFNAME Name of the bridge interface.

Example

```
<fp-0> bridge-fdb br0
br0-vr0:
 00:30:1b:b4:dc:87 eth0-vr0 local
 00:50:fc:4c:88:e4 eth1-vr0
 00:50:fc:62:ec:c6 eth1-vr0 local
 00:30:1b:b4:df:94 eth0-vr0
```

Configuring the bridge FDB hitflags refresh policy

Synopsis

```
bridge-fdb-hitflags-set PERIOD MAX_SCANNED MAX_SENT
```

PERIOD Period in seconds of FDB checking.

MAX_SCANNED Maximum number of FDB entries to scan on a given period of time.

MAX_SENT Maximum number of FDB entries to send over a given period of time.

Example

```
<fp-0> bridge-fdb-hitflags-set 30 2500 1600
```

Displaying bridge FDB hitflags parameters

Synopsis

```
hitflags [all|bridge-fdb|...]
```

No parameter Display parameters for all categories.

all Same as with no parameters.

bridge-fdb Display bridge FDB hitlfags parameters.

Example

```
<fp-0> hitflags bridge-fdb
Bridge FDB hitflags
  period_in_seconds: 30
  max_scanned: 2500
  max_sent: 1600
```

Providing options

Some capabilities can be tuned for this module.

--iface-max
: Maximum number of bridge interfaces

  Default value 127

  Memory footprint per bridge interface 32 B

Example

```
FP_OPTIONS="--mod-opt=bridge:--iface-max=16"
```

Then fast path can manage up to 16 bridge interfaces.

--port-max
: Maximum number of bridge ports

  Default value 511

  Memory footprint per bridge ports 36 B

Example

```
FP_OPTIONS="--mod-opt=bridge:--port-max=500"
```

--fdb-max
: Maximum number of FDB entries

  Default value 5000

  Memory footprint per FDB entry 40 B

Example

```
FP_OPTIONS="--mod-opt=bridge:--fdb-max=5000"
```

--iface-hash-order

Size order of bridge interfaces hash table. Value automatically updated if *--iface-max* is changed.

Default value

Range 1 .. 31

Example

```
FP_OPTIONS="--mod-opt=bridge:--iface-hash-order=8"
```

--fdb-hash-order

Size order of bridge fdb hash table. Value automatically updated if *--fdb-max* is changed.

Default value

Range 1 .. 31

Example

```
FP_OPTIONS="--mod-opt=bridge:--fdb-hash-order=8"
```

Tip: To get optimal performance, apply the following ratios to these parameters:

Parameter	Value
–iface-hash-order	`N`
–iface-max	`2 ** N`
–fdb-hash-order	`M`
–fdb-max	`2 ** M`

Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

---

2.3.4 Fast Path Filtering IPv4

Overview

Fast Path Filtering IPv4 provides IPv4 filtering in the fast path.

To ensure maximal performance, this module implements simple functions based on information found in the shared memory.

If the module cannot find in the shared memory the relevant information based on L3, L4, and L5 headers, the fast path raises an exception.

In accordance with configured filter rules with higher priorities, this exception:

- interacts with other 6WINDGate entities, or,
- drops the packet for security reasons.

Features

- Filtering, stateless 5-tuple based ACLs (Access Control Lists) (Netfilter)
- Support filter, mangle, raw and NAT tables
- Rules per VR
- “Netfilter-like” connection tracking (limited to tcp, udp, sctp, gre, ah, esp and ipip protocols)
- Fast lookup tables
- RPF (Reverse Path Filtering) Check
- **Support for the following netfilter targets**
  - DROP
  - ACCEPT
  - RETURN
  - SNAT
  - DNAT
  - MARK
  - DSCP
  - TOS
  - REJECT
  - CHECKSUM
  - MASQUERADE

  - NETMAP
  - NOTRACK
  - CT (limited to –zone, –notrack and –helper)
  - CONNMARK
  - CLASSIFY
  - user-defined chains
- Support for conntrack zones
- **Support for the following netfilter matches:**
  - comment
  - connmark
  - conntrack (limited to –ctstate, –ctdir, –ctstatus)
  - devgroup
  - dscp
  - frag
  - hashlimit
  - icmp
  - icmp6
  - limit
  - mac
  - mark
  - multiport
  - owner (limited support)
  - physdev
  - policy (see limitations below)
  - rpfilter
  - sctp
  - set
  - state
  - tcp
  - tcpmss
  - tos

  - udp
- **Support for the following ip set types:**
  - hash:ip
  - hash:net
  - hash:mac
  - hash:net,net
  - hash:net,port
  - hash:net,port,net
  - hash:ip,port
  - hash:ip,port,ip
  - hash:ip,port,net

policy match support

Options supported: –dir {in|out} –pol {none|ipsec) –proto {ah|esp} [!] –mode {tunnel|transport}

All other options are not supported.

Dependencies

6WINDGate modules

- *Fast Path Forwarding IPv4*

Linux

- Netfilter: create audit records for x_tables replaces is a kernel patch (upstream 3.9)
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=fbabf31e4d482149b5e
- RPF: netfilter: export xt_rpfilter.h to userland is a kernel patch (upstream 3.12)
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=f0c03956ac40fdc4fb

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

```
# modprobe nf_conntrack_netlink
```

Example

```
# echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
# ip link set eth1 up
# ip link set eth2 up
# ip ad ad 2.0.0.1/24 dev eth1
# ip ad ad 2.1.0.1/24 dev eth2
# ip route add 100.2.2.1/32 via 2.0.0.5
# ip route add 110.2.2.1/32 via 2.1.0.5
# iptables -A FORWARD -s 100.2.2.1 -j DROP
# iptables -vL
  Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
   pkts bytes target     prot opt in     out     source               destination

  Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
   pkts bytes target     prot opt in     out     source               destination
      0     0 DROP       all  --  any    any     100.2.2.1            anywhere

  Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
   pkts bytes target     prot opt in     out     source               destination
```

Filtering management

Displaying netfilter status

Synopsis

```
nf4
```

Example

```
<fp-0> nf4
IPv4 netfilter is off
```

Enabling filtering in the fast path

Not enabled by default. Automatically set to on when configuring Netfilter rules with the cache manager running.

Synopsis

```
nf4-set on|off
```

on or off Enable or disable Netfilter.

Example

```
<fp-0> nf4-set on
IPv4 netfilter is on
```

Displaying a Netfilter table

Synopsis

```
nf4-rules filter|mangle|raw|nat [nonzero]
```

filter or mangle or raw or nat Display the filter table, the mangle table, the raw table or the NAT table.

nonzero Display only rules with non null statistics.

Example

```
<fp-0> nf4-rules filter
Chain INPUT (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source      destination

Chain FORWARD (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source      destination
       0          0 DROP      all  --   any    any    100.2.2.1   anywhere
```

(continues on next page)

(continued from previous page)

```
Chain OUTPUT (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source      destination
```

Displaying the hook priority table or hooks in nf_conf

Synopsis

```
nf4-hook nf_conf|priority
```

nf_conf Display all hooks present in the nf_conf structure.

priority Display the hook priority table.

Example

```
<fp-0> nf4-hook priority
FP_NF_IP_PRE_ROUTING:
        raw ct mangle nat
FP_NF_IP_LOCAL_IN:
        mangle filter nat
FP_NF_IP_FORWARD:
        mangle filter
FP_NF_IP_LOCAL_OUT:
        raw ct mangle nat filter
FP_NF_IP_POST_ROUTING:
        mangle nat
```

Enabling or disabling hooks in nf_conf

Synopsis

```
nf4-hook-set TABLE|all_tables HOOK|all_hooks on|off
```

TABLE or all_tables The table the hook belongs to. all_tables means all hooks in all tables.

HOOK or all_hooks The hook to enable or disable. all_hooks means all hooks within the table selected just before.

on or off Enable or disable the hook.

Example

```
<fp-0> nf4-hook-set all_tables all_hooks on
Set filter local_in: on
Set filter forward: on
Set filter local_out: on
Set mangle pre_routing: on
Set mangle local_in: on
Set mangle forward: on
Set mangle local_out: on
Set mangle post_routing: on
Set raw pre_routing: on
Set raw local_out: on
set ct pre_routing: on
set ct local_out: on
Set nat pre_routing: on
Set nat local_in: on
Set nat local_out: on
Set nat post_routing: on
```

Displaying the Netfilter cache

Display the status of the Netfilter cache in the fast path.

Synopsis

```
nf4-cache [<num>]
```

<num> Maximum number of cache lines to display.

Example

```
<fp-0> nf4-cache
nf-cache is on
Max cached rules per entry is 8
9: 110.2.2.1 -> 100.2.2.1 tos 0 frag_flags 0x2 TCP sport 6050 dport 6050 flags A----␣
↪vr 0 indev eth2-vr0 outdev eth1-vr0 table 0 hook 2 direct-accept
      #1: target STANDARD, verdict: FP_NF_ACCEPT
10: 110.2.2.1 -> 100.2.2.1 tos 0 frag_flags 0x2 TCP sport 6050 dport 6050 flags AP---␣
↪vr 0 indev eth2-vr0 outdev eth1-vr0 table 0 hook 2 direct-accept
      #1: target STANDARD, verdict: FP_NF_ACCEPT
```

(continues on next page)

(continued from previous page)

```
11: 100.2.2.1 -> 110.2.2.1 tos 0 frag_flags 0x2 TCP sport 6050 dport 6050 flags A----␣
↪vr 0 indev eth1-vr0 outdev eth2-vr0 table 0 hook 2 direct-accept
      #1: target STANDARD, verdict: FP_NF_ACCEPT
12: 100.2.2.1 -> 110.2.2.1 tos 0 frag_flags 0x2 TCP sport 6050 dport 6050 flags AP---␣
↪vr 0 indev eth1-vr0 outdev eth2-vr0 table 0 hook 2 direct-accept
      #1: target STANDARD, verdict: FP_NF_ACCEPT
```

Enabling, disabling or invalidating Netfilter cache

Synopsis

```
nf4-cache-set [on|off|invalidate|drop (on|off)]
```

on or off Enable or disable Netfilter cache.

invalidate Invalidate Netfilter cache.

'drop on' or 'drop off' Enable or disable the Netfilter drop cache. This feature is available when the cache is enabled.

Example

```
<fp-0> nf4-cache-set on
nf-cache is on
```

Displaying the Netfilter conntrack table

Synopsis

```
nfct4 [<number of entries>] [summary]
```

<number of entries> Maximum number of conntrack to display at once.

summary Shorten displayed data to one line per conntrack.

Example

```
# iptables -F
# iptables -P INPUT DROP
# iptables -P FORWARD DROP
# iptables -P OUTPUT DROP
# iptables -A FORWARD -i eth1 -o eth2 -p tcp --dport 6050 -m state --state NEW,
↪ESTABLISHED -j ACCEPT
# iptables -A FORWARD -i eth2 -o eth1 -p tcp --sport 6050 -m state --state NEW,
↪ESTABLISHED -j ACCEPT
# fp-cli
<fp-0> nfct4
Number of flows: 1/1024
Flow: #0
       Proto: 6
       Original: src: 100.2.2.1:6050 -> dst: 110.2.2.1:6050
       Reply:    src: 110.2.2.1:6050 -> dst: 100.2.2.1:6050
       VRF-ID: 0     Zone: 0    Mark: 0x0
       Flag: 0x11, hitflag: 0x01,
                   snat: no, dnat: no,
                   assured: yes, seen_reply: no,
                   unreplied: no, expected: no,
                   update: yes, end: no
       Stats:
               Original: pkt: 24, bytes: 7392
                  Reply:    pkt: 13, bytes: 6820
<fp-0> fct4 1 summary
Number of flows: 1/1024
    index    proto                 original                                    reply    ␣
↪                                stats            flags
#00000000    00006   100.2.2.1:6050 -> 110.2.2.1:6050        | 110.2.2.1:6050 -> 100.2.
↪2.1:6050      [       100 pkt|     30800 B|        51 pkt|     28252 B]       VR0    ␣
↪  Zone0      [ASSURED] [END]
```

Configuring the conntrack refresh policy

Synopsis

```
nfct4-hitflags-set <period in seconds> <max scanned> <max sent>
```

<period in seconds> Period in seconds of connection track checking.

<max scanned> Maximum number of conntracks to scan on a given period of time.

<max sent> Maximum number of refresh messages to send over a given period of time.

Example

```
<fp-0> nfct4-hitflags-set 5 1000 1000
```

Displaying ip sets

Synopsis

```
nf-ipset
```

Example

```
<fp-0> nf-ipset
List of nf ipsets for vrfid 0:
Ipset #0:
        Name: list1
        Type: hash:net
        Family: AF_INET
        NumEntries: 2
        Members:
                20.100.0.0/16
                10.100.0.0/24
Ipset #1:
        Name: list2
        Type: hash:ip
        Family: AF_INET
        NumEntries: 2
        Members:
                20.200.0.2/32
                10.200.0.1/32
Ipset #2:
        Name: list3
        Type: hash:net
        Family: AF_INET6
        NumEntries: 1
        Members:
                fd00:0100:0000:0000:0000:0000:0000:0000/64
Ipset #3:
        Name: list4
        Type: hash:ip
        Family: AF_INET6
        NumEntries: 1
```

(continues on next page)

(continued from previous page)

```
        Members:
                fd00:0200:0000:0000:0000:0000:0000:0001/128
Ipset #4:
        Name: list5
        Type: hash:mac
        Family: AF_UNSPEC
        NumEntries: 1
        Members:
                de:ed:01:ff:ca:f4
```

Displaying hashlimit hashtables content

Synopsis

```
nf4-dump-hashtable [name]
```

Parameters

No parameter Display all existing hashtable contents

name Display the content of hashtable named <name>

Example

```
<fp-0> nf4-dump-hashtable
table#0 : ping
        0 10.100.0.1:0->0.0.0.0:0 7711262310400 348432718233600 69686543646720
        0 10.200.0.1:0->0.0.0.0:0 504919752704 348432718233600 69686543646720
```

Providing options

--max-rules

Maximum number of IPv4 Netfilter rules.

At least 18 IPv4 Netfilter rules (in filter, mangle, raw and nat tables) are created per VR

Default value 3072

Memory footprint per IPv4 Netfilter rule 35 KB

Range 0 .. 40K

--max-ct
Maximum number of IPv4 Netfilter conntracks

Default value 1024

Memory footprint per IPv4 Netfilter conntrack 100 B

Range 0 .. 1M

--max-ipsets
Maximum number of ipsets per VRF

Default value 64

Memory footprint Memory footprint (in bytes) for ipset follows the formula:

```
(8420 + 28 * max-ipset-entries) * max-ipsets * max-vr
```

See –max-vr for default values of max-vr.

Range 0 .. 1000

--max-ipset-entries
Maximum number of entries per ipset

Default value 2048

Memory footprint See *--max-ipsets*

Range 0 .. 1000

--ct-hash-order
Size order of IPv4 conntrack hash table. Value automatically updated if *--max-ct* is changed.

Default value 16

Range 16 .. 20

--cache-order
Size order of IPv4 Netfilter flows stored in cache.

Default value 14

Memory footprint per IPv4 Netfilter flow 128 B

Range 1 .. 31

--cache-hash-order
Size order of IPv4 Netfilter cache hash table. If this value is not specified, it defaults to –cache-order value for better performances.

Default value 14

Memory footprint per IPv4 Netfilter rule 16 B

Range 1 .. 31

Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

2.3.5 Fast Path Filtering IPv6

Overview

Fast Path Filtering IPv6 provides IPv6 filtering in the fast path.

To ensure maximal performance, this module implements simple functions based on information found in the shared memory.

If the module cannot find in the shared memory the relevant information based on L3, L4, and L5 headers, the fast path raises an exception.

In accordance with configured filter rules with higher priorities, this exception:

- interacts with other 6WINDGate entities, or,
- drops the packet for security reasons.

Features

- Filtering, stateless 5-tuple based ACLs (Netfilter)
- Support filter, mangle and raw tables
- Rules per VR
- “Netfilter-like” connection tracking (limited to tcp, udp, sctp, gre, ah, esp and ipip protocols)
- Fast lookup tables
- RPF Check
- **Support for the following netfilter targets**
  - DROP
  - ACCEPT
  - RETURN
  - MARK
  - DSCP
  - TOS
  - REJECT

  - CHECKSUM
  - NOTRACK
  - CT (limited to –zone and –notrack)
  - CONNMARK
  - CLASSIFY
  - user-defined chains
- Support for conntrack zones
- **Support for the following netfilter matches:**
  - comment
  - connmark
  - conntrack (–ctstate only)
  - dscp
  - frag
  - hashlimit
  - icmp
  - icmp6
  - limit
  - mac
  - mark
  - multiport
  - physdev
  - policy (see limitations below)
  - rpfilter
  - sctp
  - set
  - state
  - tcp
  - tos
  - udp
- **Support for the following ip set types:**
  - hash:ip

  - hash:net
  - hash:mac
  - hash:net,net
  - hash:net,port
  - hash:net,port,net
  - hash:ip,port
  - hash:ip,port,ip
  - hash:ip,port,net
- Next hop marking (if fast path filter module is present)

policy match support

Options supported: –dir {in|out} –pol {none|ipsec) –proto {ah|esp} [!] –mode {tunnel|transport}

All other options are not supported.

Dependencies

6WINDGate modules

- *Fast Path Filtering IPv4*

Linux

- Netfilter: create audit records for x_tables replaces is a kernel patch (upstream 3.9)
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=fbabf31e4d482149b5e
- RPF netfilter: export xt_rpfilter.h to userland is a kernel patch (upstream 3.12)
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=f0c03956ac40fdc4fb
- Netfilter: support for 'hash:mac' ipset is a kernel patch (upstream 3.18)
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=07034aeae152

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

```
# modprobe nf_conntrack_netlink
```

Example

```
# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
# ip link set eth1 up
# ip link set eth2 up
# ip ad ad 3ffe:2:100::1/64 dev eth1
# ip ad ad 3ffe:2:110::1/64 dev eth2
# ip route add 3ffe:110:2:2::1/128 via 3ffe:2:11::5
# ip route add 3ffe:100:2:2::1/128 via 3ffe:2:10::5
# ip6tables -F
# ip6tables -P INPUT ACCEPT
# ip6tables -P FORWARD ACCEPT
# ip6tables -P OUTPUT ACCEPT
# ip6tables -A FORWARD -p icmpv6 -s 3ffe:110:2:2::1 -j DROP
# ip6tables -A FORWARD -p icmpv6 -s 3ffe:100:2:2::1 -j DROP
# ip6tables -vL
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 DROP       ipv6-icmp    any    any     3ffe:110:2:2::1   anywhere
    0     0 DROP       ipv6-icmp    any    any     3ffe:100:2:2::1   anywhere

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Filtering management

Displaying Netfilter status

Synopsis

```
nf6
```

Example

```
<fp-0> nf6
IPv6 netfilter is off
```

Enabling or disabling Netfilter in the fast path

Enable IPv6 filtering in fast path. Not enabled by default. Automatically set to on when configuring Netfilter rules with the cache manager running.

Synopsis

```
nf6-set on|off
```

Parameters

on or off Enable or disable Netfilter.

```
<fp-0> nf6-set on
IPv6 netfilter is on
```

Displaying a Netfilter table

Synopsis

```
nf6-rules filter|mangle|raw [nonzero]
```

filter or mangle or raw Display the filter table, the mangle table or the raw table.

nonzero Display only rules with non null statistics.

Example

```
<fp-0> nf6-rules filter
Chain INPUT (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source               destination

Chain FORWARD (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source               destination
```

(continues on next page)

(continued from previous page)

```
       0           0 DROP       ipv6-icmp      any    any    3ffe:110:2:2::1   anywhere
       0           0 DROP       ipv6-icmp      any    any    3ffe:100:2:2::1   anywhere

Chain OUTPUT (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source               destination
```

Displaying the hook priority table or hooks in `nf_conf`

Synopsis

```
nf6-hook nf_conf|priority
```

nf_conf Display all hooks present in the `nf_conf` structure.

priority Display the hook priority table.

Example

```
<fp-0> nf6-hook priority
FP_NF_IP_PRE_ROUTING:
        raw ct mangle
FP_NF_IP_LOCAL_IN:
        mangle filter
FP_NF_IP_FORWARD:
        mangle filter
FP_NF_IP_LOCAL_OUT:
        raw ct mangle filter
FP_NF_IP_POST_ROUTING:
        mangle
```

Enabling or disabling hooks in `nf_conf`

Synopsis

```
nf6-hook-set TABLE|all_tables HOOK|all_hooks on|off
```

TABLE or all_tables The table the hook belongs to. `all_tables` means all hooks in all tables.

HOOK or all_hooks The hook to enable or disable. `all_hooks` means all hooks within the table selected just before.

on or off Enable or disable the hook.

Example

```
<fp-0> nf6-hook-set all_tables all_hooks on
set filter local_in: on
set filter forward: on
set filter local_out: on
set mangle pre_routing: on
set mangle local_in: on
set mangle forward: on
set mangle local_out: on
set mangle post_routing: on
set raw pre_routing: on
set raw local_out: on
set ct pre_routing: on
set ct local_out: on
```

Displaying the Netfilter cache

Display the status of the Netfilter cache in the fast path.

Synopsis

```
nf6-cache [<num>]
```

<num> Maximum number of cache lines to display.

Example

```
<fp-0> nf6-cache
nf6-cache is on
Max cached rules per entry is 11
2: 3ffe:100:2:2::1 -> 3ffe:110:2:2::1 tcclass 0x0 TCP sport 6050 dport 6050 flags AP---
↪ vr 0 indev eth1-vr0 outdev eth2-vr0 table 0 hook 2 direct-accept
    #1: target STANDARD, verdict: FP_NF_ACCEPT
3: 3ffe:110:2:2::1 -> 3ffe:100:2:2::1 tcclass 0x0 TCP sport 6050 dport 6050 flags AP---
↪ vr 0 indev eth2-vr0 outdev eth1-vr0 table 0 hook 2 direct-accept
    #1: target STANDARD, verdict: FP_NF_ACCEPT
```

Enabling, disabling or invalidating Netfilter cache

Synopsis

```
nf6-cache-set [on|off|invalidate|drop (on|off)]
```

on or off Enable or disable Netfilter cache.

invalidate Invalidate Netfilter cache.

'drop on' or 'drop off' Enable or disable the Netfilter drop cache. This feature is available when the cache is enabled.

Example

```
<fp-0> nf6-cache-set on
nf6-cache is on
```

Displaying the Netfilter conntrack table

Synopsis

```
nfct6 [<number of entries>] [summary]
```

Parameters

<number of entries> Maximum number of conntrack entries to display simultaneously.

summary Shorten displayed data to one line per conntrack.

Example

```
<fp-0> nfct6
Number of flows: 1/1024
Flow: #0
     Proto: 6
     Original: src: 3ffe:0100:0002:0002:0000:0000:0000:0001:6050
               dst: 3ffe:0110:0002:0002:0000:0000:0000:0001:6050
     Reply:    src: 3ffe:0110:0002:0002:0000:0000:0000:0001:6050
               dst: 3ffe:0100:0002:0002:0000:0000:0000:0001:6050
     VRF-ID: 0       Zone: 0    Mark: 0x0
```

(continues on next page)

(continued from previous page)

```
    Flag: 0x11, hitflag: 0x01,
                assured: yes, seen_reply: no,
                unreplied: no, expected: no,
                update: yes, end: no
    Stats:
            Original: pkt: 99, bytes: 32216
            Reply:    pkt: 49, bytes: 28616
```

Configuring the conntrack refresh policy

Synopsis

```
nfct6-hitflags-set <period in seconds> <max scanned> <max sent>
```

Parameters

<period in seconds> Period in seconds of connection track checking.

<max scanned> Maximum number of conntracks to scan on a given period of time.

<max sent> Maximum number of refresh messages to send over a given period of time.

Example

```
<fp-0> nfct6-hitflags-set 7 500 500
```

Displaying hashlimit hashtables content

Synopsis

```
nf6-dump-hashtable [name]
```

Parameters

No parameter Display all existing hashtable contents

name Display the content of hashtable named <name>

Example

```
<fp-0> nf6-dump-hashtable
table#0 : ping
         9 fd00:0100:0000:0000:0000:0000:0000:0001:0->
↪0000:0000:0000:0000:0000:0000:0000:0000:0 6171792834560 348432718233600␣
↪69686543646720
         9 fd00:0200:0000:0000:0000:0000:0000:0001:0->
↪0000:0000:0000:0000:0000:0000:0000:0000:0 1160680046592 348432718233600␣
↪69686543646720
```

Providing options

--max-rules
: Maximum number of IPv6 Netfilter rules.

  At least 13 IPv6 Netfilter rules (in filter, mangle and raw tables) are created per VR

  Default value 2048

  Memory footprint per IPv6 Netfilter rule 35 KB

  Range 0 .. 40K

--max-ct
: Maximum number of IPv6 Netfilter conntracks

  Default value 1024

  Memory footprint per IPv6 Netfilter conntrack 100 B

  Range 0 .. 1M

--ct-hash-order
: Size order of IPv6 conntrack hash table. Value automatically updated if *--max-ct* is changed.

  Default value 16

  Range 16 .. 20

--cache-order
: Size order of IPv6 Netfilter flows stored in cache.

  Default value 14

Memory footprint per IPv6 Netfilter flow 128 B

Range 1 .. 31

`--cache-hash-order`
Size order of IPv6 Netfilter cache hash table. If this value is not specified, it defaults to –cache-order value for better performances.

Default value 14

Memory footprint per IPv6 Netfilter rule 16 B

Range 1 .. 31

Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

---

2.3.6 Fast Path Flow Inspection / Packet Capture

Overview

Fast Path Flow Inspection / Packet Capture provides the ability to analyze packets coming in and out of the fast path interfaces (*a la tcpdump*).

Features

- Packets capture in Linux by means of standard tools like `tcpdump`
- BPF filtering automatic configuration in fast path

The fast path still processes the original packets while a copy is sent to the Linux stack for display only.

Dependencies

6WINDGate modules

- *Fast Path Baseline*

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

Warning: The BPF (Berkeley Packet Filtering) filter is automatically synchronized. To disable automatic synchronization, run the cache manager with the `-D` option:

```
# cmgr -D
```

Then, to manually enable TAP per interface:

```
# fp-cli
<fp-0> tap-iface-set eth2 on
```

The *tcpdump -i eth2* program will work the same as described in the case of automatic synchronization.

TAP management

Adding a trailer into tapped packets

Synopsis

A trailer ("FASTPATH OFFLOAD" in ascii) can be added into packets tapped in the fast path. This helps to identify if the packets have been tapped in the fast path or in the kernel. This command enables or disables this feature.

```
tap-trailer-set on|off
```

on or off Enable / disable trailer into tapped packets

Example

Enable the "FASTPATH OFFLOAD" trailer.

```
<fp-0> tap-trailer-set on
```

ICMP packets that are forwarded by the fast path when the trailer is enabled:

```
# tcpdump -nXi eth2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth2, link-type EN10MB (Ethernet), capture size 262144 bytes
16:51:02.537566 IP 10.100.0.1 > 10.200.0.1: ICMP echo request, id 347, seq 1, length 64
       0x0000:  4500 0054 5d8b 4000 3f01 c8f0 0a64 0001  E..T].@.?....d..
       0x0010:  0ac8 0001 0800 62d8 015b 0001 66b3 f35d  ......b..[..f..]
```

(continues on next page)

(continued from previous page)

```
        0x0020:  0000 0000 72e7 0800 0000 0000 1011 1213  ....r...........
        0x0030:  1415 1617 1819 1a1b 1c1d 1e1f 2021 2223  .............!"#
        0x0040:  2425 2627 2829 2a2b 2c2d 2e2f 3031 3233  $%&'()*+,-./0123
        0x0050:  3435 3637 4641 5354 5041 5448 204f 4646  4567FASTPATH.OFF
        0x0060:  4c4f 4144                                LOAD
```

Displaying the trailer status

Synopsis

```
tap-trailer
```

Example

```
<fp-0> tap-trailer
"FASTPATH OFFLOAD" trailer into tapped packet: off.
```

BPF management

Displaying BPF filters

Synopsis

```
tap-bpf [all [raw]]
```

all All BPF with decoded instructions.

raw All BPF in raw format.

Example

```
<fp-0> tap-bpf
BPF list (ifuid 0 is the virtual interface "any"):
8: eth1-vr0, instance 0 (# cmds: 1)
9: eth2-vr0, instance 0 (# cmds: 1)
```

Setting manually the interface for tapping

Synopsis

```
tap-iface-set <ifname>|any on|off
```

<ifname> or any Interface name or any for select all interface.

on or off Enable / disable TAP.

Example

Enable TAP on eth2

```
<fp-0> tap-iface-set eth2 on
```

Displaying TAP state

Synopsis

```
tap
```

Example

Displaying TAP:

```
<fp-0> tap
TAP is off
```

Enabling or disabling TAP

Synopsis

```
tap-set on|off
```

on or off Enable / disable TAP.

Example

Enable TAP:

```
<fp-0> tap-set on
TAP is on (was off)
```

2.3.7 Fast Path Forwarding IPv4

Overview

Fast Path Forwarding IPv4 provides IPv4 forwarding in the fast path.

Features

- IP forwarding
- IP fragmentation
- ECMP (Equal Multipath) with priority per route type
- VRF support
- IPv4 reverse path forwarding check
- TCPmss clamping per interface
- Next hop marking (if fast path filter module is present)
- Routes to same destination with different metrics

Dependencies

6WINDGate modules

- *Fast Path Baseline*

Linux

- Synchronization of interface flag status `forwarding` is a kernel patch (upstream 3.8).

  Without this patch, the fast path starts with forwarding enabled.

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=edc9e748934c

- RPF: Synchronization of interface flag `rp_filter` is a kernel patch (upstream 3.8).

  Without this patch, RPF must be configured manually via `fp-cli`.

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=cc535dfb6a85

- Optimization of synchronization of *ARP entries* is a kernel patch (upstream 3.13).

  Without this patch, the ARP entries stand in state STALE and the fast path continuously sends hitflags for them.

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=53385d2d1de84f4036a0919ec46964c4e81b83f5

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

Example

```
# echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
# ip addr add 192.168.1.1/24 dev eth1
# ip link set up dev eth1
# ip route add default gateway 192.168.1.254 dev eth1
# fp-cli
```

```
<fp-0> route4
# - Preferred, * - Active, > - selected
0.0.0.0/0  [05]  ROUTE gw 192.168.1.254 via eth1-vr0 (8)
```

```
<fp-0> iface
1:lo [VR-0] ifid=1 (virtual) <UP|RUNNING|FWD4|FWD6> (0x63)
          type=loop mac=00:00:00:00:00:00 mtu=16436 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
7:eth3 [VR-0] ifid=7 (port 2) <FWD4|FWD6> (0x60)
          type=ether mac=00:1b:21:c5:7f:76 mtu=1500 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
8:eth2 [VR-0] ifid=8 (port 1) <UP|RUNNING|FWD4|FWD6> (0x63)
          type=ether mac=00:1b:21:c5:7f:75 mtu=1500 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
9:eth1 [VR-0] ifid=9 (port 0) <UP|RUNNING|FWD4|FWD6> (0x63)
          type=ether mac=00:1b:21:c5:7f:74 mtu=1500 tcp4mss=0 tcp6mss=0
```

(continues on next page)

(continued from previous page)

```
          IPv4 routes=2  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
10:eth0 [VR-0] ifid=10 (virtual) <FWD4|FWD6> (0x60)
          type=ether mac=00:21:85:c1:82:58 mtu=1500 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
11:eth4 [VR-0] ifid=11 (port 3) <FWD4|FWD6> (0x60)
          type=ether mac=00:1b:21:c5:7f:77 mtu=1500 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
```

ECMP routes

Synopsis

An ECMP route is a single route to a destination, with several next hops. The traffic matching this route is dispatched among the specified next hops based on a hash of the packet source and destination IP addresses. That way, packets of a same flow follow the same path.

IPv4 ECMP routes are configured in one step: the route is created with its list of next hops.

Example

Enable IPv4 forwarding and configure `eth1` and `eth2`:

```
# echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
# ip addr add 192.168.1.1/24 dev eth1
# ip link set up dev eth1
# ip addr add 192.168.2.1/24 dev eth2
# ip link set up dev eth2
```

Add an ECMP route to `10.0.0.0/8` via gateways `192.168.1.2` and `192.168.2.5`:

```
# ip route add 10.0.0.0/8 nexthop via 192.168.1.2 nexthop via 192.168.2.5
```

Display Linux IPv4 routes:

```
# ip route
10.0.0.0/8
        nexthop via 192.168.1.2  dev eth1 weight 1
        nexthop via 192.168.2.5  dev eth2 weight 1
192.168.1.0/24 dev eth1  proto kernel  scope link  src 192.168.1.1
192.168.2.0/24 dev eth2  proto kernel  scope link  src 192.168.2.1
```


Display fast path IPv4 routes:

```
<fp-0> route4
# - Preferred, * - Active, > - selected
(254) 10.0.0.0/8 Multipath Entry (14)
     [10]  #    (p=001)  ROUTE gw 192.168.2.5 via eth2-vr0
     [09]  #    (p=001)  ROUTE gw 192.168.1.2 via eth1-vr0
```

Deleting an IPv4 ECMP is atomic. Even if a next hop was specified in the delete command, the route is entirely deleted.

Example

Delete the ECMP route to `10.0.0.0/8`:

```
# ip route del 10.0.0.0/8
```

Display Linux IPv4 routes:

```
# ip route
192.168.1.0/24 dev eth1  proto kernel  scope link  src 192.168.1.1
192.168.2.0/24 dev eth2  proto kernel  scope link  src 192.168.2.1
```

Display fast path IPv4 routes:

```
<fp-0> route4
# - Preferred, * - Active, > - selected
```

ECMP algorithm

The default fast path algorithm to select nexthops of IPv4 ECMP routes is different from the one used by linux. The fast path algorithm is more efficient. However, in some cases, this algorithm should be the same than linux.

To define the IPv4 ECMP algorithm use the `route4-ecmp-algo` and `route4-ecmp-algo-set` commands. It is valid only for linux version between 4.4 and 4.11 (both included).

Displaying IPv4 ECMP algorithm

Synopsis

```
route4-ecmp-algo
```

Parameters

Example

```
<fp-0> route4-ecmp-algo
IPv4 ECMP algo is fp
```

Defining IPv4 ECMP algorithm

The default IPv4 ECMP algorithm is an optimized one. It is possible to switch to the 'linux' algorithm only if the routing table doesn't contain IPv4 ECMP routes.

Synopsis

```
route4-ecmp-algo-set linux|fp|fp-gtp-teid
```

Parameters

linux Same as linux.

fp Optimized algorithm, different from linux.

fp-gtp-teid Same as fp with TEID from GTPv1 header taken into account.

Example

```
<fp-0> route4-ecmp-algo
IPv4 ECMP algo is fp
<fp-0> route4-ecmp-algo-set linux
IPv4 ECMP algo is linux
<fp-0> route4-ecmp-algo-set fp-gtp-teid
IPv4 ECMP algo is fp-gtp-teid
```

Route metric

Synopsis

The fast path supports several IPv4 routes to the same destination with different metrics (a.k.a. priority or preference). Unlike an ECMP route, packets can only match the route with the lowest metric.

Example

Enable forwarding and configure `eth1` and `eth2`:

```
# echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
# ip addr add 192.168.1.1/24 dev eth1
# ip link set up dev eth1
# ip addr add 192.168.2.1/24 dev eth2
# ip link set up dev eth2
```

Add 2 routes to `10.0.0.0/8`, with different metrics:

```
# ip route add 10.0.0.0/8 metric 200 via 192.168.1.2
# ip route add 10.0.0.0/8 metric 100 via 192.168.2.5
```

Display Linux IPv4 routes:

```
root@dut-vm:~# ip route
10.0.0.0/8 via 192.168.2.5 dev eth2  metric 100
10.0.0.0/8 via 192.168.1.2 dev eth1  metric 200
192.168.1.0/24 dev eth1  proto kernel  scope link  src 192.168.1.1
192.168.2.0/24 dev eth2  proto kernel  scope link  src 192.168.2.1
```

Display fast path IPv4 routes:

```
<fp-0> route4
# - Preferred, * - Active, > - selected
(254) 10.0.0.0/8 metric 100 [16]  ROUTE gw 192.168.2.5 via eth2-vr0 (15)
(254) 10.0.0.0/8 metric 200 [15]  ROUTE gw 192.168.1.2 via eth1-vr0 (14)
```

Each of these routes may itself be an ECMP route. Change route with metric 100 to an ECMP route:

```
# ip route change 10.0.0.0/8 metric 100 nexthop via 192.168.1.10 nexthop via 192.168.2.
↪11
```

Display Linux IPv4 routes:

```
root@dut-vm:~# ip route
10.0.0.0/8  metric 100
        nexthop via 192.168.1.10  dev eth1 weight 1
        nexthop via 192.168.2.11  dev eth2 weight 1
10.0.0.0/8 via 192.168.1.2 dev eth1  metric 200
192.168.1.0/24 dev eth1  proto kernel  scope link  src 192.168.1.1
192.168.2.0/24 dev eth2  proto kernel  scope link  src 192.168.2.1
```

Display fast path IPv4 routes:

```
<fp-0> route4
# - Preferred, * - Active, > - selected
(254) 10.0.0.0/8 metric 100 Multipath Entry (15)
     [20]  #    (p=001)  ROUTE gw 192.168.2.11 via eth2-vr0
     [19]  #    (p=001)  ROUTE gw 192.168.1.10 via eth1-vr0
(254) 10.0.0.0/8 metric 200 [15]  ROUTE gw 192.168.1.2 via eth1-vr0 (14)
```

Neighbors management

neigh4

Description

Display neighbors table.

Synopsis

```
neigh4 [<rt index>]
```

<rt index> Index of the wanted RT.

Example

```
<fp-0> neigh4
R[000006] GW/NEIGH 10.23.4.204 00:1b:21:cc:0d:97 via eth3_0-vr0 REACHABLE (nh:6)
```

arp-hitflags-set

Description

Set interface ARP hitflags.

Synopsis

```
arp-hitflags-set <period in seconds> <max scanned> <max sent>
```

Parameters

<period in seconds> Period in seconds of connection track checking.

<max scanned> Maximum number of ARP to scan on a given period of time.

<max sent> Maximum number of refresh messages to send over a given period of time.

Example

```
<fp-0> arp-hitflags-set 8 600 800
```

Displaying ARP/CT hitflags parameters

Description

Synopsis

```
hitflags [all]

hitflags [all|arp|conntrack|...]
```

Parameters

No parameter Display parameters for all categories.

all Same as wit no parameters.

arp Display only ARP category.

conntrack Display only IPv4 conntrack category.

Example

```
<fp-0> hitflags arp
arp hitflags
  period_in_seconds:8
  max_scanned:600
  max_sent:800
```

Addresses management

addr4

Description

Display IPv4 addresses of a given interface.

Synopsis

```
addr4 <iface>
```

Parameters

<iface> Name of the interface.

Examples

```
<fp-0> addr4 eth0
number of ip address: 1
10.0.2.15 [0]
```

Routes management

rt4

Description

Display the `rt_entry` table.

Synopsis

```
rt4 [<rt index>]
```

Parameters

No parameter Display whole `rt_entry` table.

<rt index> Index to display specifically in `rt_table`.

Example

```
<fp-0> rt4
R[000001] GW/ADDRESS 0.0.0.0 00:00:00:00:00:00 via eth1-vr0 NONE (nh:1)
R[000002] IFACE/CONNECTED src 10.22.4.104 via eth1-vr0 (nh:2)
R[000003] GW/ADDRESS 0.0.0.0 00:00:00:00:00:00 via eth3-vr0 NONE (nh:3)
R[000004] IFACE/CONNECTED src 10.23.4.104 via eth3-vr0 (nh:4)
R[000005] GW/NEIGH 10.23.4.204 00:1b:21:cc:0d:97 via eth3-vr0 REACHABLE (nh:5)
R[000006] GW/NEIGH 10.22.4.118 00:15:17:34:2a:d8 via eth1-vr0 REACHABLE (nh:6)
R[000007] GW/NEIGH 10.23.4.204 00:1b:21:cc:0d:97 via eth3-vr0 REACHABLE (nh:5)
```

nh4

Description

Dump the Next-Hop table.

Synopsis

```
nh4 [<nh index>]
```

Parameters

No parameter Display whole next hop table.

<nh index> Index to display specifically in next hop table.

Example

```
<fp-0> nh4
N4[0001]GW/ADDRESS 0.0.0.0 00:00:00:00:00:00 via eth1-vr0 NONE refcnt=1
N4[0002]IFACE/CONNECTED src 10.22.4.104 via eth1-vr0 refcnt=1
N4[0003]GW/ADDRESS 0.0.0.0 00:00:00:00:00:00 via eth3-vr0 NONE refcnt=1
N4[0004]IFACE/CONNECTED src 10.23.4.104 via eth3-vr0 refcnt=1
N4[0005]GW/NEIGH 10.23.4.204 00:1b:21:cc:0d:97 via eth3-vr0 REACHABLE refcnt=2
N4[0006]GW/NEIGH 10.22.4.118 00:15:17:34:2a:d8 via eth1-vr0 REACHABLE refcnt=1
```

Displaying IPv4 routes

Description

Display the user routing entries.

Synopsis

```
route4 [dst <addr dst>|<addr dst/prefix> [src <addr src>]]|[type TYPE]
```

Parameters

No parameter Display only routes configured by user.

dst <addr dst>|<addr dst/prefix> Search a route by destination address with or without prefix.

src <addr src> Reduce search to a specific source address.

type TYPE Display routes with a specific type:

Type	description
all	Display all kind of routes.
fpm	Display routes configured via `fpm`.
local	Display local routes, ones to hosts on directly connected networks.
neigh	Display routes to neighbor hosts.
connected	Display routes to connected hosts.
black	Display black hole routes.

Example

```
<fp-0> route4
# - Preferred, * - Active, > - selected
0.0.0.0/0  [03]  NEIGH gw 10.0.2.2 via eth0-vr0 (8)
10.200.0.0/24  [11]  NEIGH gw 10.125.0.2 via ntfp2-vr0 (16)
```

```
<fp-0> route4 type fpm
# - Preferred, * - Active, > - selected
0.0.0.0/32  [5001]  LOCAL (1)
224.0.0.0/4  [5001]  LOCAL (3)
255.255.255.255/32  [5001]  LOCAL (2)
```

```
<fp-0> route4 type local
0.0.0.0/32  [5001]  LOCAL (1)
224.0.0.0/4  [5001]  LOCAL (3)
255.255.255.255/32  [5001]  LOCAL (2)
```

```
<fp-0> route4 type neigh
# - Preferred, * - Active, > - selected
10.0.2.2/32  [03]  NEIGH gw 10.0.2.2 (N) via eth0-vr0 (7)
10.0.2.3/32  [02]  NEIGH gw 10.0.2.3 (N) via eth0-vr0 (6)
10.100.0.1/32  [12]  NEIGH gw 10.100.0.1 (N) via ntfp1-vr0 (17)
10.125.0.2/32  [11]  NEIGH gw 10.125.0.2 (N) via ntfp2-vr0 (18)
```

```
<fp-0> route4 type connected
# - Preferred, * - Active, > - selected
10.0.2.0/24  [04]  CONNECTED via eth0-vr0 (9)
10.100.0.0/24  [06]  CONNECTED via ntfp1-vr0 (11)
10.125.0.0/24  [08]  CONNECTED via ntfp2-vr0 (13)
10.175.0.0/24  [10]  CONNECTED via ntfp3-vr0 (15)
```

```
<fp-0> route4 type black
# - Preferred, * - Active, > - selected
127.0.0.0/8  [5002]  BLACKHOLE (4)
```

```
<fp-0> route4 type all
# - Preferred, * - Active, > - selected
0.0.0.0/32  [5001]  LOCAL (1)
0.0.0.0/0  [03]  NEIGH gw 10.0.2.2 via eth0-vr0 (8)
10.0.2.0/24  [04]  CONNECTED via eth0-vr0 (9)
10.0.2.2/32  [03]  NEIGH gw 10.0.2.2 (N) via eth0-vr0 (7)
10.0.2.3/32  [02]  NEIGH gw 10.0.2.3 (N) via eth0-vr0 (6)
```

(continues on next page)

(continued from previous page)

```
10.0.2.15/32  [01]  ADDRESS via eth0-vr0 (5)
10.100.0.0/24  [06]  CONNECTED via ntfp1-vr0 (11)
10.100.0.1/32  [12]  NEIGH gw 10.100.0.1 (N) via ntfp1-vr0 (17)
10.100.0.2/32  [05]  ADDRESS via ntfp1-vr0 (10)
10.125.0.0/24  [08]  CONNECTED via ntfp2-vr0 (13)
10.125.0.1/32  [07]  ADDRESS via ntfp2-vr0 (12)
10.125.0.2/32  [11]  NEIGH gw 10.125.0.2 (N) via ntfp2-vr0 (18)
10.175.0.0/24  [10]  CONNECTED via ntfp3-vr0 (15)
10.175.0.1/32  [09]  ADDRESS via ntfp3-vr0 (14)
10.200.0.0/24  [11]  NEIGH gw 10.125.0.2 via ntfp2-vr0 (16)
127.0.0.0/8  [5002]  BLACKHOLE (4)
224.0.0.0/4  [5001]  LOCAL (3)
```

Saving LPM tree in a DOT file

Dump IPv4 LPM tree in a dot file. This file can be opened with *xdot* or *dotty* to display the routing table as a graph.

Synopsis

```
route4-dot <filename> [table <tableid>]
```

<filename> The name of the file where the data is saved.

table <tableid> The identifier of the table to dump. Default is main table.

Example

```
<fp-0> route4-dot /tmp/ipv4-lpm.dot
```

Displaying the filling of each IPv4 table in memory

Synopsis

```
route4-filling
```

Example

```
<fp-0> route4-filling
IPv4 tables filling:
fp_lpm_table_shared: 128/256 (50.000000%) IPv4:64
fp_lpm_mem: 232448/8388608 (2.770996%)
fp_rt4_table: 9/50001 (0.018000%)
fp_nh4_table: 4/5001 (0.079984%)
```

iface-preference-set

Description

Set interface preference, when routing ECMP, give higher priority when selecting next hop for this interface.

Synopsis

```
iface-preference-set <iface> on|off
```

Parameters

<iface> Name of the interface.

on or off Enable or not preference flag on interface.

Example

```
<fp-0> iface-preference-set eth1 on
```

VRF support

Some commands (like `route4`) are executed in the current VRF, you can use `vrf-exec` or `vrf-set` to execute these commands in another or all VRF.

Example

```
<fp-0> route4
# - Preferred, * - Active, > - selected
100.0.0.0/16  [01]  ROUTE gw 192.168.0.1 via eth0_0-vr0 (5)
<fp-0> vrf-set 1
 New reference for VRF: 1
<fp-1> route4
# - Preferred, * - Active, > - selected
<fp-1> vrf-exec all route4
vrf0:
# - Preferred, * - Active, > - selected
0.0.0.0/0  [02]  NEIGH gw 10.0.2.2 via eth0-vr0 (8)

vrf1:
# - Preferred, * - Active, > - selected

<fp-1>
```

RPF check

RPF check action is managed on a per interface basis. This behavior is synchronized from Linux. It is however possible to force RPF check at fast path level whatever the Linux configuration is, using the `rpf4-set` command.

Displaying IPv4 RPF flag status

Synopsis

```
rpf4 <iface>
```

Parameters

<iface> Name of the interface.

Example

```
<fp-0> rpf4 eth3
eth3: IPv4 RPF is off
```

Setting IPv4 RPF flag

Synopsis

```
rpf4-set <iface> on|off|linux-sync
```

Parameters

<iface> Name of the interface.

on, off or linux-sync "on" means : always perform RPF, "off" means: don't perform RPF at all. "linux-sync" means perfrom RPF based on the config synced from the linux kernel. If this argument is omitted, only the status of the RPF check is displayed.

Example

```
<fp-0> rpf4-set eth3 on
eth3: IPv4 RPF is on
```

```
<fp-0> rpf4-set eth3 linux-sync
eth3: IPv4 RPF is linux-sync (current : on)
```

```
<fp-0> iface
9:eth3 [VR-0] ifid=9 (port 2) <UP|RUNNING|FWD4|FWD6|RPF4> (0x463)
        type=ether mac=00:1b:21:c5:7f:76 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
        if_ops: rx_dev=none tx_dev=none ip_output=none
```

note that the RPF flag displayed by the `iface` command reflects the operating status for RPF, no matter whether coming from Linux synchronization or fast path enforcement.

TCP MSS clamping

TCP MSS (Maximum Segment Size) clamping can be configured by interface.

tcpmss4-set

Description

Change default value of MSS. 0 means no change is made in packet.

Synopsis

```
tcpmss4-set <iface> <mss4>
```

Parameters

<iface> Name of the interface.

<mss4> MSS, default is 0 (disabled).

Example

```
<fp-0> tcpmss4-set eth0 1400
```

Statistics

ip4-stats

Description

Display IPv4 statistics.

Synopsis

```
ip4-stats [percore|agg[regated]] [all]
```

Parameters

percore Display all statistics per core running the fast path.

aggregated Display sum of Linux and fast path statistics.

all Display all statistics (even those that are null).

Example

```
<fp-0> ip4-stats all
 IpForwDatagrams:682
 IpInReceives:682
 IpInDelivers:447
 IpInHdrErrors:0
 IpInTruncatedPkts:0
 IpInAddrErrors:0
 IpDroppedNoArp:0
 IpDroppedNoMemory:0
 IpDroppedForwarding:0
 IpDroppedIPsec:0
 IpDroppedBlackhole:0
 IpDroppedInvalidInterface:0
 IpDroppedNetfilter:0
 IpDroppedRouteException:0
 IpReasmReqds:0
 IpReasmOKs:0
 IpReasmFails:0
 IpReasmExceptions:0
 IpFragOKs:0
 IpFragFails:0
 IpReasmTimeout:0
 IpFragCreates:0
 IpCsumErrors:0
 IpReasmErrorPacketTooShort:0
 IpReasmErrorTooManySegments:0
 IpReasmErrorHeaderEncap:0
 IpReasmErrorQueueFull:0
 IpReasmErrorIPOptionUnsupported:0
 IpReasmErrorSizeExceed:0
```

(continues on next page)

(continued from previous page)

```
IpReasmErrorLastAlreadyReceived:0
IpReasmErrorSizeOverflow:0
IpReasmErrorOverlapPrevious:0
IpReasmErrorOverlapNext:0
IpReasmErrorQueueAlloc:0
IpReasmErrorOffsetTooLarge:0
IpReasmDroppedSessionComplete:0
IpReasmDroppedSessionAlreadyFull:0
IpReasmDroppedDuplicate:0
IpNhrpPacket:0
IpNhrpErrorSend:0
```

Transparent IP options support

The default handling of IP packets using a non-default header length is to send these packets to control plane, where the kernel will have a specific management (for example, add its IP address when a header requires "Record route").

This option is used to ignore the non-default header length of the packets and forward the packets containing options as if they had a default header.

To enable or disable check of IPv4 options use the `ip4-options` and `ip4-options-set` commands. If the options checking is disabled IP packets with non-default header length are transparently forwarded.

Displaying fast path processing of IPv4 packets with options in header

Synopsis

```
ip4-options
```

Parameters

Example

```
<fp-0> ip4-options
IPv4 options are checked
```

Defining fast path processing of IPv4 packets with options in header

The default behavior of the fast path is to send IP packets with non-default header length to the control plane as exception packets.

Synopsis

```
ip4-options-set ignore|check
```

Parameters

ignore or check Check: packets with extended option are sent as exception to the control plane.

Example

```
<fp-0> ip4-options
IPv4 options are checked
<fp-0> ip4-options-set ignore
IPv4 options are ignored
```

Providing options

--max-addr
Maximum number of IPv4 addresses

Default value 4096

Memory footprint per IPv4 address 8 B

Range 0 .. 4M

--max-route
Maximum number of IPv4 routes

Default value 50000

Memory footprint per IPv4 route 50 B

Range 0 .. 4M

--max-neigh
Maximum number of IPv4 neighbors

Default value 5000

Memory footprint per IPv4 neighbor 50 B

Range 0 .. 400K

`--rt4-ecmp-algo=[fp|linux]`
IPv4 ECMP algorithm used by the fast path.

Default value fp

`--max-lpm-tables`
Specify the maximum number of IPv4/IPv6 routing tables.

Default value 256

Note:

- To check the number of tables used by IPv4 protocol, use the *route4-filling* command.
- To check the number of tables used by IPv6 protocol, use the *route6-filling* command.

`--lpm-mem-size`
Specify the size of shared memory reserved for IPv4/IPv6 LPM.

Default value 16777216

Note: This memory is used by internal structures for LPM routing algorithm. The size depends on the number of routes created by the system, and on their distribution. A statistic rule to compute it is: 65536 * VRs + IPv4 routes * 128 + IPv6 routes * 256.

- To check the LPM memory size used by IPv4 protocol, use the *route4-filling* command.
- To check the LPM memory size used by IPv6 protocol, use the *route6-filling* command.

`--max-reass-queues`
Power of 2 of the maximum number of simultaneous reassembly procedures for IPv4

Default value 6

Memory footprint None

Range 0 .. 30

`--reass-hash-order`
Size order of reassembly hash table for IPv4. Value automatically updated if *`--max-reass-queues`* is changed.

Default value 7

Range 1 .. 31

`--max-reass-time`
Maximum lifetime of a reassembly procedure for IPv4 (ms).

Default value 2000

Range 1 .. 100M

`--max-reass-interfrag`
Maximum time between two fragments for a IPv4 reassembly procedure (ms).

Default value 200

Range 1 .. 100M

Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

---

Path MTU Discovery (*PMTUD*)

Print Path MTU Discovery (*PMTUD*) configuration

Synopsis

```
pmtud
```

no arguments Display Path MTU Discovery (*PMTUD*) configuration.

Enable/disable Path MTU Discovery (*PMTUD*)

Synopsis

```
pmtud.enable <on|off>
```

on|off on: enable, off: disable Path MTU Discovery (*PMTUD*).

Expiration timeout of Path MTU Discovery (*PMTUD*)

Synopsis

```
pmtud.expiration_timeout <time>
```

time Maximum lifetime of PMTU records, expressed as a number of minutes. Modifying this value affects both existing and future records. 0 disables expiration altogether; all records become permanent.

Hash table order of Path MTU Discovery (*PMTUD*)

Synopsis

```
pmtud.hash_order <log2>
```

log2 Size of indexing hash table for PMTU records, expressed as a power of two (e.g. 13 means 2 ^ 13, that is, 8192 entries). The resulting value should be at least as large as `pmtud.nbentries` for best performance.

This parameter becomes read-only once fast path initialization is complete.

Maximum number of Path MTU Discovery (*PMTUD*) records

Synopsis

```
pmtud.nbentries <value>
```

value Maximum number of PMTU records. Records are allocated upon reception of a PMTUD message (like ICMP "Packet Too Big") for each unique tuple (e.g. TCP source/destination addresses and ports).

This parameter becomes read-only once fast path initialization is complete.

2.3.8 Fast Path Forwarding IPv6

Overview

Fast Path Forwarding IPv6 provides IPv6 forwarding in the fast path.

Features

Fast path forwarding IPv6 features:

- IP forwarding
- IP fragmentation
- ECMP (Equal Multipath) with priority per route type
- VRF support
- IPv6 reverse path forwarding check
- TCPmss clamping per interface
- Next hop marking (if the fast path filter module is present)
- Routes to same destination with different metrics

- Segment routing header processing and SRv6 route

Dependencies

Modules

- *Fast Path Forwarding IPv4*

Linux

- Synchronization of interface flag status `forwarding` is a kernel patch (upstream 3.8).

  Without this patch, the fast path starts with forwarding enabled.

  rtnl/ipv6: use netconf msg to advertise forwarding status

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=f3a1bfb
- ECMP IPv6 appeared in Linux 3.8

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=51ebd31

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=52bd4c0

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=b3ce5ae

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=1a72418
- Optimization of synchronization of *NDP entries* is a kernel patch (upstream 3.13).

  Without this patch, the NDP entries stand in state STALE and the fast path continuously sends hitflags for them.

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=53385d2d1de84f4036a0919ec46964c4e81b83f5
- Segment routing processing and SEG6 Route appeared first in Linux 4.10, SEG6LOCAL route appeared in Linux 4.14.

  SEG6 mode supported: inline, encap, l2encap

  SEG6LOCAL action supported: End, End.DX2, End.B6, End.B6.Encaps

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

Example

```
# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
# ip addr add 3ffe:2:10::1/64 dev eth1
# ip link set up dev eth1
# ip route add 3ffe:100:2:2::1/128 via 3ffe:2:10::5
# ip -6 route show dev eth1
3ffe:2:10::/64  proto kernel  metric 256
3ffe:100:2:2::1 via 3ffe:2:10::5  metric 1024
fe80::/64  proto kernel  metric 256
# fp-cli
```

```
<fp-0> route6
# - Preferred, * - Active, > - selected
3ffe:100:2:2::1/128  [03]  ROUTE gw 3ffe:2:10::5 via eth1-vr0 (6)

<fp-0> addr6 eth1
number of ip address: 1
3ffe:0002:0010:0000:0000:0000:0000:0001 [0]
```

```
<fp-0> iface
1:lo [VR-0] ifid=1 (virtual) <UP|RUNNING|FWD4|FWD6> (0x63)
          type=loop mac=00:00:00:00:00:00 mtu=16436 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
7:eth3 [VR-0] ifid=7 (port 2) <FWD4|FWD6> (0x60)
          type=ether mac=00:1b:21:c5:7f:76 mtu=1500 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
8:eth2 [VR-0] ifid=8 (port 1) <UP|RUNNING|FWD4|FWD6> (0x63)
          type=ether mac=00:1b:21:c5:7f:75 mtu=1500 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
9:eth1 [VR-0] ifid=9 (port 0) <UP|RUNNING|FWD4|FWD6> (0x63)
          type=ether mac=00:1b:21:c5:7f:74 mtu=1500 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=2
          if_ops: rx_dev=none tx_dev=none ip_output=none
10:eth0 [VR-0] ifid=10 (virtual) <FWD4|FWD6> (0x60)
```

(continues on next page)

(continued from previous page)

```
          type=ether mac=00:21:85:c1:82:58 mtu=1500 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
11:eth4 [VR-0] ifid=11 (port 3) <FWD4|FWD6> (0x60)
          type=ether mac=00:1b:21:c5:7f:77 mtu=1500 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
```

ECMP routes

Synopsis

An ECMP route is a single route to a destination, with several next hops. The traffic matching this route is dispatched among the specified next hops based on a hash of the packet source and destination IP addresses. That way, packets of a same flow follow the same path.

IPv6 ECMP routes are configured in several steps: the route is first created with a nexthop, then extra nexthops are appended by specifying a route to the same destination and with the same metric. The routes will be merged into a single ECMP route.

Example

Enable IPv6 forwarding and configure `eth1` and `eth2`:

```
# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
# ip addr add 2001:db8:1::1/64 dev eth1
# ip link set up dev eth1
# ip addr add 2001:db8:2::1/64 dev eth2
# ip link set up dev eth2
```

Add an ECMP route to `2001:db8:10::/56` via gateways `2001:db8:1::2` and `2001:db8:2::5`:

```
# ip route add 2001:db8:10::/56 via 2001:db8:1::2
# ip route append 2001:db8:10::/56 via 2001:db8:2::5
```

Display Linux IPv6 routes. Note that the ECMP route is displayed as separate routes, although it is actually a single route with several next hops:

```
# ip -6 route
2001:db8:1::/64 dev eth1  proto kernel  metric 256  pref medium
2001:db8:2::/64 dev eth2  proto kernel  metric 256  pref medium
2001:db8:10::/56 via 2001:db8:1::2 dev eth1  metric 1024  pref medium
2001:db8:10::/56 via 2001:db8:2::5 dev eth2  metric 1024  pref medium
```

(continues on next page)

(continued from previous page)

```
fe80::/64 dev mgmt0  proto kernel  metric 256  pref medium
fe80::/64 dev eth1  proto kernel  metric 256  pref medium
fe80::/64 dev eth2  proto kernel  metric 256  pref medium
```

Display fast path IPv6 routes:

```
<fp-0> route6
# - Preferred, * - Active, > - selected
(254) 2001:db8:10::/56 metric 1024 Multipath Entry (8)
     [06]  #    (p=001)  ROUTE gw 2001:db8:2::5 via eth2-vr0
     [05]  #    (p=001)  ROUTE gw 2001:db8:1::2 via eth1-vr0
```

Deleting an IPv6 ECMP route requires to remove all its next hops one by one. If no next hop is specified, then the first one will be removed. The route is actually deleted when all of its next hops were removed.

Example

Delete next hop `2001:db8:1::2` from ECMP route to `2001:db8:10::/56`:

```
# ip route del 2001:db8:10::/56 via 2001:db8:1::2
```

Display Linux IPv6 routes:

```
# ip -6 route
2001:db8:1::/64 dev eth1  proto kernel  metric 256  pref medium
2001:db8:2::/64 dev eth2  proto kernel  metric 256  pref medium
2001:db8:10::/56 via 2001:db8:2::5 dev eth2  metric 1024  pref medium
fe80::/64 dev mgmt0  proto kernel  metric 256  pref medium
fe80::/64 dev eth1  proto kernel  metric 256  pref medium
fe80::/64 dev eth2  proto kernel  metric 256  pref medium
```

Display fast path IPv6 routes:

```
<fp-0> route6
# - Preferred, * - Active, > - selected
(254) 2001:db8:10::/56 metric 1024 [06]  ROUTE gw 2001:db8:2::5 via eth2-vr0 (8)
```

Route metric

Synopsis

The fast path supports several IPv6 routes to the same destination with different metrics (a.k.a. priority or preference). Unlike an ECMP route, packets can only match the route with the lowest metric.

Example

Enable IPv6 forwarding and configure `eth1` and `eth2`:

```
# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
# ip addr add 2001:db8:1::1/64 dev eth1
# ip link set up dev eth1
# ip addr add 2001:db8:2::1/64 dev eth2
# ip link set up dev eth2
```

Add 2 routes to `2001:db8:10::/56` with different metrics:

```
# ip route add 2001:db8:10::/56 metric 200 via 2001:db8:1::2
# ip route add 2001:db8:10::/56 metric 100 via 2001:db8:2::5
```

Display Linux IPv6 routes:

```
root@dut-vm:~# ip -6 route
2001:db8:1::/64 dev eth1  proto kernel  metric 256  pref medium
2001:db8:2::/64 dev eth2  proto kernel  metric 256  pref medium
2001:db8:10::/56 via 2001:db8:2::5 dev eth2  metric 100  pref medium
2001:db8:10::/56 via 2001:db8:1::2 dev eth1  metric 200  pref medium
```

Display fast path IPv6 routes:

```
<fp-0> route6
# - Preferred, * - Active, > - selected
(254) 2001:db8:10::/56 metric 100 [08]  ROUTE gw 2001:db8:2::5 via eth2-vr0 (13)
(254) 2001:db8:10::/56 metric 200 [07]  ROUTE gw 2001:db8:1::2 via eth1-vr0 (12)
```

Each of these routes may itself be an ECMP route. Change route with metric 100 to an ECMP route by adding a next hop:

```
# ip route append 2001:db8:10::/56 metric 100 via 2001:db8:1::2
```

Display Linux IPv6 routes:

```
root@dut-vm:~# ip -6 route
2001:db8:1::/64 dev eth1  proto kernel  metric 256  pref medium
2001:db8:2::/64 dev eth2  proto kernel  metric 256  pref medium
2001:db8:10::/56 via 2001:db8:2::5 dev eth2  metric 100  pref medium
2001:db8:10::/56 via 2001:db8:1::2 dev eth1  metric 100  pref medium
2001:db8:10::/56 via 2001:db8:1::2 dev eth1  metric 200  pref medium
fe80::/64 dev mgmt0  proto kernel  metric 256  pref medium
fe80::/64 dev eth1  proto kernel  metric 256  pref medium
fe80::/64 dev eth2  proto kernel  metric 256  pref medium
```

Display fast path IPv6 routes:

```
<fp-0> route6
# - Preferred, * - Active, > - selected
(254) 2001:db8:10::/56 metric 100 Multipath Entry (13)
     [12]  #    (p=001)  ROUTE gw 2001:db8:1::2 via eth1-vr0
     [13]  #    (p=001)  ROUTE gw 2001:db8:2::5 via eth2-vr0
(254) 2001:db8:10::/56 metric 200 [12]  ROUTE gw 2001:db8:1::2 via eth1-vr0 (12)
```

Neighbors management

Displaying the neighbors table

Synopsis

```
neigh6
```

Example

```
<fp-0> neigh6
R[000006] GW/NEIGH 3ffe:2:10::6 00:1b:21:cc:0d:97 via eth1-vr0 REACHABLE (nh:6)
```

Setting interface NDP hitflags

Synopsis

```
ndp-hitflags-set <period in seconds> <max scanned> <max sent>
```

<period in seconds> Validity period of NDP hitflags, in seconds.

<max scanned> Maximum number of NDP hitflags per synchronization message with `fpm`.

<max sent> Maximum number of NDP hitflags to send.

Example

```
<fp-0> ndp-hitflags-set 8 600 800
```

Displaying NDP/CT6 hitflags parameters

Synopsis

```
hitflags [all|ndp|conntrack6|...]
```

No parameter Display parameters for all categories.

all Same as wit no parameters.

ndp Display only the NDP category.

conntrack6 Display only the IPv6 conntrack category.

Example

```
<fp-0> hitflags ndp
ndp hitflags
  period_in_seconds:1
  max_scanned:2500
  max_sent:1600
```

Addresses management

Displaying IPv6 addresses

Description

Display IPv6 addresses of a given interface.

Synopsis

```
addr6 <iface>
```

Parameters

<iface> Name of the interface.

Examples

```
<fp-0> addr6 eth0
number of ip address: 1
fd00:0175:0000:0000:0000:0000:0000:0001 [2]
```

Routes management

Displaying the rt_entry table

Synopsis

```
rt6 [<rt index>]
```

No parameter Display the whole `rt_entry` table.

<rt index> Index to dump specifically in `rt_table`.

Example

```
<fp-0> rt6
R6[000001]IFACE/LOCAL via ifid0-vr0 (nh:5001)
R6[000002]IFACE/LOCAL via ifid0-vr0 (nh:5001)
R6[000003]IFACE/BLACKHOLE via ifid0-vr0 (nh:5002)
R6[000004]IFACE/CONNECTED via eth1-vr0 (nh:1)
R6[000005]GW/ADDRESS :: 00:00:00:00:00:00 via eth1-vr0 NONE (nh:2)
R6[000006]GW/ROUTE 3ffe:2:10::5 00:00:00:00:00:00 via eth1-vr0 NONE (nh:3)
R6[000007]GW/NEIGH 3ffe:2:10::6 00:15:17:34:2a:d8 via eth1-vr0 REACHABLE (nh:4)
R6[000008]GW/ROUTE fe80::1 00:00:00:00:00:00 via eth1-vr0 NONE (nh:5)
```

Displaying the IPv6 next hop table

Synopsis

```
nh [<nh index>]
```

No parameter Display the whole next hop table.

<nh index> Index to dump specifically in next hop table.

Example

```
<fp-0> nh6
N6[0001] IFACE/CONNECTED via eth1-vr0 refcnt=1
N6[0002] GW/ADDRESS :: 00:00:00:00:00:00 via eth1-vr0 NONE refcnt=1
N6[0003] GW/ROUTE 3ffe:2:10::5 00:00:00:00:00:00 via eth1-vr0 NONE refcnt=1
N6[0004] GW/NEIGH 3ffe:2:10::6 00:15:17:34:2a:d8 via eth1-vr0 REACHABLE refcnt=1
N6[0005] GW/ROUTE fe80::1 00:00:00:00:00:00 via eth1-vr0 NONE refcnt=1
```

Displaying the user routing entries

Display the user routing entries.

Synopsis

```
route6 [dst <addr dst>|<addr dst/prefix> [src <addr src>]]|[type TYPE]
```

No parameter Display only routes configured by user.

dst <addr dst>|<addr dst/prefix> Search a route by destination address with or without prefix.

src <addr src> Reduce search to a specific source address.

type TYPE Display routes with a specific type:

Type	description
all	Display all kind of routes.
fpm	Display routes configured via `fpm`.
local	Display local routes, ones to hosts on directly connected networks.
neigh	Display routes to neighbor hosts.
connected	Display routes to connected hosts.
black	Display black hole routes.
sr6	Display IPv6 segment routing routes.

Example

```
<fp-0> route6
# - Preferred, * - Active, > - selected
3ffe:2:20::/48  [05]  ROUTE gw fe80::1 via eth1-vr0 (8)
3ffe:100:2:2::1/128  [03]  ROUTE gw 3ffe:2:10::5 via eth1-vr0 (6)
```

```
<fp-0> route6 type fpm
```

```
<fp-0> route6 type local
# - Preferred, * - Active, > - selected
fe80::/10  [5001]  LOCAL (1)
ff00::/8  [5001]  LOCAL (2)
```

```
<fp-0> route6 type neigh
# - Preferred, * - Active, > - selected
3ffe:2:10::6/128  [04]  NEIGH gw 3ffe:2:10::6 (N) via eth1-vr0 (7)
```

```
<fp-0> route6 type connected
# - Preferred, * - Active, > - selected
3ffe:2:10::/64  [01]  CONNECTED via eth1-vr0 (4)
```

```
<fp-0> route6 type black
# - Preferred, * - Active, > - selected
::/80  [5002]  BLACKHOLE (3)
```

```
<fp-0> route6 type all
# - Preferred, * - Active, > - selected
::/80  [5002]  BLACKHOLE (3)
3ffe:2:10::/64  [01]  CONNECTED via eth1-vr0 (4)
3ffe:2:10::1/128  [02]  ADDRESS via eth1-vr0 (5)
3ffe:2:10::6/128  [04]  NEIGH gw 3ffe:2:10::6 (N) via eth1-vr0 (7)
3ffe:2:20::/48  [05]  ROUTE gw fe80::1 via eth1-vr0 (8)
3ffe:100:2:2::1/128  [03]  ROUTE gw 3ffe:2:10::5 via eth1-vr0 (6)
fe80::/10  [5001]  LOCAL (1)
ff00::/8  [5001]  LOCAL (2)
```

Saving LPM tree in a DOT file

Dump IPv6 LPM tree in a dot file. This file can be opened with *xdot* or *dotty* to display the routing table as a graph.

Synopsis

```
route6-dot <filename> [table <tableid>]
```

<filename> The name of the file where the data is saved.

table <tableid> The identifier of the table to dump. Default is main table.

Example

```
<fp-0> route6-dot /tmp/ipv6-lpm.dot
```

Displaying the filling of each table in memory

Synopsis

```
route6-filling
```

Example

```
<fp-0> route6-filling
IPv6 tables filling:
fp_lpm_table_shared: 128/256 (50.000000%) IPv6:64
fp_lpm_mem: 232448/8388608 (2.770996%)
fp_rt6_table: 3/50001 (0.006000%)
fp_nh6_table: 0/5001 (0.000000%)
```

VRF support

See also:

Please see the VRF section in the *Fast Path Forwarding IPv4* documentation.

RPF check

Displaying IPv6 RPF flag status

Synopsis

```
rpf6 <iface>
```

<iface> Name of the interface.

Example

```
<fp-0> rpf6 eth3
eth3: IPv6 RPF is off
```

Setting IPv6 RPF flag

Synopsis

```
rpf6-set <iface> on|off
```

<iface> Name of the interface.

on or off Enable/disable the RPF on this interface (optional). If this argument is omitted, only the status of the RPF check is displayed.

Example

```
<fp-0> rpf6-set eth3 on
eth3: IPv6 RPF is on
```

```
<fp-0> iface
9:eth3 [VR-0] ifid=9 (port 2) <UP|RUNNING|FWD4|FWD6|RPF6> (0x663)
        type=ether mac=00:1b:21:c5:7f:76 mtu=1500 tcp6mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
        if_ops: rx_dev=none tx_dev=none ip_output=none
```

TCP MSS clamping

TCP MSS (Maximum Segment Size) clamping can be configured by interface.

Editing the default value of MSS

0 means no change is made in packets.

Synopsis

```
tcpmss6-set <iface> <mss6>
```

<iface> Name of the interface.

<mss6> MSS, default is 0 (disabled).

Example

```
<fp-0> tcpmss6-set eth0 1440
```

Statistics

Displaying IPv6 statistics

Synopsis

```
ip6-stats [percore|agg[regated]] [all]
```

percore Display all statistics per core running the fast path.

aggregated Display sum of Linux and fast path statistics.

all Display all statistics (even those that are null).

Example

```
<fp-0> ip6-stats all
 IpForwDatagrams:232
 IpInReceives:232
 IpInDelivers:124
 IpInHdrErrors:0
 IpInTruncatedPkts:0
 IpInAddrErrors:0
 IpDroppedNoArp:0
 IpDroppedNoMemory:0
 IpDroppedForwarding:0
 IpDroppedIPsec:0
 IpDroppedBlackhole:0
 IpDroppedInvalidInterface:0
 IpDroppedNetfilter:0
 IpDroppedRouteException:0
 IpReasmReqds:0
 IpReasmOKs:0
 IpReasmFails:0
 IpReasmTimeout:0
 IpReasmExceptions:0
 IpFragOKs:0
 IpFragFails:0
 IpFragCreates:0
 IpReasmErrorPacketTooShort:0
 IpReasmErrorTooManySegments:0
 IpReasmErrorHeaderEncap:0
 IpReasmErrorFragmentHeader:0
 IpReasmErrorQueueFull:0
 IpReasmErrorIPOptionTooLarge:0
 IpReasmErrorSizeExceed:0
 IpReasmErrorLastAlreadyReceived:0
 IpReasmErrorSizeOverflow:0
 IpReasmErrorOverlapPrevious:0
 IpReasmErrorOverlapNext:0
 IpReasmErrorQueueAlloc:0
 IpReasmErrorOffsetTooLarge:0
 IpReasmDroppedSessionComplete:0
 IpReasmDroppedSessionAlreadyFull:0
 IpNhrpPacket:0
 IpNhrpErrorSend:0
 IpNhrpPacket:0
 IpSr6HdrProcessed:0
 IpSr6HdrErrors:0
```

(continues on next page)

(continued from previous page)

```
IpSr6Inline:0
IpSr6InlineErrors:0
IpSr6Encap:0
IpSr6EncapErrors:0
IpSr6L2Encap:0
IpSr6L2EncapErrors:0
```

Transparent IPv6 extended header support

The default handling of IPv6 packets with Hop-by-hop extension headers is to send these packets to control plane, where the kernel will have a specific management.

This option is used to ignore the non-default extension header of the packets and forward the packets containing an extended header as if they had a default header.

To enable or disable check of IPv6 options use the `ip6-options` and `ip6-options-set` commands. If the options checking is disabled IP packets with non-default header length are transparently forwarded.

Displaying fast path processing of IPv6 packets with options in header

Synopsis

```
ip6-options
```

Example

```
<fp-0> ip6-options
IPv6 options are checked
```

Defining fast path processing of IPv6 packets with options in header

The default behavior of the fast path is to send IPv6 packets with non-default header lenght to the control plane as exception packets.

Synopsis

```
ip6-options-set ignore|check
```

ignore or check Check: packets with extended option are sent as exception to the control plane.

Example

```
<fp-0> ip6-options
IPv6 options are checked
<fp-0> ip6-options-set ignore
IPv6 options are ignored
```

SRv6 support

Segment Routing over IPv6 is a source routing technique. It uses a type of routing extension header. A segment is encoded as an IPv6 address. An ordered list of segments is encoded as an ordered list of IPv6 addresses in the routing header. The active segment is indicated by the destination address of the packet. The next active segment is indicated by a pointer in the new routing header.

There are 3 different types of nodes that may be involved in segment routing networks:

- Source node
- Transit node
- Endpoint node

To enable an interface as an active segment in Linux:

```
# sysctl -w net.ipv6.conf.all.seg6_enabled=1
# sysctl -w net.ipv6.conf.eth1.seg6_enabled=1
```

SR Source Node

A source node originates an IPv6 packets with a segment routing header.

Example

Add an IPv6 route with SRv6 encapsulation and two segments attached:

```
# ip -6 route add 2001:db8:1::/64 encap seg6 mode encap segs 2001:db8:1::1,
↪2001:db8:1::2 dev eth1
```

Display fast path IPv6 seg6 routes:

```
<fp-0> route6 type sr6
(254) 2001:db8:1::/64 metric 1024 [nh:14]  encap seg6 mode encap segs 2 [␣
↪2001:db8:1::1 2001:db8:1::2 ] SEGMENT ROUTING via eth1-vr0 (rt:17)
```

Modes supported

SEG6 modes currently supported by Fast Path:

- inline: Insert Segment Routing Header after IPv6 header
- encap: Encapsulate L3 packet in an IPv6 header with Segment Routing Header
- l2encap : Encapsulate L2 packet in an IPv6 header with Segment Routing Header

SR Transit Node

A transit node is any node forwarding an IPv6 packet where the destination address of that packet is not locally configured as a segment, it acts like a classic forwarding node. Segment routing support is not required for this type of node.

SR Endpoint Node

An endpoint node is any node receiving an IPv6 packet where the destination address of that packet is locally configured as a segment.

This is possible to define a special behavior for a segment like the example below.

Example

Add an IPv6 route to decapsulate a L2 packet within an IPv6 header with SRH and forward it via eth1 interface:

```
# ip -6 route add fdaa::1/128 encap seg6local action End.DX2 oif eth1 dev eth1
```

Display fast path IPv6 seg6local routes:

```
<fp-0> route6 type sr6
(254) fdaa::1/128 metric 1024 [nh:15]  encap seg6local action End.DX2 oif eth1-vr0␣
↪SEGMENT ROUTING via eth1-vr0 (rt:18)
```

Actions supported

SEG6LOCAL actions currently supported by Fast Path:

- End: Regular endpoint
- End.DX2: Endpoint with decapsulation and L2 cross-connect
- End.B6: Insert a specified SRH before the current SRH
- End.B6.Encaps: Regular endpoint and encapsulate L3 packet in an IPv6 header with SRH

More information about Endpoint actions behaviors: https://tools.ietf.org/html/draft-ietf-spring-srv6-network-programming-28

Providing options

`--max-addr`
: Maximum number of IPv6 addresses

  Default value 4096

  Memory footprint per IPv6 address 20 B

  Range 0 .. 4M

`--max-route`
: Maximum number of IPv6 routes

  Default value 50000

  Memory footprint per IPv6 route 50 B

  Range 0 .. 4M

`--max-neigh`
: Maximum number of IPv6 neighbors

  Default value 5000

  Memory footprint per IPv6 neighbor 50 B

  Range 0 .. 400K

`--max-reass-queues`
: Power of 2 of the maximum number of simultaneous reassembly procedures for IPv6

  Default value 6

Memory footprint None

Range 0 .. 30

`--reass-hash-order`
Size order of reassembly hash table for IPv6. Value automatically updated if *`--max-reass-queues`* is changed.

Default value 7

Range 1 .. 31

`--max-reass-time`
Maximum lifetime of a reassembly procedure for IPv6 (ms).

Default value 2000

Range 1 .. 100M

`--max-reass-interfrag`
Maximum time between two fragments for a IPv6 reassembly procedure (ms).

Default value 200

Range 1 .. 100M

`--max-sr6-segments`
Maximum number of segment for IPV6 segment routing

Default value 32

Range 1 .. 128

`--max-sr6-seg6-lwt`
Maximum number of lwtunnel SEG6 routes

Default value 5000

Memory footprint per SEG6 lwtunnel 60 B + 16 B * max-sr6-segments

Range 1 .. 400k

`--max-sr6-seg6local-lwt`
Maximum number of lwtunnel SEG6_LOCAL routes

Default value 5000

Memory footprint per SEG6 lwtunnel 92 B + 16 B * max-sr6-segments

Range 1 .. 400k

Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

---

2.3.9 Fast Path GRE

Overview

Fast Path GRE provides IPvX and Ethernet over GRE support in the fast path.

Features

- Supported features:
  - IPv4 over IPv4 GRE tunnelling (L3 tunnel)
  - IPv6 over IPv4 GRE tunnelling (L3 tunnel)
  - IPv6 over IPv6 GRE tunnelling (L3 tunnel)
  - IPv4 over IPv6 GRE tunnelling (L3 tunnel)
  - IPV4 over Ether GRE tunnelling (L2 tunnel)
  - IPV6 over Ether GRE tunnelling (L2 tunnel)
- Ethernet GRE can be encapsulated on IPv4 (gretap) or IPv6 (ip6gretap)
- Cross-VRF processing (the encapsulated and plaintext traffic may be in different VRFs, the GRE interface performs the VRF transition).

Dependencies

6WINDGate modules

- *Fast Path Baseline*
- *Fast Path Forwarding IPv4*
- *Fast Path Forwarding IPv6*

Linux

- Full Linux synchronization is a kernel patch (upstream 2.6.28).

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=c19e654ddbe3
- Support of Ether GRE (L2 tunnelling) is a kernel patch (upstream 2.6.28).

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=e1a8000228e1
- GRE over IPv6 support is a kernel patch (upstream v3.7).

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=c12b395a4664

- Cross-VRF support for IPv4 GRE is a kernel patch (upstream v3.16).

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=b57708add314

- Cross-VRF support for IPv6 GRE is a kernel patch (upstream v3.16).

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=22f08069e8b4

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

Managing GRE interfaces from the fast path

The `fp-cli` commands below allow you to manage GRE interfaces.

1. To start `fp-cli`, enter:

```
$ fp-cli
```

Displaying GRE interfaces

Synopsis

```
gre [name IFNAME]
```

IFNAME

Name of the GRE interface.

Example

```
<fp-0> gre
gre1-vr0 linked to eth1-vr0 link vrfid: 0
    mode IP
    local: 2.0.0.1 remote: 2.0.0.5
    ttl: 64 tos: 0x02
    iflags: csum key oflags: csum key
    ikey: 0x00000001 (1)
    okey: 0x00000001 (1)
```


Displaying GRE statistics

Synopsis

```
gre-stats [percore] [all]
```

Parameters

percore Display all statistics per core running the fast path.

all Display all statistics (even those that are null).

Example

```
<fp-0> gre-stats all
GREDroppedParseIPv4HeaderFailure:0
GREDroppedParseIPv6HeaderFailure:0
GREDroppedPullupIPv4HeaderFailure:0
GREDroppedMissingChecksum:0
GREDroppedUnexpectedChecksum:0
GREDroppedWrongChecksum:0
GREDroppedInitGreIPv4HeaderFailure:0
GREDroppedInitIPv4HeaderFailure:0
GREDroppedPullupIPv6HeaderFailure:0
GREDroppedInitGreIPv6HeaderFailure:0
GREDroppedInitIPv6HeaderFailure:0
GREExceptionInputUnsupportedProtocol:0
GREExceptionOutputUnsupportedProtocol:0
GREExceptionUnsupportedEtherType:0
GREExceptionUnknownIface:0
GREExceptionIPv4Route:0
GREExceptionIPv6Route:0
GREExceptionIPv4SourceSelectFailed:0
GREInvalidHeader:0
GREProtocolNotSupported:0
```

The output is available in JSON format as well.

Synopsis

```
gre-stats-son
```

```
<fp-0> gre-stats-json
{
  "GREDroppedParseIPv4HeaderFailure": 0,
  "GREDroppedParseIPv6HeaderFailure": 0,
  "GREDroppedPullupIPv4HeaderFailure": 0,
  "GREDroppedPullupIPv6HeaderFailure": 0,
  "GREExceptionUnknownIface": 0,
  "GREDroppedUnexpectedChecksum": 0,
  "GREDroppedWrongChecksum": 0,
  "GREDroppedMissingChecksum": 0,
  "GREExceptionUnsupportedEtherType": 0,
  "GREDroppedInitGreIPv4HeaderFailure": 0,
  "GREDroppedInitIPv4HeaderFailure": 0,
  "GREDroppedInitGreIPv6HeaderFailure": 0,
  "GREDroppedInitIPv6HeaderFailure": 0,
  "GREExceptionInputUnsupportedProtocol": 0,
  "GREExceptionOutputUnsupportedProtocol": 0,
  "GREExceptionIPv4Route": 0,
  "GREInvalidHeader": 0,
  "GREProtocolNotSupported": 0,
  "GREExceptionIPv6Route": 0,
  "GREExceptionIPv4SourceSelectFailed": 0,
}
```

Simple GRE L3 tunnel

We will assume the network topology is the following:

```
            +---------------+                +---------------+
            |    Remote     |                | Gateway  with |
            |    Gateway    |                |   fast path   |
            |               |2.0.0.5         |               |
<-<---------|-ethY   /-ethX-|================|-eth1-\   eth2-|--------->->
  100.2.2.1 |     greZ      |         2.0.0.1|      gre1     | 110.2.2.1
            |  10.10.10.2   |                |   10.10.10.1  |
            |               |                |               |
            +---------------+                +---------------+
```

1. Configure IP addresses and routes:

```
# ip link set eth1 up
# ip addr add 2.0.0.1/24 dev eth1
# ip link set eth2 up
# ip addr add 110.2.2.1/24 dev eth2
```

2. Create a new GRE interface:

```
# ip tunnel add gre1 mode gre local 2.0.0.1 remote 2.0.0.5 dev eth1 key 1 csum
# ip link set gre1 up
# ip addr add 10.10.10.1 peer 10.10.10.2/24 dev gre1
```

3. Create a new route:

```
# ip route add 100.2.2.1 via 10.10.10.2
```

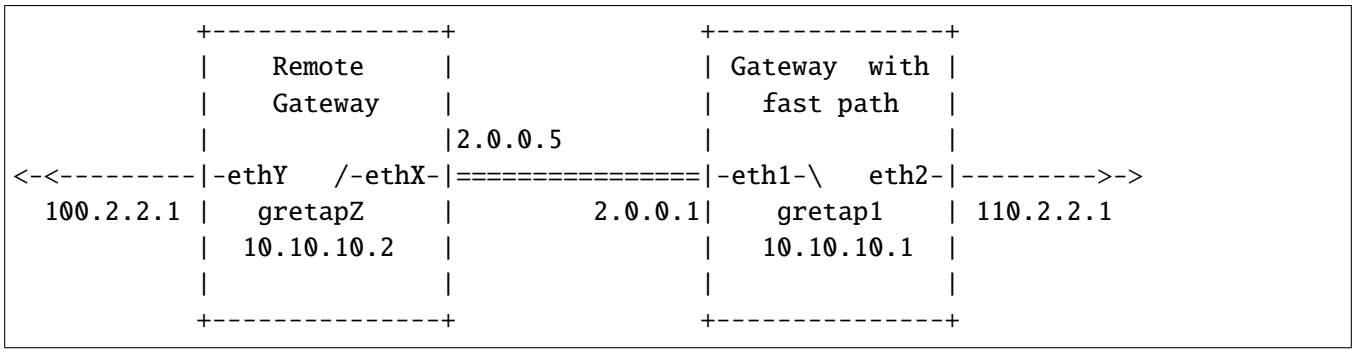
4. Display the characteristics of GRE interfaces on the fast path:

```
$ fp-cli
```

```
<fp-0> gre
gre1-vr0 linked to eth1-vr0 link vrfid: 0
    mode IP
    local: 2.0.0.1 remote: 2.0.0.5
    ttl: inherit tos: 0x00
    iflags: csum key oflags: csum key
    ikey: 0x00000001 (1)
    okey: 0x00000001 (1)
```

Simple GRE L2 tunnel

We will assume the network topology is the following:



1. Configure IP addresses and routes:

```
# ip link set eth1 up
# ip addr add 2.0.0.1/24 dev eth1
# ip link set eth2 up
# ip addr add 110.2.2.1/24 dev eth2
```

2. Create a new GRE interface:

```
# ip link add name gretap1 type gretap local 2.0.0.1 remote 2.0.0.5 dev eth1 key␣
↪1 csum
# ip link set gretap1 up
# ip addr add 10.10.10.1/24 dev gretap1
```

3. Create a new route:

```
# ip route add 100.2.2.1 via 10.10.10.2
```

4. Display the characteristics of GRE interfaces on the fast path:

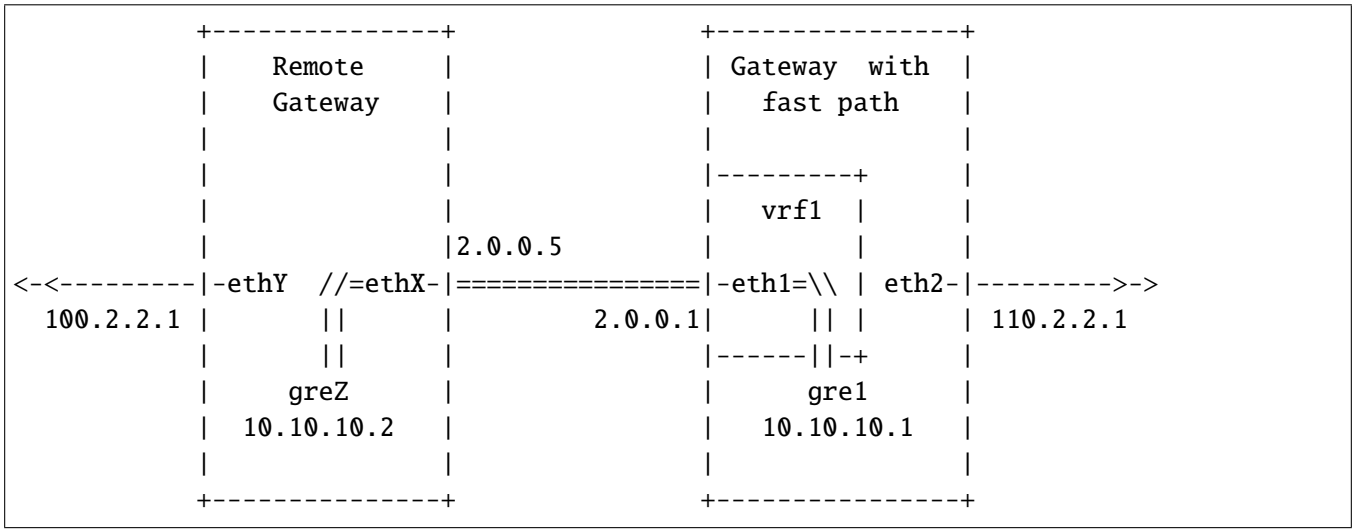
```
$ fp-cli
```

```
<fp-0> gre
gre1-vr0 linked to eth1-vr0 link vrfid: 0
    mode Ether
    local: 2.0.0.1 remote: 2.0.0.5
    ttl: inherit tos: 0x00
    iflags: csum key oflags: csum key
    ikey: 0x00000001 (1)
    okey: 0x00000001 (1)
gre_sys-vr0 linked to ifid0-vr0 link vrfid: 0 master: ovs
    mode Ether
    local: any remote: any
    ttl: inherit tos: 0x00
    iflags: oflags:
```

Note: GRE interfaces displaying 'master: ovs' will be managed by the *Fast Path OVS Acceleration* module. With kernel >= 4.4, a GRE interface will be created to represent Open vSwitch GRE ports.

Simple cross-vrf GRE tunnel

We will assume the network topology is the following:



The `gre1` interface will use 2 VRs:

- The interface `link-vr`, i.e. the VR of encapsulated GRE packets
- The interface `vr`, i.e. the VR of plaintext packets

We will create the GRE interface in the `link-vr` interface, then move it to its own VR.

1. Create the `vrf1` network namespace:

```
# vrfctl add 1 linux-fp-sync-vrf.sh
```

2. Configure IP addresses and routes:

```
# ip link set eth1 netns vrf1
# ip netns exec vrf1 ip link set eth1 up
# ip netns exec vrf1 ip addr add 2.0.0.1/24 dev eth1
# ip netns exec vrf0 ip link set eth2 up
# ip netns exec vrf0 ip addr add 110.2.2.1/24 dev eth2
```

3. Create a new GRE interface:

```
# ip netns exec vrf1 ip link add gre1 type gre local 2.0.0.1 remote 2.0.0.5
# ip netns exec vrf1 ip link set gre1 netns vrf0
# ip netns exec vrf0 ip link set gre1 up
# ip netns exec vrf0 ip addr add 10.10.10.1 peer 10.10.10.2/24 dev gre1
```

4. Create a new route:

```
# ip netns exec vrf0 ip route add 100.2.2.1 via 10.10.10.2
```

5. Display the characteristics of GRE interfaces on the fast path:

```
$ fp-cli
```

```
<fp-0> gre
gre1-vr0 linked to ifid0-vr0 link vrfid: 1
    mode IP
    local: 2.0.0.1 remote: 2.0.0.5
    ttl: inherit tos: 0x00
    iflags: oflags:
```

Providing options

Some capabilities can be tuned for this module.

--ifaces
: Maximum number of GRE interfaces. This GRE interface can be a tunnel or a L2 tunnel over IPv4 or IPv6 (i.e. link type can be gre, gretap, ip6gre or ip6gretap)

  Default value 255

  Memory footprint per GRE interfaces 100 B

  Range 0 .. 5000

Example

```
FP_OPTIONS="--mod-opt=gre:--ifaces=512"
```

Then fast path can manage up to 512 GRE interfaces.

--hash-order-ipv4
: Size order of GRE interfaces for tunnel (L3 or L2) over IPv4 hash table. Value automatically updated if *--ifaces* is changed.

  Default value 4

  Range 1 .. 31

Example

```
FP_OPTIONS="--mod-opt=gre:--hash-order-ipv4=10"
```

--hash-order-ipv6

Size order of GRE interfaces for tunnel (L3 or L2) over IPv6 hash table. Value automatically updated if *--ifaces* is changed.

Default value 4

Range 1 .. 31

Example

```
FP_OPTIONS="--mod-opt=gre:--hash-order-ipv6=10"
```

Tip: To get optimal performance, apply the following ratios to the three parameters:

Parameter	Value
–hash-order-ipv4	`N`
–hash-order-ipv6	`N`
–ifaces	`2 ** N`

Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

---

2.3.10 Fast Path IPsec IPv4

This module requires a Virtual Accelerator IPsec Application License.

Overview

Fast Path IPsec IPv4 provides IPv4 IPsec processing in the fast path.

Features

- FPN-SDK crypto API support to enable crypto processor
- AH (Authentication Header) and ESP (Encapsulating Security Payload) support
- Transport, tunnel and BEET (Bound End-to-End Tunnel) modes support
- Classifier, hash tables to improve IPsec lookup
- Linux stack originated packets handover to fast path IPsec
- Per SA parameter to control copy of DSCP (Differentiated Services Code Point)
- Partial dump of SAD (Security Association Database)
- Anti replay window and output sequence number synchronization (multiple fast paths only)
- Extended Sequence Number (ESN) and large anti-replay window as described by RFCs 4302, 4303 and 4304.
- IPsec 6in4/4in6 tunnel support.
- IPsec nat-t (nat traversal) support.
- Offload of cryptographic operations to idle fast path cores.

Supported algorithms

The following algorithms are supported by the fast path stack using `ip xfrm` commands or during the IKE phase 2:

Encryption algorithm	Generic soft.	Intel Multi Buffer	Intel QAT	Cavium Octeon
NULL	Supported	Supported	Supported	Supported
DES-CBC	Supported	Software fallback	Supported	Supported
3DES-CBC	Supported	Software fallback	Supported	Supported
AES-CBC (128/192/256)	Supported	Supported	Supported	Supported
AES-GCM-128 (128/192/256)	Supported	Supported	Supported	Not supported

Authentication algorithm	Generic soft.	Intel Multi Buffer	Intel QAT	Cavium Octeon
NULL	Supported	Supported	Supported	Supported
HMAC-MD5-96	Supported	Supported	Supported	Supported
HMAC-SHA1-96	Supported	Supported	Supported	Supported
HMAC-SHA2-256-128	Supported	Supported	Supported	Supported
HMAC-SHA2-384-192	Supported	Supported	Supported	Supported
HMAC-SHA2-512-256	Supported	Supported	Supported	Supported
AES-XCBC-96	Supported	Supported	Supported	Supported
AES-GMAC-128	Supported	Supported	Supported	Not supported

Dependencies

6WINDGate modules

- *Fast Path Forwarding IPv4*

Linux

- Control of DSCP copy in tunnel header is a kernel patch (upstream 3.10).

  Without this patch, DSCP is always copied to the outer header.

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=a947b0a93efa
- Partial dump of Linux SA database is a kernel patch.

  Without this patch, the whole SAD is dumped.

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=d3623099d350
- ESN and large anti-replay window support are available since kernel version 2.6.39.

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=2cd084678fc1eb75aec4f7ae3d339d232c00ec61

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=d8647b79c3b7e223ac051439d165bc8e7bbb832f
- ESN and large anti-replay window static configuration is an iproute2 patch.

  https://git.kernel.org/pub/scm/network/iproute2/iproute2.git/commit/?id=0151b56d1029
- IPsec output delegation is available since kernel version 4.20

  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=02b408fae3d5

Usage

Before you begin

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

There is no runtime configuration for IPsec.

When using *Linux - Fast Path Synchronization*, the `linux-fp-sync.sh` startup script executes the following commands to enable IPsec offload to fast path for packets issued by the Linux stack, using filter rules:

```
# fptun-ipsec-ctrl start
```

The status can be checked with the following command:

```
# fptun-ipsec-ctrl status
table inet fptun {
     chain postrouting {
             rt ipsec exists counter pkts 0 bytes 0 queue num 20036
             oifkind "vti" counter pkts 0 bytes 0 queue num 20036
             oifkind "vti6" counter pkts 0 bytes 0 queue num 20036
             oifkind "xfrmi" counter pkts 0 bytes 0 queue num 20036
     }
}
```

By default, one queue is configured. It is possible to load balance traffic to several queues. One instance of `fptun-nfqd` runs per queue. This can be configured with the `FPTUN_IPSEC_NB_QUEUE` option. Note that the service must be stopped before updating this option.

Example

```
# fptun-ipsec.sh stop
Try to gently kill process fptun-nfqd-ipsec-qnum-20036
fptun-nfqd ipsec stopped
# ip --all netns exec fptun-ipsec-ctrl stop

netns: vrf0
# $EDITOR /etc/fptun-ipsec.env #set FPTUN_IPSEC_NB_QUEUE to 4
# fptun-ipsec.sh start
fptun-nfqd ipsec started
# ip --all netns exec fptun-ipsec-ctrl start 4

netns: vrf0
# fptun-ipsec-ctrl status
table inet fptun {
        chain ipsec-output-delegation {
                rt ipsec exists counter pkts 0 bytes 0 queue num 20036-20039
                oifkind "vti" counter pkts 0 bytes 0 queue num 20036-20039
                oifkind "vti6" counter pkts 0 bytes 0 queue num 20036-20039
                oifkind "xfrm" counter pkts 0 bytes 0 queue num 20036-20039
        }
}
```

You must add the netfilter rules to all VRF instances you create afterwards, typically by invoking the `linux-fp-sync-vrf.sh` script:

```
# vrfctl add 1 linux-fp-sync-vrf.sh
```

Configuration example

1. Configure network interfaces

```
# ip link set eth1 up
# ip link set eth3 up
# ip addr add 10.22.4.104/24 dev eth1
# ip addr add 10.23.4.104/24 dev eth3
# ip route add default via 10.23.4.204
```

2. Configure SAs (Security Associations)

```
# ip xfrm state flush
# ip xfrm state add src 10.23.4.104 dst 10.23.4.204 proto esp spi 4096 \
  enc "cbc(aes)" 0xa69e295e5a7718450b9531cac8b73a5d \
  auth "hmac(sha1)" 0xde86d704f3227d67e59f1cdd659d3887f90690d8 \
  mode tunnel
# ip xfrm state add src 10.23.4.204 dst 10.23.4.104 proto esp spi 4352 \
  enc "cbc(aes)" 0x68f199b8d3f753a807385b3f8ab21a58 \
  auth "hmac(sha1)" 0x487bf1117963a795e1bf6f3a37c7289375679c7c \
  mode tunnel
```

3. Configure SPs (Security Policies)

```
# ip xfrm policy flush
# ip xfrm policy add src 10.22.4.0/24 dst 10.24.4.10/24 dir out priority 2000 \
  tmpl src 10.23.4.104 dst 10.23.4.204 proto esp mode tunnel
# ip xfrm policy add src 10.24.4.0/24 dst 10.22.4.10/24 dir in  priority 2000 \
  tmpl src 10.23.4.204 dst 10.23.4.104 proto esp mode tunnel
# ip xfrm policy add src 10.24.4.0/24 dst 10.22.4.10/24 dir fwd priority 2000 \
  tmpl src 10.23.4.204 dst 10.23.4.104 proto esp mode tunnel
```

4. Display SAs in the Linux kernel

```
# ip xfrm state
src 10.23.4.204 dst 10.23.4.104
        proto esp spi 0x00001100 reqid 0 mode tunnel
        replay-window 0
        auth-trunc hmac(sha1) 0x487bf1117963a795e1bf6f3a37c7289375679c7c 96
        enc cbc(aes) 0x68f199b8d3f753a807385b3f8ab21a58
        anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
        sel src 0.0.0.0/0 dst 0.0.0.0/0
src 10.23.4.104 dst 10.23.4.204
        proto esp spi 0x00001000 reqid 0 mode tunnel
        replay-window 0
        auth-trunc hmac(sha1) 0xde86d704f3227d67e59f1cdd659d3887f90690d8 96
```

(continues on next page)

(continued from previous page)

```
        enc cbc(aes) 0xa69e295e5a7718450b9531cac8b73a5d
        anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
        sel src 0.0.0.0/0 dst 0.0.0.0/0
```

5. Display SPs in the Linux kernel

```
# ip xfrm policy
src 10.24.4.0/24 dst 10.22.4.10/24
        dir fwd priority 2000
        tmpl src 10.23.4.204 dst 10.23.4.104
                proto esp reqid 0 mode tunnel
src 10.24.4.0/24 dst 10.22.4.10/24
        dir in priority 2000
        tmpl src 10.23.4.204 dst 10.23.4.104
                proto esp reqid 0 mode tunnel
src 10.22.4.0/24 dst 10.24.4.10/24
        dir out priority 2000
        tmpl src 10.23.4.104 dst 10.23.4.204
                proto esp reqid 0 mode tunnel
```

6. Display SAs in the fast path

```
# fp-cli
```

```
<fp-0> ipsec4-sad all
SAD 2 SA.
1: 10.23.4.104 - 10.23.4.204 vr0 spi 0x1000 ESP tunnel
     counter 1 (genid 1)
     AES-CBC HMAC-SHA1
     key enc:a69e295e5a7718450b9531cac8b73a5d
     digest length: 12
     key auth:de86d704f3227d67e59f1cdd659d3887f90690d8
     sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
     sa_replay_errors=0 sa_selector_errors=0
     replay width=0 seq=0x0 - oseq=0x0
2: 10.23.4.204 - 10.23.4.104 vr0 spi 0x1100 ESP tunnel
     counter 1 (genid 2)
     AES-CBC HMAC-SHA1
     key enc:68f199b8d3f753a807385b3f8ab21a58
     digest length: 12
     key auth:487bf1117963a795e1bf6f3a37c7289375679c7c
     sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
     sa_replay_errors=0 sa_selector_errors=0
     replay width=0 seq=0x0 - oseq=0x0
```


7. Display SPs in the fast path

```
<fp-0> ipsec4-spd all
Inbound SPD: 1 rules
1: 10.24.4.0/24 10.22.4.10/24 proto any vr0 protect prio 2000
     ESP tunnel 10.23.4.204 - 10.23.4.104
     sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
Outbound SPD: 1 rules
1: 10.22.4.0/24 10.24.4.10/24 proto any vr0 protect prio 2000
     cached-SA 0 (genid 0)
     ESP tunnel 10.23.4.104 - 10.23.4.204
     sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
```

Displaying IPsec SPD trie thresholds

IPv4 unhashed SPs are stored in a linked list and may also be stored in an optimized lookup structure, the IPsec trie. The lookup through these policies is done either through the linked list or through the IPsec trie, depending on thresholds.

Synopsis

```
ipsec4-trie-threshold
```

Example

```
<fp-0> ipsec4-trie-threshold
IPsec output trie threshold: 49 2048
IPsec input trie threshold: 49 2048
```

Setting IPsec SPD trie thresholds

For a given direction (in or out), if the number of unhashed IPsec policies (nb) is between the 2 thresholds (min <= nb <= max), then the trie is built and used for SPD lookup.

If nb < min or nb > max, then the linked list is used for SPD lookup.

Synopsis

```
ipsec4-trie-threshold-set [in|out] <thresh min> [<thresh max>]
```

in or out Direction of the SPs.

<thresh min> Minimum unhashed policies to build and use the IPsec trie (0..4095).

<thresh max> Maximum unhashed policies to build and use the IPsec trie (0..4095).

Example

```
<fp-0> ipsec4-trie-threshold-set 50 1500
IPsec output trie threshold: 50 1500
IPsec input trie threshold: 50 1500
```

Statistics

Displaying the SPD

Synopsis

```
ipsec4-spd [(all [(svti|xfrmi) <iface>])|raw]
```

No parameter Only display the number of global SPs.

all Display all global SPs registered in the fast path in order of priority.

svti <ifname> Specific SVTI interface name.

xfrmi <ifname> Specific XFRM interface name.

raw Display all SPs registered in the fast path in the same order as in the internal table.

Examples

```
<fp-0> ipsec4-spd
Inbound SPD: 2 rules
Outbound SPD: 2 rules
```

```
<fp-0> ipsec4-spd all
SPD hash lookup min prefix lengths: local=0, remote=0
Inbound SPD: 2 rules
```

(continues on next page)

(continued from previous page)

```
3: 10.24.4.119/32 10.22.4.118/32 proto 17 vr0 protect prio 2000
    link-vr0
    ESP tunnel 10.23.4.204 - 10.23.4.104 reqid=1
    sp_packets=4 sp_bytes=2112 sp_exceptions=0 sp_errors=0
2: 10.24.4.119/32 10.22.4.118/32 proto 6 vr0 protect prio 2000
    link-vr0
    ESP tunnel 10.23.4.204 - 10.23.4.104 reqid=2
    sp_packets=1 sp_bytes=40 sp_exceptions=0 sp_errors=0
Outbound SPD: 2 rules
3: 10.22.4.118/32 10.24.4.119/32 proto 17 vr0 protect prio 2000
    link-vr0 cached-SA 3 genid 17
    ESP tunnel 10.23.4.104 - 10.23.4.204 reqid=1
    sp_packets=6 sp_bytes=3168 sp_exceptions=1 sp_errors=1
2: 10.22.4.118/32 10.24.4.119/32 proto 6 vr0 protect prio 2000
    link-vr0 cached-SA 2 genid 20
    ESP tunnel 10.23.4.104 - 10.23.4.204 reqid=2
    sp_packets=1 sp_bytes=60 sp_exceptions=1 sp_errors=0
```

```
<fp-0> ipsec4-spd raw
SPD hash lookup min prefix lengths: local=0, remote=0
Inbound SPD: 2 total rules, 2 global rules
2: 10.24.4.119/32 10.22.4.118/32 proto 6 vr0 protect prio 2000
    link-vr0
    ESP tunnel 10.23.4.204 - 10.23.4.104 reqid=2
    sp_packets=1 sp_bytes=40 sp_exceptions=0 sp_errors=0
3: 10.24.4.119/32 10.22.4.118/32 proto 17 vr0 protect prio 2000
    link-vr0
    ESP tunnel 10.23.4.204 - 10.23.4.104 reqid=1
    sp_packets=4 sp_bytes=2112 sp_exceptions=0 sp_errors=0
Outbound SPD: 2 total rules, 2 global rules
2: 10.22.4.118/32 10.24.4.119/32 proto 6 vr0 protect prio 2000
    link-vr0 cached-SA 2 genid 20
    ESP tunnel 10.23.4.104 - 10.23.4.204 reqid=2
    sp_packets=1 sp_bytes=60 sp_exceptions=1 sp_errors=0
3: 10.22.4.118/32 10.24.4.119/32 proto 17 vr0 protect prio 2000
    link-vr0 cached-SA 3 genid 17
    ESP tunnel 10.23.4.104 - 10.23.4.204 reqid=1
    sp_packets=6 sp_bytes=3168 sp_exceptions=1 sp_errors=1
```

Displaying the SAD

Dump all SAs, or only a specific one.

Synopsis

```
ipsec4-sad [all] [(svti|xfrmi) IFNAME] [<src> <prefix> <dst> <prefix> <proto>]
```

No parameters Only display the number of SAs present in the fast path table.

all Display all SAs present in the fast path table.

svti <ifname> Specific SVTI interface name.

xfrmi <ifname> Specific XFRM interface name.

<src> SA source ip address.

<prefix> Length (in bits) of the source ip netmask prefix.

<dst> SA destination ip address.

<prefix> Length (in bits) of the destination ip netmask prefix.

<proto> Select the AH or the ESP protocol.

Examples

```
<fp-0> ipsec4-sad
SAD 4 SA.
```

```
<fp-0> ipsec4-sad all
SAD 4 SA.
2: 10.23.4.104 - 10.23.4.204 vr0 spi 0xc55d891 ESP tunnel
    x-vr0 reqid=2 genid 20 cached-SP 0
    DES-CBC HMAC-MD5
    key enc:4efe5a2ab00d7273
    key auth:8ae2e379f5d9950f9e16c5b5cb95496e
    sa_packets=1 sa_bytes=60 sa_auth_errors=0 sa_decrypt_errors=0
    sa_replay_errors=0 sa_selector_errors=0
    replay check is on width=32 seq=0 bitmap=0x00000000 - oseq=1
3: 10.23.4.104 - 10.23.4.204 vr0 spi 0xe4dbd1 ESP tunnel
    x-vr0 reqid=1 genid 17 cached-SP 0
    DES-CBC HMAC-MD5
    key enc:d6ce2c08b3ed0340
    key auth:33da206f897fd17a214052eb07b403bc
```

(continues on next page)

(continued from previous page)

```
    sa_packets=6 sa_bytes=3168 sa_auth_errors=0 sa_decrypt_errors=0
    sa_replay_errors=0 sa_selector_errors=0
    replay check is on width=32 seq=0 bitmap=0x00000000 - oseq=6
4: 10.23.4.204 - 10.23.4.104 vr0 spi 0x4d2eba4 ESP tunnel
    x-vr0 reqid=2 genid 19 cached-SP 2
    DES-CBC HMAC-MD5
    key enc:becc9cfbed4123cc
    key auth:40a6314fc26317d499389654b0ee670f
    sa_packets=1 sa_bytes=96 sa_auth_errors=0 sa_decrypt_errors=0
    sa_replay_errors=0 sa_selector_errors=0
    replay check is on width=32 seq=1 bitmap=0x00000001 - oseq=0
6: 10.23.4.204 - 10.23.4.104 vr0 spi 0x778e36c ESP tunnel
    x-vr0 reqid=1 genid 16 cached-SP 3
    DES-CBC HMAC-MD5
    key enc:c080b354f2f0d217
    key auth:efd3f0cbc5ade56761385eaa74bbbcb9
    sa_packets=4 sa_bytes=2336 sa_auth_errors=0 sa_decrypt_errors=0
    sa_replay_errors=0 sa_selector_errors=0
    replay check is on width=32 seq=4 bitmap=0x0000000f - oseq=0
```

Extended Sequence Number

Note: This feature needs the ESN (Extended Sequence Number) and large anti-replay window static configuration iproute2 patch.

AH/ESP headers support extended, 64 bit sequence numbers to detect replay.

A single IPsec SA can transfer a maximum of 2^64 IPsec packets.

Example

1. Create an SA with ESN support and a 128 packets replay window:

```
$ ip xfrm state add src 2.1.0.1 dst 2.1.0.5 spi 0x00000220 proto esp reqid 22␣
↪mode tunnel \
enc aes cle1goldorakgoldorakcle1 auth sha1 cle1goldorakgoldcle1 flag esn replay-
↪window 128
```

2. Check that your configuration is correctly synchronized in the fast path:

```
$ fp-cli
```

```
<fp-0> ipsec4-sad all
SAD 1 SA.
1: 2.1.0.1 - 2.1.0.5 vr0 spi 0x220 ESP tunnel
  x-vr0 reqid=22 counter 1 cached-SP 0 (genid 1)
  cached-svti 0 (genid 0)
  AES-CBC HMAC-SHA1 esn
  key enc:636c6531676f6c646f72616b676f6c646f72616b636c6531
  digest length: 12
  key auth:636c6531676f6c646f72616b676f6c64636c6531
  sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
  sa_replay_errors=0 sa_selector_errors=0
  replay width=128 seq=0x0 - oseq=0x0
  00000000 00000000 00000000 00000000
```

See also:

Dependencies

Large anti-replay window

Note: This feature needs the ESN and large anti-replay window static configuration iproute2 patch.

You can set the anti-replay window size between 32 and 4096 packets (maximum size allowed by the Linux kernel).

Example

1. Create an SA with a 256 packets replay window:

```
$ ip xfrm state add src 2.1.0.1 dst 2.1.0.5 spi 0x00000220 proto esp reqid 22␣
↪mode tunnel \
enc aes cle1goldorakgoldorakcle1 auth sha1 cle1goldorakgoldcle1 replay-window 256
```

2. Check that your configuration is correctly synchronized in the fast path:

```
$ fp-cli
```

```
<fp-0> ipsec4-sad all
  SAD 1 SA.
  1: 2.1.0.1 - 2.1.0.5 vr0 spi 0x220 ESP tunnel
  x-vr0 reqid=22 counter 2 cached-SP 0 (genid 2)
  cached-svti 0 (genid 0)
  AES-CBC HMAC-SHA1
```

(continues on next page)

(continued from previous page)

```
key enc:636c6531676f6c646f72616b676f6c646f72616b636c6531
digest length: 12
key auth:636c6531676f6c646f72616b676f6c64636c6531
sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
sa_replay_errors=0 sa_selector_errors=0
replay width=256 seq=0x0 - oseq=0x0
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

See also:

Dependencies

IPv4 in IPv6 IPsec tunnel example

We will encapsulate IPv4 packets in a static IPv6 IPsec tunnel.

1. Create the inbound IPsec configuration:

   a. Create an IPv6 IPsec SA for IPv4 packets:

   ```
   $ ip xfrm state add src 3ffe:2:11::5 dst 3ffe:2:11::1 \
     proto esp spi 0x00000221 mode tunnel \
     enc aes cle1goldorakgoldorakcle1 auth sha1 cle1goldorakgoldcle1 \
     flag af-unspec
   ```

   b. Create an inbound IPv4 IPsec SP:

   ```
   $ ip xfrm policy add src 110.2.2.1/32 dst 100.2.2.1/32 dir in \
     tmpl src 3ffe:2:11::5 dst 3ffe:2:11::1 proto esp mode tunnel
   ```

   c. Create a forward IPv4 IPsec SP:

   ```
   $ ip xfrm policy add src 110.2.2.1/32 dst 100.2.2.1/32 dir fwd \
     tmpl src 3ffe:2:11::5 dst 3ffe:2:11::1 proto esp mode tunnel
   ```

2. Create the outbound IPsec configuration:

   a. Create an IPv6 IPsec SA for IPv4 packets:

   ```
   $ ip xfrm state add src 3ffe:2:11::1 dst 3ffe:2:11::5 \
     proto esp spi 0x00000220 mode tunnel \
     enc aes cle1goldorakgoldorakcle2 auth sha1 cle1goldorakgoldcle2 \
     flag af-unspec
   ```

   b. Create an outbound IPv4 IPsec SP:

```
$ ip xfrm policy add src 100.2.2.1/32 dst 110.2.2.1/32 dir out \
  tmpl src 3ffe:2:11::1 dst 3ffe:2:11::5 proto esp mode tunnel
```

3. Check that your configuration is correctly synchronized in the fast path:

   a. Start `fp-cli`:

```
$ fp-cli
```

   b. Display the SAs in the *Fast Path IPsec IPv6* table:

```
<fp-0> ipsec6-sad all
IPv6 SAD 2 SA.
1: 3ffe:2:11::5 - 3ffe:2:11::1 vr0 spi 0x221 ESP tunnel
  x-vr0 counter 1 genid 1 cached-SP: 0
  AES-CBC HMAC-SHA1
  key enc:636c6531676f6c646f72616b676f6c646f72616b636c6531
  digest length: 12
  key auth:636c6531676f6c646f72616b676f6c64636c6531
  sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
  sa_replay_errors=0 sa_selector_errors=0
  replay width=0 seq=0x0 - oseq=0x0
2: 3ffe:2:11::1 - 3ffe:2:11::5 vr0 spi 0x220 ESP tunnel
  x-vr0 counter 1 genid 2 cached-SP: 0
  AES-CBC HMAC-SHA1
  key enc:636c6531676f6c646f72616b676f6c646f72616b636c6532
  digest length: 12
  key auth:636c6531676f6c646f72616b676f6c64636c6532
  sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
  sa_replay_errors=0 sa_selector_errors=0
  replay width=0 seq=0x0 - oseq=0x0
```

   c. Display the SPs in the Fast Path IPsec IPv4 table:

```
<fp-0> ipsec4-spd all
SPD hash lookup min prefix lengths: local=0, remote=0
Inbound SPD: 1 rules
1: 110.2.2.1/32 100.2.2.1/32 proto any vr0 protect prio 0
  link-vr0
  ESP tunnel 3ffe:2:11::5 - 3ffe:2:11::1
  sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
Outbound SPD: 1 rules
1: 100.2.2.1/32 110.2.2.1/32 proto any vr0 protect prio 0
  link-vr0 cached-SA 0 (genid 0)
  ESP tunnel 3ffe:2:11::1 - 3ffe:2:11::5
  sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
```

See also:

To dynamically configure IPsec tunnels, see the *Control Plane Security - IKEv1 and IKEv2* documentation.

RFC compliance for HMAC-SHA2 cryptographic algorithms

HMAC-SHA256 truncation length

The standard HMAC-SHA256 truncation length in IPsec is 128 bits.

However, by default, the Linux kernel sets the truncation length to 96 bits (HMAC-SHA256-96). The fast path ignores the truncation size configured in the Linux kernel and always assumes that it is 128 bits.

You can manually configure RFC (Request For Comment) compliant HMAC-SHA256 IPsec SAs *via* the following `iproute2` command:

```
# ip xfrm state add src 10.23.4.104 dst 10.23.4.204 proto esp spi 5246 \
  enc "cbc(aes)" 0xa69e295e5a7718450b9531cac8b73a5d \
  auth-trunc "hmac(sha256)" \
  0x3dbce667c4c31fa24d88d9b7d64a16d415c78f469b0ae7f734803d2ec1bbe844 128
```

instead of:

```
# ip xfrm state add src 10.23.4.104 dst 10.23.4.204 proto esp spi 5246 \
  enc "cbc(aes)" 0xa69e295e5a7718450b9531cac8b73a5d \
  auth "hmac(sha256)" \
  0x3dbce667c4c31fa24d88d9b7d64a16d415c78f469b0ae7f734803d2ec1bbe844
```

See also:

RFC 4868 (https://tools.ietf.org/html/rfc4868.html) Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec, section 2.6.

HMAC-SHA2 algorithms and IPv4 AH

The AH header ICV field may include explicit padding if required to ensure that the AH header is a multiple of 32 bits (IPv4) or 64 bits (IPv6).

However, the Linux kernel always ensures that the AH header is a multiple of 64 bits, regardless of the IP version. If the output of the selected algorithm is 96 bits (e.g. HMAC-SHA1-96) or 192 bits (e.g. HMAC-SHA384-192), aligning the AH header on 32 or 64 bits is the same, but if the output of the selected algorithm is 128 bits (HMAC-SHA256-128) or 256 bits (HMAC-SHA512-256), Linux pads IPv4 AH headers although it MUST not.

The fast path always performs the padding as specified in the AH RFC.

You can manually configure RFC compliant IPsec IPv4 SAs by setting the flag `align4`, at least for HMAC-SHA256 and HMAC-SHA512 algorithms. However, a good practice is to always set this flag on IPsec IPv4 SAs, whatever the authentication algorithm.

Example

```
# ip xfrm state add src 10.23.4.104 dst 10.23.4.204 proto ah \
  spi 0x00127764 mode tunnel auth-trunc "hmac(sha256)" \
  0x8dcce7d80f7c8bb81e6a526b9d5d7ce2e7a474e3406c40953108b6d92b61cb77 128 \
  flag align4
```

See also:

RFC 4302 (https://tools.ietf.org/html/rfc4302.html) IP Authentication Header, section 3.3.3.2.1.

IPsec NAT-T example

We will encapsulate the IPsec packet in a IPv4 UDP payload, and NAT will be done on the same device.

Example

1. NAT configuration

   ```
   # iptables -t nat -A POSTROUTING -o ntfp2 -s 10.200.0.2/0 -j MASQUERADE
   ```

2. Configure *Control Plane Security - IKEv1 and IKEv2*

   ```
   # cat /etc/ike/ipsec.conf
   config setup

   conn %default
       keyexchange=ikev2
       keyingtries=1
       mobike=no
       ikelifetime=57600s
       rekeymargin=5760s
       keylife=28800s

   conn nat-t
       auto=route
       left=10.200.0.2
       right=10.125.0.1
       leftid=10.125.0.2
       rightid=10.125.0.1
       type=tunnel
       leftsubnet=10.200.0.0/24
       rightsubnet=10.100.0.0/24
       authby=psk
   ```

(continues on next page)

(continued from previous page)

```
    ike=aes128-sha1-modp2048
    esp=aes128-sha1-modp2048
```

The IPsec IPv4 SAs and SPs are generated automatically by *Control Plane Security - IKEv1 and IKEv2*, see *Control Plane Security - IKEv1 and IKEv2* documentation about how to dynamically configure IPsec tunnels.

1. Display SPs in the Linux kernel

```
# ip xfrm policy
src 10.100.0.0/24 dst 10.200.0.0/24
        dir fwd priority 287712
        tmpl src 10.125.0.1 dst 10.200.0.2
                proto esp reqid 1 mode tunnel
src 10.100.0.0/24 dst 10.200.0.0/24
        dir in priority 287712
        tmpl src 10.125.0.1 dst 10.200.0.2
                proto esp reqid 1 mode tunnel
src 10.200.0.0/24 dst 10.100.0.0/24
        dir out priority 287712
        tmpl src 10.200.0.2 dst 10.125.0.1
                proto esp reqid 1 mode tunnel
src 0.0.0.0/0 dst 0.0.0.0/0
        socket in priority 0
src 0.0.0.0/0 dst 0.0.0.0/0
        socket out priority 0
src 0.0.0.0/0 dst 0.0.0.0/0
        socket in priority 0
src 0.0.0.0/0 dst 0.0.0.0/0
        socket out priority 0
src ::/0 dst ::/0
        socket in priority 0
src ::/0 dst ::/0
        socket out priority 0
src ::/0 dst ::/0
        socket in priority 0
src ::/0 dst ::/0
        socket out priority 0
```

2. trigger IKE negotiations

3. Display SAs in the Linux kernel

```
# ip xfrm state
src 10.200.0.2 dst 10.125.0.1
        proto esp spi 0xcacf0d61 reqid 1 mode tunnel
```

(continues on next page)

(continued from previous page)

```
        replay-window 0 flag af-unspec
        auth-trunc hmac(sha1) 0xe65464fd16c7b8d234bca701e9a52fd7c8ba1b57 96
        enc cbc(aes) 0xf1ffcdd8173204a1cde5ca0e55b123ce
        encap type espinudp sport 4500 dport 4500 addr 0.0.0.0
        anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
src 10.125.0.1 dst 10.200.0.2
        proto esp spi 0xc5b51de0 reqid 1 mode tunnel
        replay-window 32 flag af-unspec
        auth-trunc hmac(sha1) 0xed1d8b8a30844365dbf9657041da76d725842cc3 96
        enc cbc(aes) 0xac1d42cb032259724a14f97441b309a3
        encap type espinudp sport 4500 dport 4500 addr 0.0.0.0
        anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
```

4. Check that the SAs is correctly synchronized in the fast path:

   a. Start `fp-cli`:

   ```
   # fp-cli
   ```

   b. Display the SAs in the Fast Path IPsec IPv4 table:

   ```
   <fp-0> ipsec4-sad all
   SAD 2 SA.
   2: 10.125.0.1 - 10.200.0.2 vr0 spi 0xc5b51de0 ESP tunnel
        reqid=1 counter 1 (genid 2)
        AES-CBC HMAC-SHA1
        key enc:ac1d42cb032259724a14f97441b309a3
        digest length: 12
        key auth:ed1d8b8a30844365dbf9657041da76d725842cc3
        NAT traversal: sport: 4500 dport: 4500
        sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
        sa_replay_errors=0 sa_selector_errors=0
        replay width=32 seq=0x0 - oseq=0x0
        00000000
   3: 10.200.0.2 - 10.125.0.1 vr0 spi 0xcacf0d61 ESP tunnel
        reqid=1 counter 1 (genid 3)
        AES-CBC HMAC-SHA1
        key enc:f1ffcdd8173204a1cde5ca0e55b123ce
        digest length: 12
        key auth:e65464fd16c7b8d234bca701e9a52fd7c8ba1b57
        NAT traversal: sport: 4500 dport: 4500
        sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
        sa_replay_errors=0 sa_selector_errors=0
        replay width=0 seq=0x0 - oseq=0x0
   ```


AES-GCM and AES-GMAC cryptographic algorithms

AES-GCM and AES-GMAC (Advanced Encryption Standard - Galois Message Authentication Code) are AEAD (Authenticated Encryption with Associated Data) cryptographic algorithms. AES-GCM provides confidentiality and data origin authentication, while AES-GMAC only provides data origin authentication.

AES-GCM

The configuration parameters of the AES-GCM algorithm are:

- the AES (Advanced Encryption Standard) key (128, 192 or 256 bits)
- the salt (32 bits), prepended to the packet IV (Initialization Vector) to form a Nonce
- the ICV (Integrity Check Value) length. Only the mandatory size of 128 bits is supported by the fast path IPsec implementation.

When configuring an SA via `iproute2`, the salt is appended to the AES key. Since AES-GCM provides both confidentiality and authentication, no additional authentication algorithm must be specified.

Example

1. Create an ESP SA with algorithm AES-GCM and a 128-bit key, a 32-bit salt and 128-bit ICV:

```
# ip xfrm state add src 10.23.4.104 dst 10.23.4.204 proto esp spi 4096 \
  aead "rfc4106(gcm(aes))" 0x531bf5714440d166757353f29511ca42146e8b0c 128 \
  mode tunnel
```

2. Display the SA in the fast path:

```
<fp-0> ipsec4-sad all
SAD 1 SA.
1: 10.23.4.104 - 10.23.4.204 vr0 spi 0x1000 ESP tunnel
     counter 1 (genid 1)
     AES-GCM
     key enc:531bf5714440d166757353f29511ca42
     nonce salt:146e8b0c
     digest length: 16
     sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
     sa_replay_errors=0 sa_selector_errors=0
     replay width=0 seq=0x0 - oseq=0x0
```

Warning: The fast path IPsec implementation only supports a full-length 128-bit ICV, the optional lengths of 64 or 96 bits are not supported. The fast path ignores the ICV length specified in Linux and will only send and accept IPsec packets protected by AES-GCM with a 128-bit ICV.

Note: If you use the multibuffer crypto library, AES-GCM is accelerated for 128 bit, 192 or 256 bit keys.

See also:

RFC 4106 (https://tools.ietf.org/html/rfc4106.html) The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)

AES-GMAC

AES-GMAC is a special case of AES-GCM, designed to only provide data origin authentication, typically to implement authenticated ESP-null.

Like AES-GCM, the configuration parameters of the AES-GMAC algorithm are:

- the AES key (128, 192 or 256 bits)
- the salt (32 bits), prepended to the packet IV to form a Nonce

Only the full-length 128 bit ICV is authorized by the standard.

When configuring an SA via `iproute2`, the salt is appended to the AES key.

Example

1. Create an ESP SA with algorithm AES-GMAC and a 128-bit key, a 32-bit salt and 128-bit ICV:

```
# ip xfrm state add src 10.23.4.104 dst 10.23.4.204 proto esp spi 4097 \
  aead "rfc4543(gcm(aes))" 0x40d3a54c5ae9ee8f23f73729975a3db58eb5cdbb 128 \
  mode tunnel
```

2. Display the SA in the fast path:

```
<fp-0> ipsec4-sad all
SAD 1 SA.
1: 10.23.4.104 - 10.23.4.204 vr0 spi 0x1001 ESP tunnel
     counter 2 (genid 3)
     AES-GMAC
     key enc:40d3a54c5ae9ee8f23f73729975a3db5
     nonce salt:8eb5cdbb
     digest length: 16
```

(continues on next page)

(continued from previous page)

```
    sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
    sa_replay_errors=0 sa_selector_errors=0
    replay width=0 seq=0x0 - oseq=0x0
```

Note: Linux only supports AES-GMAC for the ESP algorithm, not for the AH algorithm.

Note: If you use the multibuffer crypto library, AES-GMAC is accelerated for 128 bit, 192 or 256 bit keys.

See also:

RFC 4543 (https://tools.ietf.org/html/rfc4543.html) The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH

Offload of cryptographic operations

Maximal throughput of a tunnel is limited. In order to increase this limit cryptographic operations are offloaded to idle fast path cores. Gain provided by this feature depends of the traffic model and the number of idle fast path cores. Typically the maximal throughput of a tunnel with IMIX traffic is double if a fast path core is available to do the cryptographic operations.

Details of the way to configure offloading of the cryptographic operations is detailed in *FPN-SDK Cryptographic offloading*

By default the crytographic offloading is done only for packets received from the tunnel. The cryptographic offloading for packets sent into the tunnel can be enabled through the cli but can impact packet ordering (especially if there are many flows aggregated in the tunnel) and cause issue with the anti-replay windows of the IPsec SA.

Providing options

To change the maximum number of IPsec tunnels, several runtime parameters must be changed. An IPsec tunnel has 2 IPsec SAs, 1 IPsec SP in and 1 IPsec SP out.

The number of IPsec SPs (in and out) is specified by the max-sp parameter.

The number of IPsec SAs is specified by the max-sa parameter. This parameter must be at least the double of the max-sp parameter. Each SA uses a dedicated cryptographic session, and the maximum number of cryptographic sessions must be changed accordingly: the number of cryptographic sessions must be equal or greater than max-sa (see *FPN-SDK cryptography option*).

`--max-sp`
: Maximum number of SPs

  Default value 8192

Memory footprint per IPsec SP 1 KB

Range 0 .. 400K

`--max-sa`
Maximum number of SAs

Default value 8192

Memory footprint per SA 5 KB

This footprint takes into account 1 cryptographic session per SA.

Range 0 .. 400K

`--sa-hash-order`
Size order of IPv4 IPsec SAD hash table. Value automatically updated if `--max-sa` is changed.

Default value 16

Range 16 .. 20

`--sp-hash-order`
Size order of IPv4 IPsec SPD (Security Policy Database) hash table. Value automatically updated if `--max-sp` is changed.

Default value 9

Range 8 .. 16

Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

---

2.3.11 Fast Path IPsec IPv6

This module requires a Virtual Accelerator IPsec Application License.

Overview

Fast Path IPsec IPv6 provides IPv6 IPsec processing in the fast path.

Features

- FPN-SDK crypto API support to enable crypto processor
- AH and ESP support
- Transport, tunnel and BEET modes support
- Classifier, hash tables to improve IPsec lookup
- Linux stack originated packets handover to fast path IPsec
- Per SA parameter to control copy of DSCP
- Partial dump of SAD
- Anti replay window and output sequence number synchronization (multiple fast paths only)
- Extended Sequence Number (ESN) and large anti-replay window as described by RFCs 4302, 4303 and 4304.
- IPsec 6in4/4in6 tunnel support.
- IPsec nat-t (nat traversal) support.
- Offload of cryptographic operations to idle fast path cores.

Supported algorithms

The following algorithms are supported by the fast path stack using `ip xfrm` commands or during the IKE phase 2:

Encryption algorithm	Generic soft.	Intel Multi Buffer	Intel QAT	Cavium Octeon
NULL	Supported	Supported	Supported	Supported
DES-CBC	Supported	Software fallback	Supported	Supported
3DES-CBC	Supported	Software fallback	Supported	Supported
AES-CBC (128/192/256)	Supported	Supported	Supported	Supported
AES-GCM-128 (128/192/256)	Supported	Supported	Supported	Not supported

Authentication algorithm	Generic soft.	Intel Multi Buffer	Intel QAT	Cavium Octeon
NULL	Supported	Supported	Supported	Supported
HMAC-MD5-96	Supported	Supported	Supported	Supported
HMAC-SHA1-96	Supported	Supported	Supported	Supported
HMAC-SHA2-256-128	Supported	Supported	Supported	Supported
HMAC-SHA2-384-192	Supported	Supported	Supported	Supported
HMAC-SHA2-512-256	Supported	Supported	Supported	Supported
AES-XCBC-96	Supported	Supported	Supported	Supported
AES-GMAC-128	Supported	Supported	Supported	Not supported

Dependencies

6WINDGate modules

- *Fast Path IPsec IPv4*
- *Fast Path Forwarding IPv6*

Linux

- Control of DSCP copy in tunnel header is a kernel patch (upstream 3.10).

  Without this patch, DSCP is always copied to the outer header.

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=a947b0a93efa
- Partial dump of Linux SA database is a kernel patch.

  Without this patch, the whole SAD is dumped.

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=d3623099d350
- ESN and large anti-replay window support are available since kernel version 2.6.39.

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=2cd084678fc1eb75aec4f7ae3d339d232c00ec61

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=d8647b79c3b7e223ac051439d165bc8e7bbb832f
- ESN and large anti-replay window static configuration is an iproute2 patch.

  https://git.kernel.org/pub/scm/network/iproute2/iproute2.git/commit/?id=0151b56d1029
- IPsec output delegation is available since kernel version 4.20

  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=02b408fae3d5

Usage

Before you begin

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

There is no runtime configuration for IPsec.

Please see the Fast Path IPsec IPv4 documentation, section *Before you begin* to perform the initial steps.

Configuration example

The following example is relevant to an Ubuntu machine.

```
ip xfrm policy add dir fwd src 3ffe:110:9:9::1/128 dst 3ffe:100:9:9::1/128 index␣
↪0x80000062 priority 2000 tmpl  src 3ffe:9:11::5 dst 3ffe:9:11::1 proto esp reqid 99␣
↪mode tunnel
ip xfrm policy add dir in  src 3ffe:110:9:9::1/128 dst 3ffe:100:9:9::1/128 index␣
↪0x80000058 priority 2000 tmpl  src 3ffe:9:11::5 dst 3ffe:9:11::1 proto esp reqid 99␣
↪mode tunnel
ip xfrm policy add dir out src 3ffe:100:9:9::1/128 dst 3ffe:110:9:9::1/128 index␣
↪0x80000051 priority 2000 tmpl  src 3ffe:9:11::1 dst 3ffe:9:11::5 proto esp reqid 99␣
↪mode tunnel
ip xfrm policy list
src 3ffe:100:9:9::1/128 dst 3ffe:110:9:9::1/128
        dir out priority 2000 ptype main
        tmpl src 3ffe:9:11::1 dst 3ffe:9:11::5
                proto esp reqid 99 mode tunnel
src 3ffe:110:9:9::1/128 dst 3ffe:100:9:9::1/128
        dir in priority 2000 ptype main
        tmpl src 3ffe:9:11::5 dst 3ffe:9:11::1
                proto esp reqid 99 mode tunnel
src 3ffe:110:9:9::1/128 dst 3ffe:100:9:9::1/128
        dir fwd priority 2000 ptype main
        tmpl src 3ffe:9:11::5 dst 3ffe:9:11::1
                proto esp reqid 99 mode tunnel

# fp-cli
```

```
<fp-0> ipsec6-spd all
IPv6 SPD hash lookup min prefix lengths: local=0, remote=0
Inbound SPD: 1 rules
1: 3ffe:110:9:9::1/128 3ffe:100:9:9::1/128 proto any vr0 protect prio 2000
     link-vr0
    ESP tunnel 3ffe:9:11::5 - 3ffe:9:11::1 reqid=99
     sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
Outbound SPD: 1 rules
1: 3ffe:100:9:9::1/128 3ffe:110:9:9::1/128 proto any vr0 protect prio 2000
     link-vr0 cached-SA 0 genid 0
    ESP tunnel 3ffe:9:11::1 - 3ffe:9:11::5 reqid=99
     sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
```

```
ip xfrm state add src 3ffe:9:11::5 dst 3ffe:9:11::1 spi 0x00000991 proto esp reqid 99␣
↪mode tunnel enc "cbc(des)" 0x706f6e672d2d3939
```

(continues on next page)

(continued from previous page)

```
ip xfrm state add src 3ffe:9:11::1 dst 3ffe:9:11::5 spi 0x00000432 proto esp reqid 99␣
↪mode tunnel enc "cbc(des)" 0x1974040657494E44

ip xfrm state list
src 3ffe:9:11::1 dst 3ffe:9:11::5
     proto esp spi 0x00000432 reqid 99 mode tunnel
     replay-window 0
     enc cbc(des) 0x1974040657494e44
     sel src ::/0 dst ::/0
src 3ffe:9:11::5 dst 3ffe:9:11::1
     proto esp spi 0x00000991 reqid 99 mode tunnel
     replay-window 0
     enc cbc(des) 0x706f6e672d2d3939
     sel src ::/0 dst ::/0
```

```
<fp-0> ipsec6-sad all
IPv6 SAD 2 SA.
1: 3ffe:9:11::5 - 3ffe:9:11::1 vr0 spi 0x991 ESP tunnel
      x-vr0 reqid=99 genid 1 cached-SP: 0
      DES-CBC
      key enc:706f6e672d2d3939
      sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
      sa_replay_errors=0 sa_selector_errors=0
      replay check is off width=0 seq=0 bitmap=0x00000000 - oseq=0
2: 3ffe:9:11::1 - 3ffe:9:11::5 vr0 spi 0x432 ESP tunnel
      x-vr0 reqid=99 genid 2 cached-SP: 0
      DES-CBC
      key enc:1974040657494e44
      sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
      sa_replay_errors=0 sa_selector_errors=0
      replay check is off width=0 seq=0 bitmap=0x00000000 - oseq=0
```

Security associations management

Please see the Fast Path IPsec IPv4 documentation: the commands to manage security associations are the same.

Security policies management

Please see the Fast Path IPsec IPv4 documentation: the commands to manage security policies are the same.

Statistics

Displaying the SPD

Synopsis

```
ipsec6-spd [all|raw]
```

No parameter Only display the number of global IPv6 SPs.

all Display all global IPv6 SPs registered in the fast path in order of priority.

raw Display all IPv6 SPs registered in the fast path in the same order as in the internal table.

Examples

```
<fp-0> ipsec6-spd
IPv6 SPD hash lookup min prefix lengths: local=0, remote=0
Inbound SPD: 1 rules
Outbound SPD: 1 rules
```

```
<fp-0> ipsec6-spd all
IPv6 SPD hash lookup min prefix lengths: local=0, remote=0
Inbound SPD: 1 rules
1: 3ffe:110:9:9::1/128 3ffe:100:9:9::1/128 proto any vr0 protect prio 2000
     link-vr0
    ESP tunnel 3ffe:9:11::5 - 3ffe:9:11::1 reqid=99
     sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
Outbound SPD: 1 rules
1: 3ffe:100:9:9::1/128 3ffe:110:9:9::1/128 proto any vr0 protect prio 2000
     link-vr0 cached-SA 0 genid 0
    ESP tunnel 3ffe:9:11::1 - 3ffe:9:11::5 reqid=99
     sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
```

```
<fp-0> ipsec6-spd raw
IPv6 SPD hash lookup min prefix lengths: local=0, remote=0
Inbound SPD: 1 total rules, 1 global rules
1: 3ffe:110:9:9::1/128 3ffe:100:9:9::1/128 proto any vr0 protect prio 2000
     link-vr0
```

(continues on next page)

(continued from previous page)

```
    ESP tunnel 3ffe:9:11::5 - 3ffe:9:11::1 reqid=99
     sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
Outbound SPD: 1 total rules, 1 global rules
1: 3ffe:100:9:9::1/128 3ffe:110:9:9::1/128 proto any vr0 protect prio 2000
     link-vr0 cached-SA 0 genid 0
    ESP tunnel 3ffe:9:11::1 - 3ffe:9:11::5 reqid=99
     sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
```

Displaying the SAD

Dump all SAs, or only a specific one.

Synopsis

```
ipsec6-sad [all] [(svti|xfrmi) <ifname>] [<src> <prefix> <dst> <prefix> <proto>]
```

No parameters Only display the number of IPv6 SAs present in the fast path table.

all Display all IPv6 SAs present in the fast path table.

svti <ifname> Specific SVTI interface name.

xfrmi <ifname> Specific XFRM interface name.

<src> SA source IPv6 address.

<prefix> Length (in bits) of the source IPv6 netmask prefix.

<dst> SA destination IPv6 address.

<prefix> Length (in bits) of the destination IPv6 netmask prefix.

<proto> Select the AH or the ESP protocol.

Examples

```
<fp-0> ipsec6-sad
IPv6 SAD 2 SA.
```

```
<fp-0> ipsec6-sad all
IPv6 SAD 2 SA.
1: 3ffe:9:11::5 - 3ffe:9:11::1 vr0 spi 0x991 ESP tunnel
      x-vr0 reqid=99 genid 1 cached-SP: 0
      DES-CBC
```

(continues on next page)

(continued from previous page)

```
      key enc:706f6e672d2d3939
      sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
      sa_replay_errors=0 sa_selector_errors=0
      replay check is off width=0 seq=0 bitmap=0x00000000 - oseq=0
2: 3ffe:9:11::1 - 3ffe:9:11::5 vr0 spi 0x432 ESP tunnel
      x-vr0 reqid=99 genid 2 cached-SP: 0
      DES-CBC
      key enc:1974040657494e44
      sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
      sa_replay_errors=0 sa_selector_errors=0
      replay check is off width=0 seq=0 bitmap=0x00000000 - oseq=0
```

Extended Sequence Number

Note: This feature needs the ESN and large anti-replay window static configuration iproute2 patch.

AH/ESP headers support extended, 64 bit sequence numbers to detect replay.

A single IPsec SA can transfer a maximum of 2^64 IPsec packets.

Example

1. Create an SA with ESN support and a 128 packets replay window:

```
$ ip xfrm state add src 3ffe:2:11::1 dst 3ffe:2:11::5 spi 0x00000220 proto esp␣
↪reqid 22 mode tunnel \
enc aes cle1goldorakgoldorakcle1 auth sha1 cle1goldorakgoldcle1 flag esn replay-
↪window 128
```

2. Check that your configuration is correctly synchronized in the fast path:

```
$ fp-cli
```

```
<fp-0> ipsec6-sad all
IPv6 SAD 1 SA.
1: 3ffe:2:11::1 - 3ffe:2:11::5 vr0 spi 0x220 ESP tunnel
  x-vr0 reqid=22 counter 1 genid 1 cached-SP: 0
  AES-CBC HMAC-SHA1 esn
  key enc:636c6531676f6c646f72616b676f6c646f72616b636c6531
  digest length: 12
  key auth:636c6531676f6c646f72616b676f6c64636c6531
```

(continues on next page)

(continued from previous page)

```
  sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
  sa_replay_errors=0 sa_selector_errors=0
  replay width=128 seq=0x0 - oseq=0x0
  00000000 00000000 00000000 00000000
```

See also:

Dependencies

Large anti-replay window example

Note: This feature needs the ESN and large anti-replay window static configuration iproute2 patch.

You can set the anti-replay window size between 32 and 4096 packets (maximum size allowed by the Linux kernel).

Example

1. Create an SA with a 256 packets replay window:

```
$ ip xfrm state add src 3ffe:2:11::1 dst 3ffe:2:11::5 spi 0x00000220 proto esp␣
↪reqid 22 mode tunnel \
enc aes cle1goldorakgoldorakcle1 auth sha1 cle1goldorakgoldcle1 replay-window 256
```

2. Check that your configuration is correctly synchronized in the fast path:

```
$ fp-cli
```

```
<fp-0> ipsec6-sad all
IPv6 SAD 1 SA.
1: 3ffe:2:11::1 - 3ffe:2:11::5 vr0 spi 0x220 ESP tunnel
  x-vr0 reqid=22 counter 2 genid 2 cached-SP: 0
  AES-CBC HMAC-SHA1
  key enc:636c6531676f6c646f72616b676f6c646f72616b636c6531
  digest length: 12
  key auth:636c6531676f6c646f72616b676f6c64636c6531
  sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
  sa_replay_errors=0 sa_selector_errors=0
  replay width=256 seq=0x0 - oseq=0x0
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

See also:

Dependencies

IPv6 in IPv4 IPsec tunnel example

We will encapsulate IPv6 packets in a static IPv4 IPsec tunnel.

1. Create the inbound IPsec endpoint:

   a. Create an IPv4 IPsec SA for IPv6 packets:

   ```
   $ ip xfrm state add src 2.1.0.5 dst 2.1.0.1 proto esp spi 0x00000221 mode␣
   ↪tunnel \
   sel src ::/0 enc aes cle1goldorakgoldorakcle1 auth sha1 cle1goldorakgoldcle1
   ```

   b. Create an inbound IPv6 IPsec SP:

   ```
   $ ip xfrm policy add src 3ffe:110:2:2::1/128 dst 3ffe:100:2:2::1/128 dir in \
   tmpl src 2.1.0.5 dst 2.1.0.1 proto esp mode tunnel
   ```

   c. Create a forward IPv6 IPsec SP:

   ```
   $ip xfrm policy add src 3ffe:110:2:2::1/128 dst 3ffe:100:2:2::1/128 dir fwd \
   tmpl src 2.1.0.5 dst 2.1.0.1 proto esp mode tunnel
   ```

2. Create the outbound IPsec endpoint:

   a. Create an IPv4 IPsec SA for IPv4 packets:

   ```
   $ ip xfrm state add src 2.1.0.1 dst 2.1.0.5 proto esp spi 0x00000220 mode␣
   ↪tunnel \
   sel src ::/0 enc aes cle1goldorakgoldorakcle2 auth sha1 cle1goldorakgoldcle2
   ```

   b. Create an outbound IPv4 IPsec SP:

   ```
   $ ip xfrm policy add src 3ffe:100:2:2::1/128 dst 3ffe:110:2:2::1/128 dir out \
   tmpl src 2.1.0.1 dst 2.1.0.5 proto esp mode tunnel
   ```

3. Check that your configuration is correctly synchronized in the fast path:

   a. Start `fp-cli`:

   ```
   $ fp-cli
   ```

   b. Display the SAs in the *Fast Path IPsec IPv4* table:

   ```
   <fp-0> ipsec4-sad all
   SAD 2 SA.
   1: 2.1.0.5 - 2.1.0.1 vr0 spi 0x221 ESP tunnel
     x-vr0 counter 1 cached-SP 0 (genid 1)
     AES-CBC HMAC-SHA1
   ```

(continues on next page)

(continued from previous page)

```
  key enc:636c6531676f6c646f72616b676f6c646f72616b636c6531
  digest length: 12
  key auth:636c6531676f6c646f72616b676f6c64636c6531
  sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
  sa_replay_errors=0 sa_selector_errors=0
  replay width=0 seq=0x0 - oseq=0x0
2: 2.1.0.1 - 2.1.0.5 vr0 spi 0x220 ESP tunnel
  x-vr0 counter 1 cached-SP 0 (genid 2)
  AES-CBC HMAC-SHA1
  key enc:636c6531676f6c646f72616b676f6c646f72616b636c6532
  digest length: 12
  key auth:636c6531676f6c646f72616b676f6c64636c6532
  sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
  sa_replay_errors=0 sa_selector_errors=0
  replay width=0 seq=0x0 - oseq=0x0
```

c. Display the SPs in the Fast Path IPsec IPv6 table:

```
<fp-0> ipsec6-spd all
IPv6 SPD hash lookup min prefix lengths: local=0, remote=0
Inbound SPD: 1 rules
1: 3ffe:110:2:2::1/128 3ffe:100:2:2::1/128 proto any vr0 protect prio 0
  link-vr0
 ESP tunnel 2.1.0.5 - 2.1.0.1
  sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
Outbound SPD: 1 rules
1: 3ffe:100:2:2::1/128 3ffe:110:2:2::1/128 proto any vr0 protect prio 0
  link-vr0 cached-SA 0 genid 0
 ESP tunnel 2.1.0.1 - 2.1.0.5
  sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
```

See also:

To dynamically configure IPsec tunnels, see the *Control Plane Security - IKEv1 and IKEv2* documentation.

Offload of cryptographic operations

Maximal throughput of a tunnel is limited. In order to increase this limit cryptographic operations are offloaded to idle fast path cores. Gain provided by this feature depends of the traffic model and the number of idle fast path cores. Typically the maximal throughput of a tunnel with IMIX traffic is double if a fast path core is available to do the cryptographic operations.

Details of the way to configure offloading of the cryptographic operations is detailed in *FPN-SDK Cryptographic offloading*

By default the crytographic offloading is done only for packets received from the tunnel. The cryptographic offloading for packets sent into the tunnel can be enabled through the cli but can impact packet ordering (especially if there are many flows aggregated in the tunnel) and cause issue with the anti-replay windows of the IPsec SA.

Providing options

`--sa-hash-order`
Size order of IPv6 IPsec SAD hash table. Value automatically updated if ipsec:–max-sa is changed.

Default value 16

Range 16 .. 20

`--sp-hash-order`
Size order of IPv6 IPsec SPD hash table. Value automatically updated if ipsec:–max-sp is changed.

Default value 9

Range 8 .. 16

Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

---

2.3.12 Fast Path LAG

Overview

Fast Path LAG provides bonding, or LAG, support in the fast path.

Features

- Supported algorithms:
  - `balance-rr` mode (Round-robin policy)
  - `active-backup` mode (use the `active-slave` interface to send packets)
  - `balance-xor` mode (xor policy)
  - `802.3ad` mode IEEE (802.3ad Dynamic link aggregation)
  - flow hash computed from Ethernet protocol, vlan ID, IP addresses, L4 protocols and ports out of the most inner packet inside Ethernet, VLAN, IPv4, IPv6, IP tunnels, GRE.
- Supported hash policies (only for `balance-xor` and `802.3ad`):
  - `layer2` xmit hash policy

  - `layer2+3` xmit hash policy
  - `layer3+4` xmit hash policy
  - `encap2+3` xmit hash policy
  - `encap3+4` xmit hash policy
- Manage LAG interfaces:
  - Displaying LAG interfaces and their slaves.
  - Create a LAG interface.
  - Delete a LAG interface and its links to slave interfaces.
  - Link a LAG interface (master) to another interface (slave).
  - Unlink an interface (slave) from a LAG interface (master).
  - Set the LAG policies. The default is balance-rr (round robin).
  - Set the LAG xmit hash policy (default is layer2).
- Compatibility with VRF processing.
- NUMA awareness

Dependencies

6WINDGate modules

- *Fast Path Baseline*

Linux

- Basic Linux synchronization is a kernel patch (upstream 2.6.33)

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=88ead977109d
- Linux bonding mode synchronization is a kernel patch (upstream 3.13). Without this patch, the mode is balance-rr by default.

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=90af231106c0b8d223c27d35464af95cb3d9cacf
- Linux active-slave interface synchronization for the active-backup mode is a kernel patch (upstream 3.13).

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=ec76aa49855f
- 802.3ad synchronization is a kernel patch (upstream 3.14)

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=4ee7ac7526d4a9413cafa733d824edfe49fdcc46

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=1d3ee88ae0d605629bf369ab0b868dae8ca62a48

http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=df7dbcbbafc0b8f3fb31a40c6f3c4a7e15cb0b40

- Linux bonding xmit hash policy synchronization is a kernel patch (upstream 3.19). Without this patch, the xmit hash policy is layer2 by default.

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=f70161c67231f54f784529d7447ce4386d258b7a

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

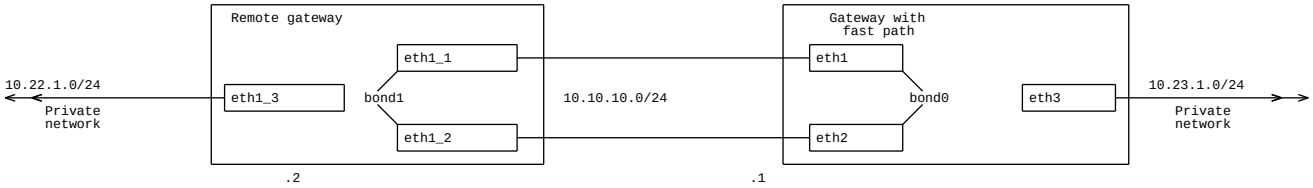
You can configure LAG interfaces via `iproute2` standard shell commands.

Managing LAG interfaces

Simple LAG

This example is relevant to a CentOS 7 machine.

We will assume the network topology is the following:



1. Configure IP addresses and routes:

```
$ sudo ip link set eth3 up
$ sudo ip addr add 10.23.1.1/24 dev eth3
$ sudo ip link set eth1 down
$ sudo ip link set eth2 down
```

2. Create a bonding interface:

```
$ sudo modprobe bonding
$ sudo ip addr add 10.10.10.1/24 dev bond0
$ sudo ip link set dev bond0 up
$ sudo ip link set eth1 master bond0
$ sudo ip link set eth2 master bond0
```

3. **[Optional]** For the `active-backup` mode instead of the default `balance-rr` mode, the first command must be replaced by:

```
$ sudo modprobe bonding mode=active-backup miimon=100
```

4. Create a new route:

```
$ sudo ip route add 10.22.1.0/24 dev bond0
```

5. Display the characteristics of LAG interfaces on the fast path:

```
$ ip -d address show
11: eth1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast␣
↪master bond0 state UP group default qlen 1000
    link/ether 52:54:00:00:01:00 brd ff:ff:ff:ff:ff:ff promiscuity 0

12: eth2: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast␣
↪master bond0 state UP group default qlen 1000
    link/ether 52:54:00:00:01:00 brd ff:ff:ff:ff:ff:ff promiscuity 0

14: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state␣
↪UP group default
    link/ether 52:54:00:00:01:00 brd ff:ff:ff:ff:ff:ff promiscuity 0
    bond
    inet 10.10.10.1/24 scope global bond0
       valid_lft forever preferred_lft forever
```

```
$ cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: load balancing (round-robin)
MII Status: up
MII Polling Interval (ms): 0
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 52:54:00:00:01:00
Slave queue ID: 0

Slave Interface: eth2
MII Status: up
Speed: 1000 Mbps
Duplex: full
```

(continues on next page)

(continued from previous page)

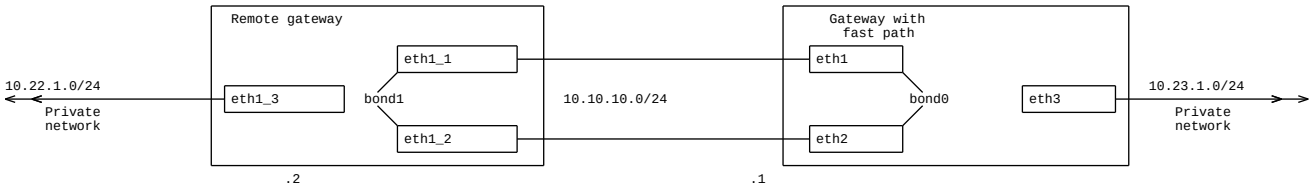
```
Link Failure Count: 0
Permanent HW addr: 52:54:00:00:02:00
Slave queue ID: 0
```

```
$ fpcmd lag
bond0-vr0
  mode: round-robin
  slaves (2):
    eth1-vr0 can_tx=yes
         state=active link=up queue_id=0 link_failure_count=0
         perm_hwaddr=52:54:00:00:01:00
    eth2-vr0 can_tx=no
         state=backup link=up queue_id=0 link_failure_count=0
         perm_hwaddr=52:54:00:00:02:00
```

LAG with XOR mode

This example is relevant to a CentOS 7 machine.

We will assume the network topology is the following:



1. Configure IP addresses and routes:

```
$ sudo ip link set eth3 up
$ sudo ip addr add 10.23.1.1/24 dev eth3
$ sudo ip link set eth1 down
$ sudo ip link set eth2 down
```

2. Create a bonding interface:

```
$ sudo modprobe bonding mode=balance-xor
$ sudo ip addr add 10.10.10.1/24 dev bond0
$ sudo ip link set dev bond0 up
$ sudo ip link set eth1 master bond0
$ sudo ip link set eth2 master bond0
```

3. **[Optional]** If your distribution does not support mode synchronization, set the fast path LAG mode:

```
$ fp-cli lag-mode-set bond0 xor
```

4. Create a new route:

```
$ sudo ip route add 10.22.1.0/24 dev bond0
```

5. Display the characteristics of LAG interfaces on the fast path:

```
$ ip -d address show
11: eth1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast␣
↪master bond0 state UP group default qlen 1000
    link/ether 52:54:00:00:01:00 brd ff:ff:ff:ff:ff:ff promiscuity 0

12: eth2: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast␣
↪master bond0 state UP group default qlen 1000
    link/ether 52:54:00:00:01:00 brd ff:ff:ff:ff:ff:ff promiscuity 0

14: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state␣
↪UP group default
    link/ether 52:54:00:00:01:00 brd ff:ff:ff:ff:ff:ff promiscuity 0
    bond
    inet 10.10.10.1/24 scope global bond0
       valid_lft forever preferred_lft forever
```

```
$ cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: load balancing (xor)
Transmit Hash Policy: layer2 (0)
MII Status: up
MII Polling Interval (ms): 0
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 52:54:00:00:01:00
Slave queue ID: 0

Slave Interface: eth2
MII Status: up
Speed: 1000 Mbps
```

(continues on next page)

(continued from previous page)

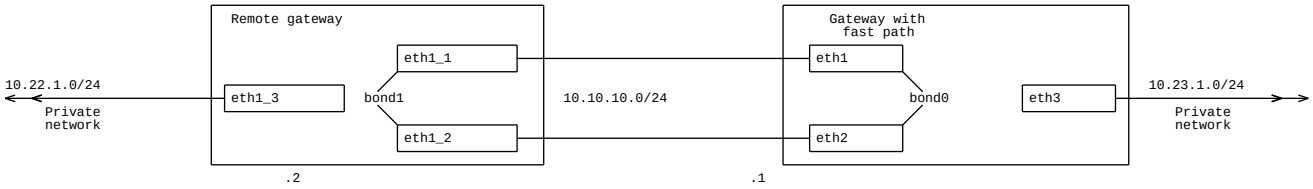
```
Duplex: full
Link Failure Count: 0
Permanent HW addr: 52:54:00:00:02:00
Slave queue ID: 0
```

```
$ fpcmd lag
bond0-vr0
  mode: xor
  xmit_hash_policy: layer2
  slaves (2):
    eth1-vr0 can_tx=yes
         state=active link=up queue_id=0 link_failure_count=0
         perm_hwaddr=52:54:00:00:01:00
    eth2-vr0 can_tx=no
         state=backup link=up queue_id=0 link_failure_count=0
         perm_hwaddr=52:54:00:00:02:00
```

LAG with XOR mode and specific hash policy

This example is relevant to a CentOS 7 machine.

We will assume the network topology is the following:



1. Configure IP addresses and routes:

```
$ sudo ip link set eth3 up
$ sudo ip addr add 10.23.1.1/24 dev eth3
$ sudo ip link set eth1 down
$ sudo ip link set eth2 down
```

2. Create a bonding interface with a specific hash policy:

```
$ sudo modprobe bonding mode=balance-xor xmit-hash-policy=layer2+3
$ sudo ip addr add 10.10.10.1/24 dev bond0
$ sudo ip link set dev bond0 up
$ sudo ip link set eth1 master bond0
$ sudo ip link set eth2 master bond0
```


3. **[Optional]** If your distribution does not support mode synchronization, set the fast path LAG mode:

```
$ fp-cli lag-mode-set bond0 xor
```

4. **[Optional]** If your distribution does not support hash policy synchronization, set the fast path LAG xmit hash policy:

```
$ fp-cli lag-hash-policy-set bond0 layer2+3
```

5. Create a new route:

```
$ sudo ip route add 10.22.1.0/24 dev bond0
```

6. Display the characteristics of LAG interfaces on the fast path:

```
$ ip -d address show
11: eth1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast␣
↪master bond0 state UP group default qlen 1000
    link/ether 52:54:00:00:01:00 brd ff:ff:ff:ff:ff:ff promiscuity 0

12: eth2: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast␣
↪master bond0 state UP group default qlen 1000
    link/ether 52:54:00:00:01:00 brd ff:ff:ff:ff:ff:ff promiscuity 0

14: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state␣
↪UP group default
    link/ether 52:54:00:00:01:00 brd ff:ff:ff:ff:ff:ff promiscuity 0
    bond
    inet 10.10.10.1/24 scope global bond0
       valid_lft forever preferred_lft forever
```

```
$ cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: load balancing (xor)
Transmit Hash Policy: layer2+3 (2)
MII Status: up
MII Polling Interval (ms): 0
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
```

(continues on next page)

(continued from previous page)

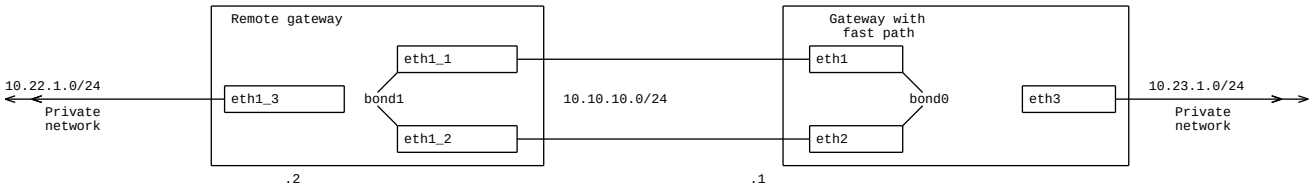
```
Permanent HW addr: 52:54:00:00:01:00
Slave queue ID: 0

Slave Interface: eth2
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 52:54:00:00:02:00
Slave queue ID: 0
```

```
$ fpcmd lag
bond0-vr0
  mode: xor
  xmit_hash_policy: layer2+3
  slaves (2):
    eth1-vr0 can_tx=yes
         state=active link=up queue_id=0 link_failure_count=0
         perm_hwaddr=52:54:00:00:01:00
    eth2-vr0 can_tx=no
         state=backup link=up queue_id=0 link_failure_count=0
         perm_hwaddr=52:54:00:00:02:00
```

LAG with 802.3ad mode

This example is relevant to a CentOS 7 machine.

We will assume the network topology is the following:



1. Configure IP addresses and routes:

```
$ sudo ip link set eth3 up
$ sudo ip addr add 10.23.1.1/24 dev eth3
$ sudo ip link set eth1 down
$ sudo ip link set eth2 down
```

2. Restart Linux bonding module:

```
$ sudo modprobe -r bonding
```

3. Create a bonding interface:

```
$ sudo modprobe bonding mode=802.3ad lacp_rate=1 miimon=100 max_bonds=0
$ sudo ip link add bond0 type bond
$ sudo ip link set dev bond0 down
$ sudo ip addr add 10.10.10.1/24 dev bond0
$ sudo ip link set dev bond0 up
$ sudo ip link set eth1 master bond0
$ sudo ip link set eth2 master bond0
```

4. Create a new route:

```
$ sudo ip route add 10.22.1.0/24 dev bond0
```

5. Display the characteristics of LAG interfaces on the fast path:

```
$ ip -d address show
11: eth1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast␣
↪master bond0 state UP group default qlen 1000
    link/ether 52:54:00:00:01:00 brd ff:ff:ff:ff:ff:ff promiscuity 0

12: eth2: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast␣
↪master bond0 state UP group default qlen 1000
    link/ether 52:54:00:00:01:00 brd ff:ff:ff:ff:ff:ff promiscuity 0

14: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state␣
↪UP group default
    link/ether 52:54:00:00:01:00 brd ff:ff:ff:ff:ff:ff promiscuity 0
    bond
    inet 10.10.10.1/24 scope global bond0
       valid_lft forever preferred_lft forever
```

```
$ cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: IEEE 802.3ad Dynamic link aggregation
Transmit Hash Policy: layer2 (0)
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

802.3ad info
```

(continues on next page)

(continued from previous page)

```
LACP rate: fast
Min links: 0
Aggregator selection policy (ad_select): stable
Active Aggregator Info:
        Aggregator ID: 1
        Number of ports: 1
        Actor Key: 9
        Partner Key: 1
        Partner Mac Address: 00:00:00:00:00:00

Slave Interface: eth1
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 52:54:00:00:01:00
Slave queue ID: 0
Aggregator ID: 1
Actor Churn State: monitoring
Partner Churn State: monitoring
Actor Churned Count: 0
Partner Churned Count: 0
details actor lacp pdu:
    system priority: 65535
    port key: 9
    port priority: 255
    port number: 1
    port state: 79
details partner lacp pdu:
    system priority: 65535
    oper key: 1
    port priority: 255
    port number: 1
    port state: 1

Slave Interface: eth2
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 52:54:00:00:02:00
Slave queue ID: 0
Aggregator ID: 2
Actor Churn State: monitoring
```

(continues on next page)

(continued from previous page)

```
Partner Churn State: monitoring
Actor Churned Count: 0
Partner Churned Count: 0
details actor lacp pdu:
    system priority: 65535
    port key: 9
    port priority: 255
    port number: 2
    port state: 71
details partner lacp pdu:
    system priority: 65535
    oper key: 1
    port priority: 255
    port number: 1
    port state: 1
```

```
$ fpcmd lag
bond0-vr0
  mode: 802.3ad
  xmit_hash_policy: layer2
  info_aggregator: 2, info_num_ports: 1
  slaves (2):
    eth1-vr0 can_tx=yes
          state=active link=up queue_id=0 link_failure_count=0
          perm_hwaddr=52:54:00:00:01:00
          aggregator_id=1
    eth2-vr0 can_tx=no
          state=backup link=up queue_id=0 link_failure_count=0
          perm_hwaddr=52:54:00:00:02:00
          aggregator_id=2
```

Managing LAG interfaces with fp-cli

Starting fp-cli

1. The `fp-cli` commands below allow you to manage LAG interfaces:

```
$ fp-cli
```

Displaying LAG interfaces and their slaves

Synopsis

```
lag
```

Example

```
<fp-0> lag
bond0-vr0
  mode: round-robin
  slaves (2):
    eth1-vr0 can_tx=yes
          state=active link=up queue_id=0 link_failure_count=0
          perm_hwaddr=00:21:85:c1:82:58
    eth4-vr0 can_tx=yes
          state=active link=up queue_id=0 link_failure_count=0
          perm_hwaddr=00:00:46:50:4e:00
bond1-vr0
  mode: 802.3ad
  xmit_hash_policy: layer2
  info_aggregator: 1, info_num_ports: 1
  slaves (1):
    eth3-vr0 can_tx=no
          state=active link=up queue_id=0 link_failure_count=0
          perm_hwaddr=00:30:1b:b4:df:94
          aggregator_id=1

bond2-vr0
  mode: xor
  xmit_hash_policy: layer2+3
  slaves (1):
    eth5-vr0 can_tx=yes
          state=active link=up queue_id=0 link_failure_count=0
          perm_hwaddr=00:00:46:50:c2:04
```

Setting the LAG policies

The default is balance-rr (round robin).

Synopsis

```
lag-mode-set <master_ifname> round-robin|xor
```

<master_ifname> Name of the bonding interface.

round-robin Transmit packets in sequential order from the first available slave through the last. This mode provides load balancing and fault tolerance.

xor Transmit according to the hash policy matching the xmit-hash-policy value.

The default applied policy is dependent on whether or not the distribution provides netlink notification support for the xmit-hash-policy setting:

- if support is provided, the default applied policy is layer2.
- otherwise, the default applied policy is layer2+3.

This mode provides load balancing and fault tolerance.

See also:

Setting the LAG xmit-hash-policy

Example

```
<fp-0> lag-mode-set bond0 round-robin
```

Note: You can use the 802.3ad mode only if the *Linux - Fast Path Synchronization* supports this feature.

Setting the LAG xmit-hash-policy

Select the transmit hash policy to use for slave selection in balance-xor, 802.3ad modes.

Synopsis

```
lag-hash-policy-set <master_ifname> layer2|layer2+3|layer3+4|encap2+3|encap3+4
```

<master_ifname> Name of the bonding interface.

layer2 The hash is computed from the source and destination MAC addresses modulo the number of slaves.

layer2+3 This policy uses a combination of layer2 and layer3 protocol information to generate the hash.

The hash is computed from the IP/IPv6 source and destination addresses modulo the number of slaves. If the packet contains a VLAN header, its ID is used as well.

This policy is intended to provide a more balanced distribution of traffic than layer2 alone, especially in environments where a layer3 gateway device is required to reach most destinations.

layer3+4 This policy uses upper layer protocol information, when available, to compute the hash. This allows for traffic to a particular network peer to span multiple slaves, although a single connection will not span multiple slaves.

encap2+3 This policy uses the same method than layer2+3, but it dissects all recognized encapsulation layers (refer to *fpn-flow api* for more details), if any. It uses the most inner L3 information to generate the hash.

encap3+4 This policy uses the same method than layer3+4, but it dissects all recognized encapsulation layers (refer to *fpn-flow api* for more details), if any. It uses the most inner L3/L4 information to generate the hash.

Example

```
<fp-0> lag-hash-policy-set bond0 layer3+4
```

Force the LAG to be socket aware

The LAG policy is socket independent i.e. application of this policy can forward a packet allocated in a socket to another socket. But cross socket reduces performance (see *Performance optimization of FPN-SDK Add-on for DPDK* for more details).

It is possible to take into account the socket information to apply the LAG policy thanks `fp-cli`. When LAG is socket aware, the LAG policy is applied only on the subset of interface belonging to the same socket of the considered packet.

Synopsis

```
numa-aware-set lag on|off
```

By default application of the LAG policy is socket independent.

For more details on the CLI commands to manage NUMA awareness, see *NUMA awareness configuration*.

Providing options

Some capabilities can be tuned for this module.

--iface-max
: Maximum number of LAG interfaces.

  Default value 32

  Memory footprint per LAG interfaces 136 B

Example

```
FP_OPTIONS="--mod-opt=lag:--iface-max=16"
```

Then fast path can manage up to 16 LAG interfaces.

--max-slaves
: Maximum number of slaves for all LAG interfaces

  Default value 64

  Memory footprint per LAG slave interfaces 128 B

Example

```
FP_OPTIONS="--mod-opt=lag:--max-slaves=16"
```

max-slaves-per-iface
: Maximum number of slaves per LAG interface

  Default value

  Range 0 .. max-slaves

Example

```
FP_OPTIONS="--mod-opt=lag:--max-slaves-per-iface=16"
```

--slaves-hash-order

Size order of LAG slave interfaces hash table Value automatically updated if *--max-slaves* is changed.

Default value

Range 1 .. 31

Example

```
FP_OPTIONS="--mod-opt=lag:--slaves-hash-order=10"
```

Tip: To get optimal performance, apply the following ratios to the two parameters:

Parameter	Value
–slaves-hash-order	`N`
–max-slaves	`2 ** N`

Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

---

2.3.13 Fast Path License

Overview

The Fast Path License module controls the availability of the license in the fast path.

Features

Request license information to the licensing daemon, and control fast path capabilities according to the enabled features.

Dependencies

6WINDGate modules

- *Fast Path Baseline*
- Licensing daemon (*vrld*).

Usage

Displaying license status

The license status of the fast path can be displayed from `fp-cli`.

Synopsis

```
license
```

Example

```
<fp-0> license
VALID: License is valid.
- Product: on
- IPsec: on
- CGNAT: off
```

If the license is invalid, check the configuration of *vrld*.

Displaying license statistics

If a license is missing or is invalid, the packets are sent as exception, and a statistics counter is increased for each packet.

These counters are available in global statistics from `fp-cli`.

Example

```
<fp-0> stats-global
  (...)
  fp_missing_product_license:20
  fp_missing_ipsec_license:6
  (...)
```

2.3.14 Fast Path MACVLAN

Overview

Fast Path MACVLAN provides MACVLAN devices in the fast path.

Features

- Supported features:
  - VEPA mode
  - Private mode
  - Passthru mode

Dependencies

6WINDGate modules

- *Fast Path Baseline*

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

Fast path

You can manage MACVLAN devices via the `fp-cli` commands below.

Displaying MACVLAN devices

Synopsis

```
macvlan
```

Example

```
<fp-0> macvlan
ntfp2-vr0:
        macvlan3-vr0: mode: private
        macvlan1-vr0: mode: private
ntfp3-vr0:
        macvlan2-vr0: mode: passthru
```

Displaying MACVLAN information

Synopsis

```
iface
```

Example

```
<fp-0> iface
...
10:macvlan1 [VR-0] ifid=10 (virtual) <FWD4|FWD6> (0x18)
        type=macvlan mac=26:ff:f5:17:53:6d mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
        if_ops: rx_dev=none rx_early=none tx_dev=macvlan ip_output=none
        mode private, link ntfp2
11:macvlan2 [VR-0] ifid=11 (virtual) <FWD4|FWD6> (0x18)
        type=macvlan mac=de:ed:03:d6:ff:33 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
        if_ops: rx_dev=none rx_early=none tx_dev=macvlan ip_output=none
        mode passthru, link ntfp3
```

(continues on next page)

(continued from previous page)

```
12:macvlan3 [VR-0] ifid=12 (virtual) <FWD4|FWD6> (0x18)
        type=macvlan mac=a6:09:fe:65:09:c7 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
        if_ops: rx_dev=none rx_early=none tx_dev=macvlan ip_output=none
        mode private, link ntfp2
...
```

Providing options

Some capabilities can be tuned for this module.

--ifaces
: Maximum number of MACVLAN interfaces

  Default value 127

  Memory footprint per MACVLAN interfaces 50 B

  Range 0 .. 5000

Example

```
FP_OPTIONS="--mod-opt=macvlan:--ifaces=256"
```

Then fast path can manage up to 256 MACVLAN interfaces.

--hash-order
: Size order of MACVLAN interfaces hash table. Value automatically updated if *--ifaces* is changed.

  Default value 8

  Range 1 .. 31

Example

```
FP_OPTIONS="--mod-opt=macvlan:--hash-order=9"
```

Tip: To get optimal performance, apply the following ratios to the two parameters:

Parameter	Value
–hash-order	`N`
–ifaces	`2 ** N`

Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

2.3.15 Fast Path MPLS

Overview

Fast Path MPLS provides MPLS functionalities in the fast path.

MPLS is defined by **RFC 3031** (https://tools.ietf.org/html/rfc3031.html).

Features

- Packet forwarding
- Label Swapping
- Penultimate Hop Popping
- IPv4 and IPv6 support for Next Hop Label Forwarding Entry
- IPv4 Explicit NULL Label (defined by **RFC 3032** (https://tools.ietf.org/html/rfc3032.html))
- IPv6 Explicit NULL Label (defined by **RFC 3032** (https://tools.ietf.org/html/rfc3032.html))

Dependencies

6WINDGate modules

- *Fast Path Baseline*
- *Fast Path Forwarding IPv4*
- *Fast Path Forwarding IPv6*

Linux

- MPLS (pop and swap) is a kernel patch (upstream 4.1).

  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=0189197f4416
- MPLS (push) is a kernel patch (upstream 4.3).

  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=e3e4712ec096

- MPLS input flag synchronization is a kernel patch (upstream 4.11).

  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=24045a03b879

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

With *Linux - Fast Path Synchronization* you can manage MPLS via standard Linux commands.

Synchronizing Linux and the fast path

1. Load required modules:

```
# modprobe mpls_router
# modprobe mpls_iptunnel
```

Managing MPLS routes from Linux

The Linux commands below allow you to manage MPLS routing.

Enabling MPLS forwarding on the system

```
# sysctl -w net.mpls.platform_labels=LABEL
```

LABEL Higher label value allowed in the routing table. More details here: https://www.kernel.org/doc/Documentation/networking/mpls-sysctl.txt

Enabling MPLS input on an interface

```
# sysctl -w net.mpls.conf.IFNAME.input=1
```

IFNAME Interface's name. More details here: https://www.kernel.org/doc/Documentation/networking/mpls-sysctl.txt

Adding an MPLS 'pop' route entry

```
# ip -f mpls route add IN_LABEL via INET ADDR dev IFNAME
```

IN_LABEL Input label to pop.

INET Family of the nexthop: inet for IPv4 or inet6 for IPv6.

ADDR IP nexthop address.

IFNAME Interface name of the output interface.

Example

Add a route to forward and pop packet with label 123 via 1.2.3.4@eth0.

```
# ip -f mpls route add 123 via inet 1.2.3.4 dev eth0
```

Adding an MPLS 'swap' route entry

```
# ip -f mpls route add IN_LABEL as OUT_LABELS via INET ADDR dev IFNAME
```

IN_LABEL Input label to swap.

OUT_LABELS One or more output labels. In case of multiple output labels, they are separated by a '/'.

INET Family of the nexthop: inet for IPv4 or inet6 for IPv6.

ADDR IP nexthop address.

IFNAME Interface name of the output interface.

Example

Add a route to forward packet with label 123 via 1.2.3.4@eth0 and swap the label 123 by the label 456.

```
# ip -f mpls route add 123 as 456 via inet 1.2.3.4 dev eth0
```

Adding an MPLS 'push' route entry

```
# ip route add SUBNET nexthop encap mpls OUT_LABELS via INET ADDR dev IFNAME
```

SUBNET Subnet target of the route.

OUT_LABELS One or more output labels. In case of multiple output labels, they are separated by a '/'.

INET Family of the nexthop: inet for IPv4 or inet6 for IPv6.

ADDR IP nexthop address.

IFNAME Interface name of the output interface.

Example

Add a route to forward packets destinated to 10.0.0.0/8 through mpls with the label 789 via 1.2.3.4@eth0.

```
# ip route add 10.0.0.0/8 nexthop encap mpls 789 via inet 1.2.3.4 dev eth0
```

Note that an iproute2 v4.4.0 (iproute2-ss160111) is needed.

Displaying information about the MPLS routing table

```
# ip -f mpls route
```

Deleting an MPLS route entry

```
# ip -f mpls route del IN_LABEL
```

IN_LABEL Input label of the route to delete.

Managing MPLS routes from the fast path

The `fp-cli` command below allows you to manage MPLS routes from the fast path.

Displaying MPLS routes

```
mpls
```

Example

```
<fp-0> mpls
R[000001] vrfid 0 label [200] via inet 10.200.0.1 dev ntfp2-vr0
R[000002] vrfid 0 label [300] as [400] via inet 10.200.0.1 dev ntfp2-vr0
```

2.3.16 Fast Path NAT

Overview

Fast Path NAT provides Network Address Translation in the fast path.

To ensure maximal performance, this module implements simple functions based on information found in the shared memory.

If the module cannot find in the shared memory the relevant information based on L3, L4, and L5 headers, the fast path raises an exception.

In accordance with configured filter rules with higher priorities, this exception:

- interacts with other 6WINDGate entities, or,
- sends the packet to Linux networking stack
- drops the packet for security reasons.

The connection tracking establishment and the ALG are managed by the Linux networking stack thanks to the exception mechanism.

Features

- Static and dynamic NAT/PAT
- Connection tracking and ALG (Linux-based)

Dependencies

6WINDGate modules

- *Fast Path Filtering IPv4*

Linux

- Netfilter: create audit records for x_tables replaces is a kernel patch (upstream 3.9)

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=fbabf31e4d482149b5e
- RPF netfilter export xt_rpfilter.h to userland is a kernel patch (upstream 3.12)

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=f0c03956ac40fdc4fb

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

```
# modprobe nf_conntrack_netlink
```

Example

1. Set up a NAT rule under Linux:

```
# echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
# echo 1 > /proc/sys/net/netfilter/nf_conntrack_tcp_be_liberal
# ip link set eth1 up
# ip link set eth2 up
# ip ad ad 2.0.0.1/24 dev eth1
# ip ad ad 2.1.0.1/24 dev eth2
# ip route add 100.2.2.1/32 via 2.0.0.5
# ip route add 110.2.2.1/32 via 2.1.0.5
# iptables -P INPUT ACCEPT
# iptables -P FORWARD ACCEPT
# iptables -P OUTPUT ACCEPT
# iptables -t nat -F
# iptables -t nat -A POSTROUTING -s 100.0.0.0/8 -o eth2 -j SNAT --to-source 2.1.0.
↪1
# iptables -vL -t nat
  Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
   pkts bytes target     prot opt in     out     source       destination
```

(continues on next page)

(continued from previous page)

```
  Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
   pkts bytes target     prot opt in     out     source       destination

  Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
   pkts bytes target     prot opt in     out     source       destination

  Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
   pkts bytes target     prot opt in     out     source       destination
      0     0 SNAT       all  --  any    eth2    100.0.0.0/8  anywhere     to:2.1.
↪0.1
```

Note: Address pool and port range are also supported, see below:

```
# iptables -t nat -A POSTROUTING -p tcp -o eth2 -j SNAT --to-source 2.0.0.1-2.0.0.
↪50:1024-2048
```

2. Launch the `fp-cli` module and check fast path statistics:

```
# fp-cli
```

```
<fp-0> nf4-rules nat
Chain PREROUTING (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source       destination

Chain INPUT (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source       destination

Chain OUTPUT (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source       destination

Chain POSTROUTING (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source       destination
       0          0 SNAT      all  --   any    eth2   100.0.0.0/8  anywhere
```

The NAT rule is correctly implemented on the fast path.

NAT management

Displaying the NAT status in the fast path

Use the *nf4-hook* command.

Example

```
<fp-0> nf4-hook priority
FP_NF_IP_PRE_ROUTING:
        ct nat
FP_NF_IP_LOCAL_IN:
        nat
FP_NF_IP_FORWARD:

FP_NF_IP_LOCAL_OUT:
        ct nat
FP_NF_IP_POST_ROUTING:
        nat
```

Enabling or disabling the NAT in the fast path

Use the *nf4-hook-set* command.

Example

```
<fp-0> nf4-hook-set nat all_hooks on
Set nat pre_routing: on
Set nat local_in: on
Set nat local_out: on
Set nat post_routing: on
<fp-0> nf4-hook-set ct all_hooks on
Set ct pre_routing: on
Set ct local_out: on
```

2.3.17 Fast Path OVS Acceleration

Overview

Fast Path OVS Acceleration provides Open vSwitch acceleration in the fast path.

It implements high performance virtual switching, transparently synchronized with the standard Linux Open vSwitch control plane and data plane, using OpenFlow or the Open vSwitch command line. This does not require any modification to Open vSwitch, Linux applications, management or orchestration software.

OVS acceleration can be combined with other fast path protocols such as VXLAN, GRE, NAT/filtering, IPsec and more to provide enhanced services at the hypervisor level.

Synchronization with the Open vSwitch control plane is provided by *Linux - Fast Path Synchronization*. Statistics synchronization require *Control Plane OVS*.

Features

- Flows matching attributes:
  - Ethertype
  - VLAN 802.1q
  - IP
  - IPv6
  - UDP/TCP (v4 and v6)
  - ICMP
  - ICMPv6
  - ARP
- Actions:
  - push/pop VLAN header
  - set attribute in the packet
    - MAC address
    - TCP/UDP port
    - IPv6 (addresses, traffic class, flow label, hop limit)
    - IPv4 (addresses, TOS, TTL)
    - SKB mark
  - output
- VXLAN and VXLAN GBP (VXLAN Group Based Policy) tunnelling (require *Fast Path VXLAN*).

- GRE tunnelling (requires *Fast Path GRE*).
- Transparent synchronization with Open vSwitch control plane through *Linux - Fast Path Synchronization* (flow and port statistics synchronization require *Control Plane OVS*).
- Supports a maximum of 256 ports at one time.
- Supports a maximum of 65536 flows by default. This value can be changed when the fast path is launched with the –flows plugin option.
- Supports megaflow. This Open vSwitch feature limits the number of flow creation/deletion by allowing to wildcard flow fields during matching.
- Supports recirculation. This Open vSwitch feature is used in recent versions to implement 'resubmit'. It tags a flow with a recirculation id, and has the packets go through the flow table again. It is used to chain flows.
- Supports a flow cache to speed-up the packet processing. The cache can provide a boost of performance between 10% to 60% in function of the number of flows.

Dependencies

6WINDGate modules

- *Fast Path Baseline* - Plugins
- *Fast Path VXLAN* (for VXLAN tunnelling)
- *Fast Path GRE* (for GRE tunnelling)
- *Linux - Fast Path Synchronization*
- *Control Plane OVS* (for flow and port statistics synchronization)

Linux

- *Control Plane OVS*
- VXLAN GBP support is an Open vSwitch patch (upstream 2.4) and needs Linux kernel >= 4.0

Usage

Principles

- The Open vSwitch daemons should be restarted after starting the fast path. Indeed, the fast path generates new network interfaces. If the daemons are not stopped, the bridges will have to be destroyed and recreated to re-apply the virtual switch configuration.
- The Open vSwitch daemons can be restarted. The interfaces that are put in a bridge must be set up and promiscuous.

Using an Open vSwitch distribution package

This section implies that 6WIND packages were already installed (at least *Fast Path Baseline*, Fast Path OVS Acceleration and *Linux - Fast Path Synchronization*).

Example

Here is an example of configuration for a virtual bridge between two physical ports:

1. Install the Open vSwitch package:

```
```

2. Configure and start the fast path.
3. Start the linux synchronization.
4. Restart openvswitch service:

```
# /etc/init.d/openvswitch-switch stop
# /etc/init.d/openvswitch-switch start
```

5. Configure a bridge between two ports:

```
# ovs-vsctl add-br br0
# ovs-vsctl add-port br0 eth0
# ovs-vsctl add-port br0 eth1
```

6. Add an *OpenFlow* controller (optional, the installation of such controller is not covered by this document):

```
# ovs-vsctl set-controller br0 tcp:192.168.0.27:6633
```

7. Set the interfaces up and promiscuous:

```
# ip link set eth0 up
# ip link set eth0 promisc on
# ip link set eth1 up
# ip link set eth1 promisc on
# ip link set br0 up
```

See also:

- The *Fast Path Baseline* documentation
- The *OpenStack* and Open vSwitch documentation
- The *Linux - Fast Path Synchronization* documentation

Managing a VXLAN port

The Open vSwitch commands below allow you to manage VXLAN ports.

Creating a VXLAN port

```
# ovs-vsctl add-port BRIDGENAME PORTNAME -- set interface PORTNAME type=vxlan \
  options:remote_ip=REMOTE_IP options:key=VNI options:dst_port=DSTPORT
```

BRIDGENAME Open vSwitch bridge's name.

PORTNAME VXLAN port's name.

VNI VXLAN Network Id.

REMOTE_IP remote ip address.

DSTPORT Port number of the VXLAN port (Linux default value is 8472).

Example

Create new Open vSwitch VXLAN port (vxlan1) in Open vSwitch bridge br0 with vni *1* and remote ip address *10.125.0.2*.

```
# ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=vxlan \
  options:remote_ip=10.125.0.2 options:key=1 options:dst_port=8472
```

Creating a VXLAN GBP port

```
# ovs-vsctl add-port BRIDGENAME PORTNAME -- set interface PORTNAME type=vxlan \
  options:remote_ip=REMOTE_IP options:key=VNI options:dst_port=DSTPORT options:exts=gbp
```

Example

Create a new VXLAN GBP port and set the mark *0xabcd* in gbp field. And also drop any incomming VXLAN GBP packet to the VXLAN GBP port with the matching mark *0x1234*.

```
# ovs-vsctl add-port br0 vxlan1 -- set interface vxlan1 type=vxlan \
  options:remote_ip=10.125.0.2 options:key=1 options:dst_port=8472 options:exts=gbp

# ovs-ofctl add-flow br0 'in_port=1,actions=load:0xabcd->NXM_NX_TUN_GBP_ID[],NORMAL'

# ovs-ofctl add-flow br0 'priority=10,in_port=3,tun_gbp_id=0x1234,actions=drop'
```

Adapting maximum retention time for idle flows

By default, idle flows will be kept for 10 seconds in ovs-vswitchd flow table before disappearing. You may want to prolong the maximum time (in ms) idle flows will remain cached.

For instance, in order to set 60s as the maximum before idle flows removal, use:

```
# ovs-vsctl set Open_vSwitch . other_config:max-idle=60s
```

Note: This is a maximum value. You don't have an absolute guarantee that idle flows will be kept that long. Refer to the official ovs-vswitchd documentation for details.

Note: When performing zero-loss performance tests, it is interesting to have a high value like 60s. Indeed, between two iterations, flows may otherwise disappear and packets would then go through the slow path before synchronization with the fast path. The slow path being what it is, packets would get lost and the resulting performance tests would be wrong.

About the flow cache

The Fast Path OVS Acceleration provides a cache to speed-up the packet processing. The cache can provide a boost of performance between 10% to 60% in function of the number of flows.

The cache is disabled by default. It can be dynamically enabled with a fp-cli command (*fp-cli-cache-set*).

The default cache size is calculated in function of the cpu cache size, to avoid any memory access for performance reason. Example of maximal number of flows handled by the cache is function of the last level cpu cache size:

- last level cpu cache=10MB, default cache size is limited to 32768 flows.
- last level cpu cache=20MB, default cache size is limited to 65536 flows.

The cache has several limitations that can impact the performance negatively in some use cases:

- The cache size is limited in number of flows. If Fast Path OVS Acceleration needs to process more than this maximal size, the cache impacts the performance negatively.
- The cache uses the RSS hash computed by the network device. If the NIC doesn't support this feature (i.e. virtual device like *Virtio Host PMD*, Virtio Guest XEN-KVM PMD), or the driver is unable to copy the hash into the packet descriptor (i.e. *Mellanox ConnectX-3 EN series PMD*), the cache can not be used for the packets coming from this interface.
- If two packets have the same RSS hash (i.e. non ip packet, packet with same src/dst ip and src/dst port), only one is processed by the cache. In this case, the second packet processing costs more CPU cycles with the cache than without it.

- The cache is not used for GRE packets. As all packets from a GRE tunnel have the same RSS hash (i.e. the ip addresses and ports of the outer packets are identical), it will generate many cache conflicts.
- The cache is not used for recirc packets. As the RSS hash of a recirc packet is the same that the initial packet injected in Fast Path OVS Acceleration, it will always conflict with the cache entry of the initial packet.

Providing options

You can dynamically set 3 parameters when starting the fast path:

- Number of flows with option `--mod-opt=fp-vswitch:--flows=<flow-number>`. Default is `65536`.
- Number of ports with option `--mod-opt=fp-vswitch:--ports=<port-count>`. Default is `256`.
- Number of masks with option `--mod-opt=fp-vswitch:--masks=<mask-number>`. Default is `32768`.
- Hash order with option `--mod-opt=fp-vswitch:--hash-order=<hash-order>`. Default is `16`.
- Cache hash order with option `--mod-opt=fp-vswitch:--cache-hash-order=<hash-order>`. Default is computed in function of the cpu cache size. For more information, see *about-flow-cache* .

Tip: To get optimal performance, apply the following ratios to the three parameters:

Parameter	Value
–hash-order	`N`
–flows	`2 ** N`
–masks	`2 ** (N - 1)`

`fp-cli` commands

Enabling Fast Path OVS Acceleration provides the following additional `fp-cli` commands.

fp-vswitch-cache

Description

Show if the Fast Path OVS Acceleration cache is enabled or disabled.

Synopsis

```
fp-vswitch-cache
```

Example

```
<fp-0> fp-vswitch-cache
cache is off
```

fp-vswitch-cache-set

Description

Enable/Disable the Fast Path OVS Acceleration cache.

Synopsis

```
fp-vswitch-cache-set on|off
```

Example

```
<fp-0> fp-vswitch-cache-set on
cache is on (was off)
```

fp-vswitch-ports

Description

Print the list of ports synchronized in the fast path.

Synopsis

```
fp-vswitch-ports [percore] [all]
```

Parameters

percore Display statistic values for each core.

all Display all statistics (even those that are null).

Example

```
<fp-0> fp-vswitch-ports all
0: ovs-system (internal)
  rx_pkts:0
  tx_pkts:0
  rx_bytes:0
  tx_bytes:0
1: br0 (internal)
  rx_pkts:0
  tx_pkts:0
  rx_bytes:0
  tx_bytes:0
2: eth1 (netdev)
  rx_pkts:56
  tx_pkts:53
  rx_bytes:6529
  tx_bytes:6253
3: eth2 (netdev)
  rx_pkts:53
  tx_pkts:56
  rx_bytes:6253
  tx_bytes:6529
```

fp-vswitch-stats

Description

Print statistics about packets and flows in fp-vswitch module. All statistics below are number of packets, except control plane flow statistics, which are number of flows

Synopsis

```
fp-vswitch-stats
```

Flow statistics

flow_not_found The flow was not found in the shared memory. Packet goes to exception.

flow_pullup_failed Flow extraction failed, data in mbuf could not be made contiguous. Packet goes to exception.

flow_pullup_too_small Flow extraction failed, not enough data was made contiguous. Packet goes to exception.

flow_frag_lookup_fail Lookup of the so-called "later" fragment flow (respectively the "first" fragment flow) failed when matching the first fragment of a packet (respectively non-first fragments of a packet). Packet holding the fragment goes to exception.

Output statistics

output_ok Successfully sent out of fp-vswitch plugin.

output_failed_no_mbuf Failed to duplicate packet before sending it.

output_failed_no_ifp Failed to find an interface to send it to.

output_failed_operative Interface to send is down.

output_toobig_dropped Non-IP packet does not fit in the interface's MTU.

output_failed Output on a GRE port failed.

output_failed_unknown_type OVS port type is not supported.

Action statistics

output_dropped No action found for the flow. Packet is dropped.

userspace Packet processed by the ovs-vswitchd daemon.

push_vlan VLAN header was added.

pop_vlan VLAN header was removed.

push_mpls MPLS header was added.

pop_mpls MPLS header was removed.

recirc Recirculation action was executed.

set_ethernet Ethernet header was changed.

set_mpls MPLS header was changed.

set_priority Unused.

set_tunnel_id Packet was encapsulated in a tunnel.

set_ipv4 IPv4 header was changed.

set_ipv6 IPv6 header was changed.

set_tcp TCP header was changed.

set_udp UDP header was changed.

set_mark SKB mark was changed.

Control plane flow statistics

flow_add_failed Flow could not be added to shared memory.

flow_update_failed Flow could not be updated in shared memory.

flow_delete_failed Flow could not be deleted from shared memory.

Example

```
<fp-0> fp-vswitch-stats
  cache_hit:60
  cache_miss:5
  flow_not_found:3
  flow_pullup_failed:0
  flow_pullup_too_small:0
  output_ok:65
  output_failed_no_mbuf:0
```

(continues on next page)

(continued from previous page)

```
output_failed_no_ifp:0
output_failed_operative:0
output_toobig_dropped:0
output_failed:0
output_failed_unknown_type:0
output_dropped:0
userspace:0
push_vlan:0
pop_vlan:0
push_mpls:0
pop_mpls:0
recirc:0
set_ethernet:0
set_mpls:0
set_priority:0
set_tunnel_id:0
set_ipv4:0
set_ipv6:0
set_tcp:0
set_udp:0
flow_add_failed:0
flow_update_failed:0
flow_delete_failed:0
```

fp-vswitch-flows

Description

Dump the current flow table as a human-readable C-like structure. Only flows with traffic are displayed: flows are removed as soon as traffic stops. The output is similar to `ovs-dpctl` display for key, mask and action (default value).

Note: This command doesn't dump the controller's flow table.

Synopsis

```
fp-vswitch-flows [help|[{+|-}]{item}] [...]
```

Parameters

To display the items below, prefix them with a plus sign (+).

To hide the items below, prefix them with a minus sign (-).

help List all available items.

flow Affect `flow.*` items globally.

next Next flow index (enabled by default).

flow.key Affect `flow.key.*` items globally.

flow.actions Defined flow actions.

flow.actions_len Size of `flow.actions[]` in bytes.

flow.dup Number of output and recirc in `flow.actions[]`.

flow.except Indicate if the packets matching this flow are sent as exceptions.

flow.index Flow index.

flow.hash Flow hash.

flow.state Flow state (unspecified = 0, active = 1).

flow.age Flow age. Unused.

flow.key.l1.ovsport Input port.

flow.key.l2.src Ethernet source address.

flow.key.l2.dst Ethernet destination address.

flow.key.l2.ether_type Ethernet frame type.

flow.key.l2.vlan_tci If 802.1Q, TCI | VLAN_CFI; otherwise 0.

flow.key.l3.frag FLOW_FRAG_* flags.

flow.key.l3.tos IP ToS (type of service) (including DSCP and ECN).

flow.key.l3.ttl IP TTL (Time to live)/Hop limit.

flow.key.l3.proto IP protocol or lower 8 bits of ARP opcode.

flow.key.l3.ip.src IPv4 source address.

flow.key.l3.ip.dst IPv4 destination address.

flow.key.l3.ip.arp.sha ARP source hardware address.

flow.key.l3.ip.arp.tha ARP target hardware address.

flow.key.l3.ip6.src IPv6 source address.

flow.key.l3.ip6.dst IPv6 destination address.

flow.key.l3.ip6.label IPv6 flow label.

flow.key.l3.ip6.ndp.target IPv6 neighbor discovery (ND) target.

flow.key.l3.ip6.ndp.sll IPv6 neighbor discovery (ND) source hardware address.

flow.key.l3.ip6.ndp.tll IPv6 neighbor discovery (ND) target hardware address.

flow.key.l4.flags TCP flags.

flow.key.l4.sport TCP/UDP/SCTP source port.

flow.key.l4.dport TCP/UDP.SCTP destination port.

flow.key.tunnel.id Encapsulating tunnel ID.

flow.key.tunnel.src Tunnel outer IPv4 src addr.

flow.key.tunnel.dst Tunnel outer IPv4 dst addr.

flow.key.tunnel.flags Tunnel flags.

flow.key.tunnel.tos Tunnel ToS.

flow.key.tunnel.ttl Tunnel TTL

Examples

```
<fp-0> fp-vswitch-flows
FPVS flow table (max 65536 flows):
  sizeof(fpvs_flow_entry_t): 2432
  sizeof(struct fpvs_flow): 2400
  Flow max age: 2

.table = {
  [4] = { .ufid = 0x3bd045ef-b29b-4249-8525-aa0f4a9b444b, .pkts = 35, .bytes = 4541,
  .flow.key = recirc(0),in_port(2),eth(src=de:ed:01:1d:3f:0f,dst=de:ed:02:08:d7:ad),
↪eth_type(0x0800),ipv4(src=0.0.0.0,dst=0.0.0.0,proto=0,tos=0,ttl=0,frag=0),l4(sport=0,
↪dport=0,flags=0),
  .flow.mask = recirc(00000000),in_port(ffffffff),eth(src=ff:ff:ff:ff:ff:ff,
↪dst=ff:ff:ff:ff:ff:ff),eth_type(0xffff),vlan(id=ffff),ipv4(src=0.0.0.0,dst=0.0.0.0,
↪proto=0,tos=0,ttl=0,frag=ff),l4(sport=0,dport=0,flags=0),
  .flow.actions = actions(output:3),
  },
  [5] = { .ufid = 0x1595126b-3554-4d47-ae52-a9d38ed11698, .pkts = 32, .bytes = 4271,
```

(continues on next page)

(continued from previous page)

```
  .flow.key = recirc(0),in_port(3),eth(src=de:ed:02:08:d7:ad,dst=de:ed:01:1d:3f:0f),
↪eth_type(0x0800),ipv4(src=0.0.0.0,dst=0.0.0.0,proto=0,tos=0,ttl=0,frag=0),l4(sport=0,
↪dport=0,flags=0),
  .flow.mask = recirc(00000000),in_port(ffffffff),eth(src=ff:ff:ff:ff:ff:ff,
↪dst=ff:ff:ff:ff:ff:ff),eth_type(0xffff),vlan(id=ffff),ipv4(src=0.0.0.0,dst=0.0.0.0,
↪proto=0,tos=0,ttl=0,frag=ff),l4(sport=0,dport=0,flags=0),
  .flow.actions = actions(output:2),
  },
}
```

fp-vswitch-masks

Description

Dump the current mask table as a human-readable C-like structure. The output is similar to ovs-dpctl display.

Synopsis

```
fp-vswitch-masks
```

Examples

```
<fp-0> fp-vswitch-masks
FPVS flow mask table (max 32768 masks):
  sizeof(fpvs_mask_entry_t): 192
  sizeof(struct fpvs_mask): 160

.table = {
  [2] = {
    ref_count = 2,
    range = [0x18, 0x38],
    key = recirc(00000000),in_port(ffffffff),eth(src=ff:ff:ff:ff:ff:ff,
↪dst=ff:ff:ff:ff:ff:ff),eth_type(0xffff),vlan(id=ffff),
↪l3(value=0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
↪proto=0,tos=0,ttl=0,frag=ff),l4(sport=0,dport=0,flags=0)
  },
}
```

2.3.18 Fast Path Policy-Based Routing

Overview

Fast Path Policy-Based Routing extends fast path routing by providing a subset of Linux PBR functionality.

PBR is a way to forward packets based on multiple criteria, not only the IP destination. It is implemented in Linux using `iproute2` commands `ip rule` and `ip route`.

Features

- management of several routing tables
- management of PBR rules. The following subset of `ip rule` options are supported:
  - selector:
    - `priority`: priority of the rule (a.k.a. `pref` or `order`)
    - `from`: source address or prefix
    - `to`: destination address or prefix
    - `fwmark`: fwmark of the packet
    - `iif`: input interface (a.k.a. `dev`)
    - `not`: flag that inverts the match result
  - action:
    - `lookup`: LPM lookup in a routing table (a.k.a. `table`)
- IPv4 and IPv6 support
- VRF support

Dependencies

6WINDGate modules

- *Fast Path Baseline*
- *Fast Path Forwarding IPv4*

and optionally:

- *Fast Path Forwarding IPv6*

Linux

None

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

PBR can be divided into 2 parts:

- Policy routing rules: a set of rules that select packets based on various information (source address, input interface…) and apply a routing action, typically perform a route lookup in a routing table.
- Routing tables: Linux creates 2 of them by default:
  - the `local` table (table 255), automatically populated by the kernel and containing routes local to the machine (local loopback, broadcast, local addresses…)
  - the `main` table (table 254), where all routes are added by default

  Additionally, a user may create and populate custom tables.

With *Linux - Fast Path Synchronization*, PBR may be managed via standard Linux commands:

- configuration of PBR rules
- configuration of routes in various routing tables

Rules

The routing policy database is the first stage of route lookup. An ordered list of PBR rules exists in each VRF and for each IP version.

The following IPv4 rules are created by default:

```
# ip rule show
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default
```

These rules mean:

- for all packets, do a LPM lookup in the `local` table. If a route is found, return it, otherwise jump to next rule
- for all packets, do a LPM lookup in the `main` table. If a route is found, return it, otherwise jump to next rule
- for all packets, do a LPM lookup in the `default` table. If a route is found, return it, otherwise return an error

The following IPv6 rules are created by default:

```
# ip -6 rule show
0:      from all lookup local
32766:  from all lookup main
```

These rules mean:

- for all packets, do a LPM lookup in the `local` table. If a route is found, return it, otherwise jump to next rule
- for all packets, do a LPM lookup in the `main` table. If a route is found, return it, otherwise return an error

All these rules may be deleted, and new rules may be added.

The Linux commands below enable to manage PBR rules. All these commands will be synchronized to the fast path.

Rules are managed by the `ip rule` command:

```
# ip rule help
Usage: ip rule [ list | add | del | flush | save ] SELECTOR ACTION
       ip rule restore
SELECTOR := [ not ] [ from PREFIX ] [ to PREFIX ] [ tos TOS ]
              ~~~     ~~~~            ~~
            [ fwmark FWMARK[/MASK] ]
              ~~~~~~
            [ iif STRING ] [ oif STRING ] [ pref NUMBER ]
              ~~~                           ~~~~
ACTION := [ table TABLE_ID ]
            ~~~~~
          [ realms [SRCREALM/]DSTREALM ]
          [ goto NUMBER ]
          SUPPRESSOR
SUPPRESSOR := [ suppress_prefixlength NUMBER ]
              [ suppress_ifgroup DEVGROUP ]
TABLE_ID := [ local | main | default | NUMBER ]
```

The underlined selector and action types are supported. Some of them have aliases:

- selector types:
  - `not`: flag that inverts the match result
  - `from`: source address or prefix
  - `to`: destination address or prefix
  - `fwmark`: mark of the packet
  - `iif`: input interface (a.k.a. `dev`)
  - `pref`: priority of the rule (a.k.a. `priority` or `order`)
- action types:

- `table`: LPM lookup in a routing table (a.k.a. `lookup`)

Adding a rule

`iproute2` enables to add new rules. Fast Path Policy-Based Routing supports selector fields `priority`, `from`, `to`, `iif`, `fwmark` and `not`, and action `lookup`. All selector fields are optional.

The routing table in the `lookup` action may be referenced by its 32 bit ID or by a literal name listed in `/etc/iproute2/rt_tables`.

Example

Examples of IPv4 rules:

```
# ip rule add priority 100 from 10.100.0.1 lookup 180
# ip rule add priority 150 from 10.100.0.0/24 lookup main
# ip rule add priority 170 iif eth1 lookup 200
# ip rule add priority 170 from 10.50.0.0/16 iif eth2 lookup 210
# ip rule add priority 190 not iif vxlan1 lookup 220
```

Examples of IPv6 rules:

```
# ip -6 rule add priority 100 from 3ffe:304:124:100::1 lookup 180
# ip -6 rule add priority 150 from 3ffe:304:124:100::/64 lookup main
# ip -6 rule add priority 170 iif eth1 lookup 200
# ip -6 rule add priority 170 from 3ffe:304:124:50::/64 iif eth2 lookup 210
# ip -6 rule add priority 190 not iif vxlan1 lookup 220
```

Note: There is no unique identifier for a rule. Several rules may share the same priority and duplicate rules are even supported.

It is recommended to assign a unique priority to each rule and use it as an identifier. This precaution is important if one wants to later delete the right PBR rule.

Note: The interface specified in the `iif` field is a physical or logical interface identified by its ifname. It does not need to exist at the time the rule is created. It may appear later, be deleted, be created again with a different ifindex. Linux and the fast path will properly recognize the interface and apply the rules as expected.

Displaying rules in the Linux kernel

Example

Display IPv4 rules:

```
# ip rule show
0:      from all lookup local
100:    from 10.100.0.1 lookup 180
150:    from 10.100.0.0/24 lookup main
170:    from all iif eth1 lookup 200
170:    from 10.50.0.0/16 iif eth2 lookup 210
190:    not from all iif vxlan1 lookup 220
32766:  from all lookup main
32767:  from all lookup default
```

Display IPv6 rules:

```
# ip -6 rule show
0:      from all lookup local
100:    from 3ffe:304:124:100::1 lookup 180
150:    from 3ffe:304:124:100::/64 lookup main
170:    from all iif eth1 lookup 200
170:    from 3ffe:304:124:50::/64 iif eth2 lookup 210
190:    not from all iif vxlan1 lookup 220
32766:  from all lookup main
```

Displaying rules in the fast path

Example

Display IPv4 rules:

```
<fp-0> pbr4-rule
    0: from all lookup 255
  100: from 10.100.0.1 lookup 180
  150: from 10.100.0.0/24 lookup 254
  170: from all iif eth1 lookup 200
  170: from 10.50.0.0/16 iif eth2 lookup 210
  190: not from all iif vxlan1 lookup 220
32766: from all lookup 254
32767: from all lookup 253
```

Display IPv6 rules:


```
<fp-0> pbr6-rule
    0: from all lookup 255
  100: from 3ffe:304:124:100::1 lookup 180
  150: from 3ffe:304:124:100::/64 lookup 254
  170: from all lookup 200
  170: from 3ffe:304:124:50::/64 lookup 210
  190: not from all lookup 220
32766: from all lookup 254
```

Deleting a rule

Note: As stated in *Adding a rule*, there is no unique identifier for a rule. The Linux implementation of PBR rule deletion is peculiar, even puzzling. For example, if the current rules are:

```
100: from 10.100.0.1 iif eth1 lookup 180
110: from all iif eth1 lookup 200
```

Then, the following command will delete the first rule, which is probably not what the user expected:

```
ip rule delete from all iif eth1
```

This is the reason why it is recommended to assign a different priority to each rule. It can then be deleted by specifying its priority:

```
ip rule delete priority 110
```

Anyhow, the fast path manager will always delete the same PBR rule as the kernel.

Example

Delete the IPv4 rule with priority 150:

```
# ip rule del priority 150
```

Delete the IPv6 rule with priority 150:

```
# ip -6 rule del priority 150
```

Note: All rules may be deleted, including default rules. It is even possible, though not recommended, to delete rule 0 (`from all lookup local`)

```
# ip rule del priority 0
```

Default rules may be manually restored, like any PBR rule:

```
# ip rule add priority 0 lookup local
# ip rule add priority 32766 lookup main
# ip rule add priority 32767 lookup default
```

Flushing rules

`iproute2` enables to flush all rules in one command. It will delete all rules, except rules with priority 0.

Example

Flush all IPv4 rules, except rules with priority 0:

```
# ip rule flush
```

Delete all IPv6 rules, except rules with priority 0:

```
# ip -6 rule flush
```

Behavior for unsupported rules

Some rules maybe be unsupported by fast path, either because of unsupported match or unsupported action. In case of unsupported match, a packet hitting this rule will automatically be sent as exception to be processed by Linux kernel. In case of unsupported action, a packet hitting the rule will be sent as exception only if there is a match. The unsupported parts are displayed at fast path level.

```
<fp-0> pbr4-rule
    0: from all lookup 255
  100: from 1.0.0.0/24 [unsup. action]
  120: [unsup. match] from 1.0.0.0/24 lookup 29
  200: [unsup. match] from all [unsup. action]
32766: from all lookup 254
32767: from all lookup 253
```

```
<fp-0> pbr6-rule
    0: from all lookup 255
  100: from 3ffe::/64 [unsup. action]
```

(continues on next page)

(continued from previous page)

```
  110: from 3ffe::/64 [unsup. action]
  120: from 3ffe::/64 lookup 29
  140: [unsup. match] from all lookup 7
  200: [unsup. match] from all [unsup. action]
  900: from 3ffe::/64 lookup 254
32766: from all lookup 254
```

Routes

The Linux commands below enable to manage routes in various routing tables. All these commands will be synchronized to the fast path.

Note: Routes from Linux `local` table are not synchronized to the fast path. The fast path manager populates the fast path `local` table with special routes mainly deduced from addresses configured on the machine.

Adding a route into a specific routing table

By default, `iproute2` configures routes in the `main` table (table 254). But other routing tables may be configured, such as the `local` table (table 255), the `default` table (table 253) or custom tables created and populated by the user.

A routing table may be referenced by its 32 bit ID or by a literal name listed in `/etc/iproute2/rt_tables`. This file may be updated to add new table names.

A route may be added into a specific table by specifying `table TABLEID` in the `ip route` command, were TABLEID is its numerical ID or a literal name. Adding a route to a nonexistent table will create it.

Linux routing tables and routes are then synchronized in the fast path (except the `local` table).

Example

Here, we add IPv4 routes into the `main` table, then in table 200. This will create the IPv4 table 200:

```
# ip link set eth2 up
# ip addr add 10.23.1.101/24 dev eth2

# ip route add 10.24.1.0/24 via 10.23.1.201
# ip route add 10.25.1.0/24 via 10.23.1.201 table 200
```

Here, we add IPv6 routes into the `main` table, then in table 200. This will create the IPv6 table 200:


```
# ip link set eth2 up
# ip addr add 3ffe:304:124:2301::101/64 dev eth2

# ip route add 3ffe:304:124:2401::101/64 via 3ffe:304:124:2301::201
# ip route add 3ffe:304:124:2501::101/64 via 3ffe:304:124:2301::201 table 200
```

Displaying routes in Linux kernel

By default, `iproute2` displays routes in the `main` table (table 254):

Example

```
# ip route show
10.23.1.0/24 dev eth2  proto kernel  scope link  src 10.23.1.101
10.24.1.0/24 via 10.23.1.201 dev eth2
```

```
# ip -6 route show
3ffe:304:124:2301::/64 dev eth2  proto kernel  metric 256  pref medium
3ffe:304:124:2401::/64 via 3ffe:304:124:2301::201 dev eth2  metric 1024  pref medium
fe80::/64 dev eth2  proto kernel  metric 256  pref medium
```

To display routes in a specific table, specify `table TABLEID`:

```
# ip route show table 200
10.25.1.0/24 via 10.23.1.201 dev eth2
```

```
root@dut-vm:~# ip -6 route show table 200
3ffe:304:124:2501::/64 via 3ffe:304:124:2301::201 dev eth2  metric 1024  pref medium
```

```
# ip route show table local
local 10.0.2.15 dev mgmt0  proto kernel  scope host  src 10.0.2.15
broadcast 10.23.1.0 dev eth2  proto kernel  scope link  src 10.23.1.101
local 10.23.1.101 dev eth2  proto kernel  scope host  src 10.23.1.101
broadcast 10.23.1.255 dev eth2  proto kernel  scope link  src 10.23.1.101
broadcast 127.0.0.0 dev lo  proto kernel  scope link  src 127.0.0.1
local 127.0.0.0/8 dev lo  proto kernel  scope host  src 127.0.0.1
local 127.0.0.1 dev lo  proto kernel  scope host  src 127.0.0.1
broadcast 127.255.255.255 dev lo  proto kernel  scope link  src 127.0.0.1
```

```
# ip -6 route show table local
local ::1 dev lo  proto none  metric 0  pref medium
```

(continues on next page)

(continued from previous page)

```
local 3ffe:304:124:2301:: dev lo  proto none  metric 0  pref medium
local 3ffe:304:124:2301::101 dev lo  proto none  metric 0  pref medium
local fe80:: dev lo  proto none  metric 0  pref medium
local fe80:: dev lo  proto none  metric 0  pref medium
local fe80::200:46ff:fe50:4e00 dev lo  proto none  metric 0  pref medium
local fe80::dced:2ff:fe1c:341a dev lo  proto none  metric 0  pref medium
ff00::/8 dev eth2  metric 256  pref medium
```

To display routes in all tables, specify `table all`:

```
# ip route show table all
10.25.1.0/24 via 10.23.1.201 dev eth2  table 200
10.23.1.0/24 dev eth2  proto kernel  scope link  src 10.23.1.101
10.24.1.0/24 via 10.23.1.201 dev eth2
local 10.0.2.15 dev mgmt0  table local  proto kernel  scope host  src 10.0.2.15
broadcast 10.23.1.0 dev eth2  table local  proto kernel  scope link  src 10.23.1.101
local 10.23.1.101 dev eth2  table local  proto kernel  scope host  src 10.23.1.101
broadcast 10.23.1.255 dev eth2  table local  proto kernel  scope link  src 10.23.1.101
broadcast 127.0.0.0 dev lo  table local  proto kernel  scope link  src 127.0.0.1
local 127.0.0.0/8 dev lo  table local  proto kernel  scope host  src 127.0.0.1
local 127.0.0.1 dev lo  table local  proto kernel  scope host  src 127.0.0.1
broadcast 127.255.255.255 dev lo  table local  proto kernel  scope link  src 127.0.0.1
fe80::/64 dev eth2  proto kernel  metric 256  pref medium
unreachable default dev lo  table unspec  proto kernel  metric 4294967295  error -101␣
↪pref medium
local ::1 dev lo  table local  proto none  metric 0  pref medium
local fe80:: dev lo  table local  proto none  metric 0  pref medium
local fe80:: dev lo  table local  proto none  metric 0  pref medium
local fe80::200:46ff:fe50:4e00 dev lo  table local  proto none  metric 0  pref medium
local fe80::dced:2ff:fe1c:341a dev lo  table local  proto none  metric 0  pref medium
ff00::/8 dev eth2  table local  metric 256  pref medium
unreachable default dev lo  table unspec  proto kernel  metric 4294967295  error -101␣
↪pref medium
```

```
# ip -6 route show table all
3ffe:304:124:2501::/64 via 3ffe:304:124:2301::201 dev eth2  table 200  metric 1024 ␣
↪pref medium
unreachable default dev lo  table unspec  proto kernel  metric 4294967295  error -101␣
↪pref medium
3ffe:304:124:2301::/64 dev eth2  proto kernel  metric 256  pref medium
3ffe:304:124:2401::/64 via 3ffe:304:124:2301::201 dev eth2  metric 1024  pref medium
fe80::/64 dev eth2  proto kernel  metric 256  pref medium
unreachable default dev lo  table unspec  proto kernel  metric 4294967295  error -101␣
↪pref medium
```

(continues on next page)

(continued from previous page)

```
local ::1 dev lo  table local  proto none  metric 0  pref medium
local 3ffe:304:124:2301:: dev lo  table local  proto none  metric 0  pref medium
local 3ffe:304:124:2301::101 dev lo  table local  proto none  metric 0  pref medium
local fe80:: dev lo  table local  proto none  metric 0  pref medium
local fe80:: dev lo  table local  proto none  metric 0  pref medium
local fe80::200:46ff:fe50:4e00 dev lo  table local  proto none  metric 0  pref medium
local fe80::dced:2ff:fe1c:341a dev lo  table local  proto none  metric 0  pref medium
ff00::/8 dev eth2  table local  metric 256  pref medium
unreachable default dev lo  table unspec  proto kernel  metric 4294967295  error -101␣
↪pref medium
```

Displaying routes in the fast path

The fp-cli `route4` command displays IPv4 routes in all tables, or in a specific table if `table TABLEID` is specified.

Example

```
<fp-0> route4
# - Preferred, * - Active, > - selected
(254) 10.24.1.0/24  [08]  ROUTE gw 10.23.1.201 via eth2-vr0 (13)
(200) 10.25.1.0/24  [08]  ROUTE gw 10.23.1.201 via eth2-vr0 (14)
```

```
<fp-0> route4 table 200
# - Preferred, * - Active, > - selected
(200) 10.25.1.0/24  [08]  ROUTE gw 10.23.1.201 via eth2-vr0 (14)
```

```
<fp-0> route4 type all
# - Preferred, * - Active, > - selected
(255) 0.0.0.0/32  [5001]  LOCAL (1)
(255) 10.0.2.15/32  [01]  ADDRESS via mgmt0-vr0 (5)
(255) 10.23.1.101/32  [05]  ADDRESS via eth2-vr0 (10)
(255) 127.0.0.0/8  [5002]  BLACKHOLE (4)
(255) 224.0.0.0/4  [5001]  LOCAL (3)
(255) 255.255.255.255/32  [5001]  LOCAL (2)
(254) 10.23.1.0/24  [06]  CONNECTED via eth2-vr0 (11)
(254) 10.24.1.0/24  [08]  ROUTE gw 10.23.1.201 via eth2-vr0 (13)
(200) 10.25.1.0/24  [08]  ROUTE gw 10.23.1.201 via eth2-vr0 (14)
```

The fp-cli `route6` command displays IPv6 routes in all tables, or in a specific table if `table TABLEID` is specified:

```
<fp-0> route6
# - Preferred, * - Active, > - selected
(254) 3ffe:304:124:2401::/64  [03]  ROUTE gw 3ffe:304:124:2301::201 via eth2-vr0 (6)
(200) 3ffe:304:124:2501::/64  [03]  ROUTE gw 3ffe:304:124:2301::201 via eth2-vr0 (7)
```

```
<fp-0> route6 type all table 200
# - Preferred, * - Active, > - selected
(200) 3ffe:304:124:2501::/64  [03]  ROUTE gw 3ffe:304:124:2301::201 via eth2-vr0 (7)
```

```
<fp-0> route6 type all
# - Preferred, * - Active, > - selected
(255) ::/80  [5002]  BLACKHOLE (3)
(255) 3ffe:304:124:2301::101/128  [02]  ADDRESS via eth2-vr0 (5)
(255) fe80::/10  [5001]  LOCAL (1)
(255) ff00::/8  [5001]  LOCAL (2)
(254) 3ffe:304:124:2301::/64  [01]  CONNECTED via eth2-vr0 (4)
(254) 3ffe:304:124:2401::/64  [03]  ROUTE gw 3ffe:304:124:2301::201 via eth2-vr0 (6)
(200) 3ffe:304:124:2501::/64  [03]  ROUTE gw 3ffe:304:124:2301::201 via eth2-vr0 (7)
```

Note: The fast path maintains an extra table (table 0), that holds neighbor entries (ARP and NDP (Neighbor Discovery Protocol) entries) in the form of /32 or /128 routes of type NEIGH. These entries are hidden by default, except if you request to display routes of type `neigh` or `all`.

Deleting a route from a specific routing table

By default, `iproute2` deletes routes in the `main` table (table 254). But another routing table may be specified by adding `table TABLEID` to the command.

Example

```
# ip route del 10.24.1.0/24
```

```
# ip route del 10.25.1.0/24 table 200
```

```
# ip route del 3ffe:304:124:2401::101/64
```

```
# ip route del 3ffe:304:124:2501::101/64 table 200
```

Flushing routes from a specific routing table

`iproute2` enables to flush all routes in one command, with mandatory parameters.

Among parameters, a table ID may be specified, `all` standing for all tables.

Example

Flush all IPv4 rules in table 200:

```
# ip route flush table 200
```

Flush all IPv4 routes in table `main`:

```
# ip route flush table main
```

Flush all IPv6 routes in table 200:

```
# ip -6 route flush table 200
```

Flush all IPv6 routes in table `main`:

```
# ip -6 route flush table main
```

Note: It is of course not recommended to flush the `local` table, which is automatically populated by the kernel. In case you did it though, the routes may be restored by setting all interfaces down then up again, including `lo`.

```
# for i in /sys/class/net/*; do \
  DEV=$(basename $i); \
  ip link set $DEV down; \
  ip link set $DEV up; \
done
```

Providing options

`--max-rules`

Maximum number of PBR rules.

At least 5 PBR rules (3 for IPv4 and 2 for IPv6) are created per VR

Default value 1024

Memory footprint per PBR rule 100 B

Range 0 .. 400K

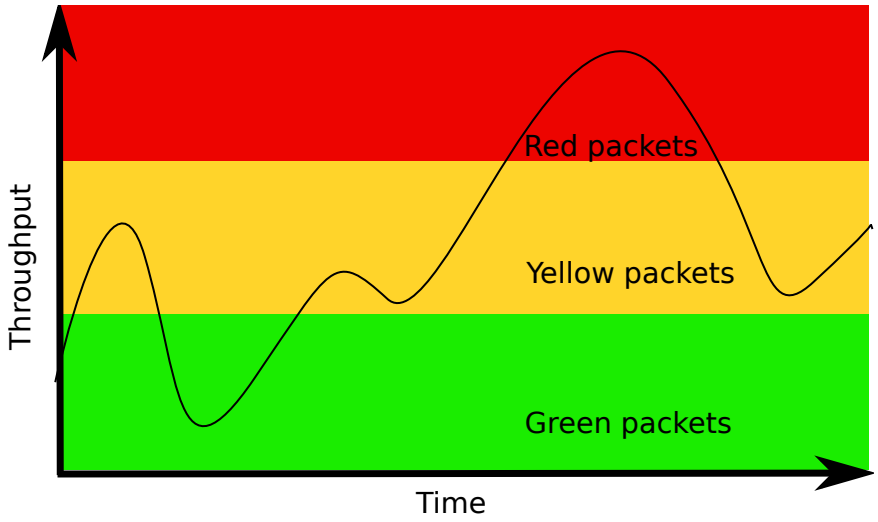
Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

2.3.19 Fast Path QoS Basic

Overview

Fast Path QoS Basic is a fast path trTCM (two rate Three Color Marker) which helps implementing QoS (Quality of Service) on your network.

Fast Path QoS Basic marks packets as either Green, Yellow, or Red. You can then process packets accordingly (by dropping red packets, for example).



Fast Path QoS Basic operates in Color-Aware Mode (in which packets may have been previously colored) or in In Color-Blind Mode (in which are not precolored).

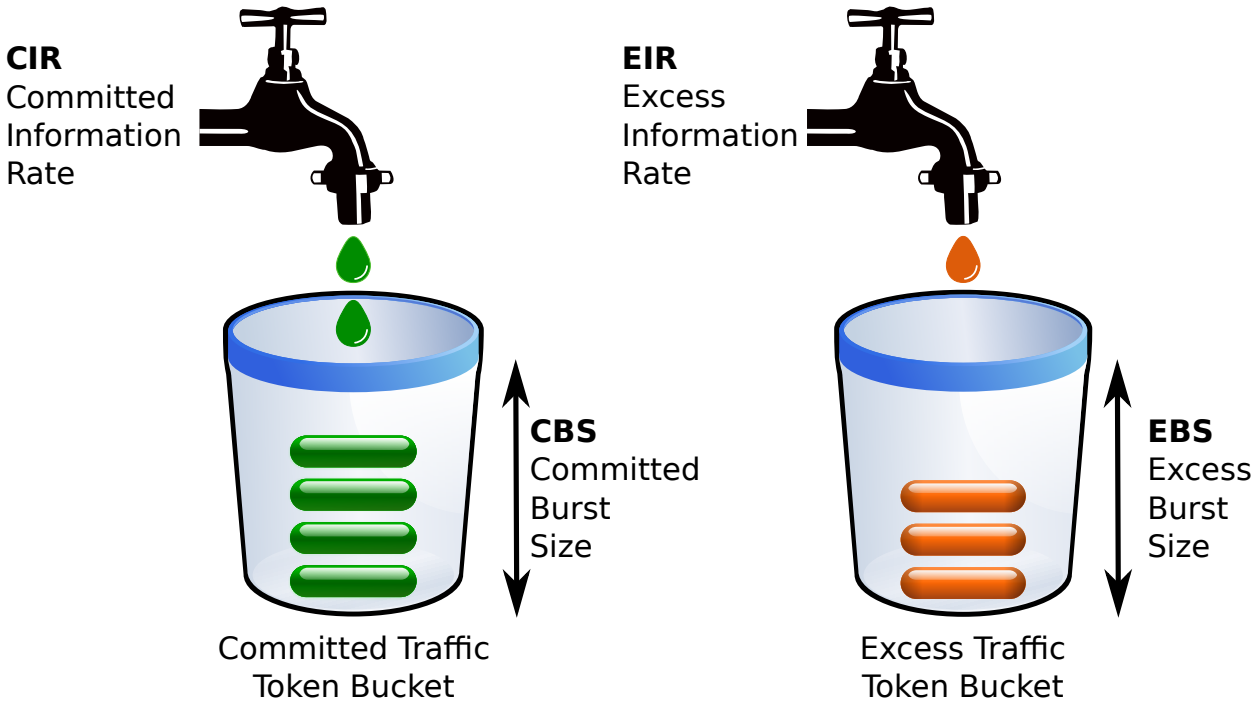
Fast Path QoS - Exception Rate Limitation is a feature of *Fast Path QoS* that limits the rate of packets sent as exceptions from the fast path to Linux.

Features

- **RFC 4115** (https://tools.ietf.org/html/rfc4115.html) trTCM
- Color-aware and color-blind support
- Per packets or per bytes accounting
- FPN-SDK traffic conditioner API
- Exception Rate Limiting to control the rate of packets sent as exceptions from fast path to Linux

The implementation is based on **RFC 4115** (https://tools.ietf.org/html/rfc4115.html). The operation of the marker is described by two token generation rates, CIR (Committed Information Rate) and EIR (Excess Information Rate), and two token bucket sizes, CBS (Committed Burst Size) and EBS (Excess Burst Size).

Token bucket generation rate	Token bucket size
CIR	CBS
EIR	EBS



The number or token buckets is:

- incremented periodically, and,
- decremented whenever a packet meets a bucket's criteria (i.e., if its size in less than or equal to the size corresponding to the number or tokens).

The rate at which the counter is decremented determines the average bandwidth. The value of the threshold determines the burstiness (intermittent increases and decreases in frequency of packet transmission).

The buckets status evolves over time, as illustrated in the table below:

Time	Committed Traffic Token Bucket	Excess Traffic Token Bucket
0	Full	Full
t	As long as the token count is lower than the CBS, it is incremented by one CIR times per second.	As long as the token count is lower than the EBS, it is incremented by one EIR times per second.

In the simplest case (green packet in color-aware mode), packets are processed as follows:

- If a green packet meets the Committed Traffic Token Bucket's criteria, it remains marked as green.
- If a green packet does not meet the Committed Traffic Token Bucket's criteria, but does meet the Excess Traffic Token Buckets criteria, it is marked as orange.
- If a green packet does not meet the Committed Traffic Token Bucket's criteria, nor the Excess Traffic Token Buckets criteria, it is marked as red.

In addition to the RFC units, the API proposes to count either per-bytes or per-packets, and to rely or not on current packet color (color-aware or color-blind).

Algorithm

The algorithm is as follows, with $T_c(t)$ and $T_e(t)$ the current number of tokens for the Committed Traffic Token Bucket and Excess Traffic Token Bucket respectively:

- When a green packet of size B (B = 1 in case of per-packets counting) arrives at time t, then
  - if $T_c(t)$ - B > 0, the packet is marked green, and $T_c(t)$ is decremented by B
  - else if $T_e(t)$ - B > 0, the packet is marked yellow, and $T_e(t)$ is decremented by B
  - else the packet is marked red.
- When a yellow packet of size B arrives at time t, then
  - if $T_e(t)$ - B > 0, the packet is marked yellow, and $T_e(t)$ is decremented by B
  - else the packet is marked red.
- In color-blind operation, the algorithm assumes that the packet is marked green.

Features implemented in the fast path using the traffic conditioner API:

- Rate limiting of exceptions packets (MCORE_TC_ERL option)
- Rate limiting of packets received(/sent) on(/from) the specified interface (both physical and virtual interfaces)
- Conditional rate limiting using filters

Double policer case

If 2 sets of parameters are defined (using tc-policer-add), then this is a policer with a double condition.

In that case both trTCM must return a color other than red for the packet to be accepted. If the packet is accepted, then the credits of both trTCMs are consumed. If not, no credit is consumed. The color returned by this double condition trTCM is the highest of the 2."

ERL in Details

Fast Path QoS - Exception Rate Limitation is a feature of *Fast Path QoS* that limits the rate of packets sent as exceptions from the fast path to Linux.

Features

- Exception Rate Limitation to limit the rate of packets sent as exceptions from the fast path to Linux

When the MCORE_TC_ERL option is enabled, the fast path limits the rate of exceptions sent to the slow path. The rate can be configured by "fp-cli tc-erl-add" command.

A priority is associated with each exception: low, medium and high. The pass or drop action depends on this priority and the packet color, as marked by the QoS marker:

- If exception priority is low, green packets pass and yellow or red packets are dropped,
- If exception priority is medium, green and yellow packets pass, and red packets are dropped,
- If exception priority is high, packets pass,
- If exception priority is unknown, packets are dropped.

Whenever a packet is found to be an exception, a class value (8 bits) is associated with it:

- The two most significant bits define the priority of the exception (3 for high priority, 2 for medium, 1 for low, and 0 for unknown),
- The six last significant bits define the type of the exception (ARP is required; IKE negotiation is needed, etc.).

The default setting is low priority for all exceptions type, so that a simple token bucket is used to let packets go through or to drop them.

Dependencies

6WINDGate modules

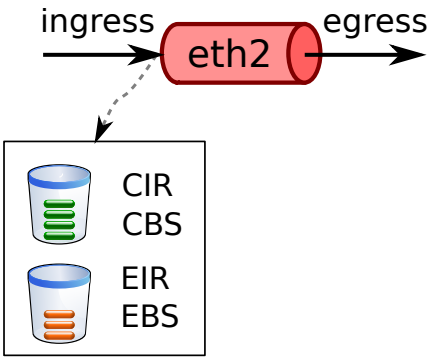
- *Fast Path Baseline*
- *Fast Path QoS*

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

Configuring a traffic conditioner attached to an interface in the fast path

TC can be set per interface and per direction. The rate limiting can be applied separately for incoming and outgoing packets on a given interface.



The fast path implements a proxy over the NETFPC channel to allow fp-cli to interact with the FPN-SDK API.

tc-iface-add

Configure a traffic conditioner attached to an interface:

```
# tc-iface-add <iface> ingress|egress <CIR> <CBS> <EIR> <EBS> [G|M|K]pps|bps
```

Parameters

<iface> Interface name, physical interface or virtual interface.

ingress|egress Direction.

<CIR> CIR. Expressed in:

- multiples of BPS (bits/second)
- multiples of PPS (packets/second)

<CBS> CBS. Expressed in:

- bytes
- packets

A committed depth of 0 disables a traffic conditioner.

<EIR> EIR. Expressed in the same unit as CIR.

<EBS> EBS. Expressed in the same unit as EIR.

[G|M|K]pps|bps

Unit and multiplier used for CIR, CBS, EIR and EBS.

- pps means that values are expressed in terms of packets:
  - rates are multiples of PPS (CIR and EIR)
  - burst sizes are in packets (CBS and EBS)
- bps means that values are expressed in terms of bits.
  - rates are multiples of BPS (CIR and EIR)
  - burst sizes are in bytes (CBS and EBS)
- G, M and K multipliers apply to rates (CIR and EIR). They do not apply to burst sizes (CBS and EBS).
  - multipliers are powers of 1000 (K=1000, M=1000^2, G=1000^3)

tc-iface-del

Delete a traffic conditioner from an interface:

```
# tc-iface-del <iface> ingress|egress
```

Parameters

<iface> Interface name, physical interface or virtual interface.

ingress|egress Direction.

tc-iface

Display traffic conditioner rules configurations:

```
# tc-iface [<iface> ingress|egress]
```

Parameters

<iface> Interface name, physical interface or virtual interface.

ingress|egress Direction.

tc-iface-stats

Display traffic conditioner rules statistics:

```
# tc-iface-stats [<iface> ingress|egress]
```

Parameters

<iface> Interface name, physical interface or virtual interface.

ingress|egress Direction.

tc-iface-json

Display configured traffic conditioners and their statistics in json format:

```
tc-iface-json [vrfid all|<vrfid>]
```

Parameters

<vrfid> Specifies the vrf of interfaces whose traffic conditioner will be displayed. Default 0. `all` dumps all vrfs.

tc-iface-stats-reset

Reset traffic conditioner statistics:

```
tc-iface-stats-reset [<iface>]
```

Parameters

<iface> Interface name, physical interface or virtual interface.

Examples

- Limit the bandwidth of traffic received on `eth2_0` to 4000 KBPS (kilobits/sec = 1000 bits/second) with a maximum burst size of 512000 bytes:

```
tc-iface-add eth2_0 ingress 4000 512000 0 0 Kbps
```

  - The CIR is set to 4000*1000 BPS = 4000000 BPS.
  - The CBS is set to 512000 bytes = 4096000 bits.
  - The EIR is set to 0 bits.
  - The EBS is set to 0 bytes.

- Display traffic conditioner rules for a single traffic conditioner:

```
tc-iface eth2_0
```

- Display all configured traffic conditioners:

```
tc-iface
```

- Display statistics for a single traffic conditioner:

```
tc-iface-stats eth2_0
```

- Display all configured traffic statistics:

```
tc-iface-stats
```

- Display configured traffic conditioners and their statistics:

```
tc-iface-json [vrfid all|VRFID]

If no vrfid is specified, only traffic conditioners in vrfid 0 will be
displayed.
```

- Reset statistics on a single traffic conditioner:

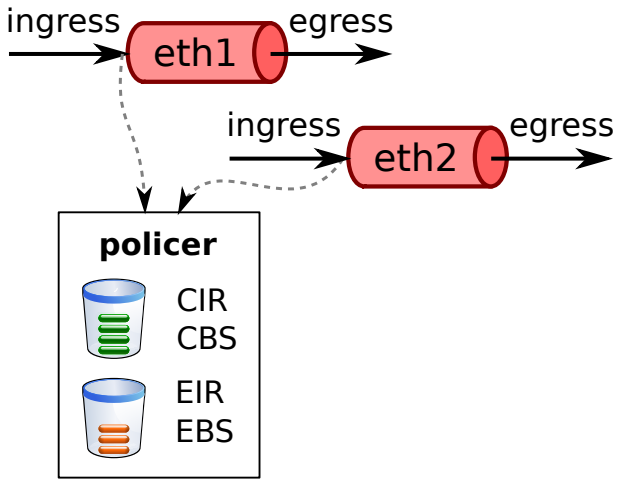
```
tc-iface-stats-reset eth2_0
```

- Reset all statistics traffic conditioner:

```
tc-iface-stats-reset
```

Configuring a traffic conditioner policer shared by several interfaces in the fast path

A traffic conditioner can be shared between several interfaces by creating a shared policer. The traffic flowing through the group of interfaces will consume tokens in the same traffic conditioner.



tc-policer-add

Create a shared policer:

```
tc-policer-add <name> <CIR> <CBS> <EIR> <EBS> [G|M|K]pps|bps [<CIR2> <CBS2> <EIR2>
↪<EBS2> [G|M|K]pps|bps]
```

Parameters

<name>

Policer name.

Other parameters are identical to those of the *tc-iface-add* command.

tc-iface-bind

Bind an interface ingress or egress to the policer:

```
tc-iface-bind <iface> ingress|egress <name>
```

Parameters

<iface> Interface name, physical interface or virtual interface.

ingress|egress Direction.

<name> Policer name.

tc-iface-del

Unbind an interface ingress or egress from a policer:

```
tc-iface-del <iface> ingress|egress
```

The same command *tc-iface-del* is used regardless whether the interface is attached to a standalone traffic conditioner or bound to a shared policer. If it is a standalone traffic conditioner, it is deleted. If it is a shared policer, the interface is unbound from the policer, but the policer itself is not deleted.

tc-policer-del

Delete a shared policer. All interfaces bound to this policer are unbound:

```
tc-policer-del <name>
```

tc-policer

Display shared policer configurations:

```
tc-policer [<name>]
```

tc-policer-stats

Display shared policer statistics:

```
tc-policer-stats [<name>]
```

tc-policer-json

Display shared policer configurations and their statistics in json format:

```
tc-policer-json
```

tc-policer-stats-reset

Reset policer statistics:

```
tc-policer-stats-reset [<name>]
```

Examples

```

```

- Limit the bandwidth of traffic received on `eth1_0` and `eth2_0` to 4000 KBPS with a maximum burst size of 512000 bytes, by sharing policer `policer1`:

```
tc-policer-add policer1 4000 512000 0 0 Kbps

tc-iface-bind eth1_0 ingress policer1
tc-iface-bind eth2_0 ingress policer1
```

- Limit the bandwidth of traffic received on `eth1_0` to 6000 KBPS with a maximum burst size of 768000 bytes, or 360 pps with a maximum burst size of 360 packets for policer `policer2`:

```
tc-policer-add policer2 6000 768000 0 0 Kbps 360 360 0 0 pps
tc-iface-bind eth1_0 ingress policer2
```

- Display traffic conditioner rules for policer `policer1`:

```
tc-policer policer1
```

- Display statistics for policer `policer1`:

```
tc-policer-stats policer1
```

- Display shared policer configurations and their statistics in json format:

```
tc-policer-json
```

- Display traffic conditioner rules for interfaces `eth1_0` and `eth2_0`:

```
tc-iface eth1_0
tc-iface eth2_0
```

- Display statistics for interfaces `eth1_0` and `eth2_0`:

```
tc-iface-stats eth1_0
tc-iface-stats eth2_0
```

- Unbind `eth1_0` ingress from policer `policer1`:

```
tc-iface-del eth1_0 ingress
```

  `eth2_0` remains attached to `policer1`.

Configuring traffic conditioner filters in the fast path

A filtered traffic conditioner is applied only for packets that match a filter: the packet is masked at a specified offset, and compared with a value. If they are the same, the packet matches the filter.

For each direction (ingress or egress), an interface has a table a filtered traffic conditioner that can be set. The filters are evaluated sequentially. If a filter entry is unset, the next ones are ignored. When a packet matches a filter matches, the associated policer is selected for this packet. If no filter match, the interface default policer is selected, if any.

The filters are created independently and can be shared between several filtered traffic conditioner.

tc-filter-add

Create a shared filter:

```
tc-filter-add <name> [not] <offset> <hex-mask> <hex-value>
```

Parameters

<name> Filter name.

not Invert the filter.

<offset> Offset in the packet where the filter should be applied.

<hex-mask> Hexadecimal mask applied to packet data.

<hex-value> Expected value after mask is applied.

tc-filter-del

Delete a shared filter.

```
tc-filter-del <name>
```

<name> Filter name.

tc-filter

Display shared filter configurations:

```
tc-filter [<name>]
```

<name> Filter name.

If name is not specified, all filters are displayed.

tc-filter-stats

Display shared filter statistics:

```
tc-filter-stats [<name>]
```

<name> Filter name.

If name is not specified, all filter statistics are displayed.

tc-filter-json

Display shared filter configurations and their statistics in json format:

```
tc-filter-json
```

tc-filter-stats-reset

Reset filter statistics:

```
tc-filter-stats-reset [<name>]
```

tc-filter-attach

Attach a filter and a policer to the interface filter table.

```
tc-filter-attach <iface> ingress|egress <index> <filter-name> <policer-name>
```

<iface> The name of the interface.

<index> The index of the filter in the table.

<filter-name> The name of the existing filter.

<policer-name> The name of the existing policer.

tc-filter-detach

Detach a filter and a policer from an interface.

```
tc-filter-detach <iface> ingress|egress <index>
```

<iface> The name of the interface.

<index> The index of the filter in the table.

tc-filter-iface

Display filters attached to an interface.

```
tc-filter-iface [<iface>]
```

<iface> The name of the interface.

If the interface name is not specified, all interfaces are displayed.

tc-filter-iface-json

Display filters attached to interfaces in json format.

```
tc-filter-iface [vrfid <vrfid>|all]
```

<vrfid> The vrf of interfaces whose traffic conditioner filters will be displayed. If not specified, display the ones from current vrf.

Examples

- Limit the bandwidth of broadcast traffic received on `eth1_0` and `eth2_0` to 4000 Kbps with a maximum burst size of 512000 bytes.

  Limit traffic from a specific mac address `00:09:C0:10:10:10` to 8000 Kbps with a maximum burst size of 1024000 bytes.

  Limit the rest of the traffic to 100 Mbps (megabits/sec = 1000^2 bits/second) with a maximum burst size of 12800000 bytes.

```
tc-policer-add broadcast-pol 4000 512000 0 0 Kbps
tc-policer-add src-mac-pol 8000 1024000 0 0 Kbps
tc-policer-add default-pol 100 12800000 0 0 Mbps
tc-filter-add broadcast-filt 0 ffffffffffff ffffffffffff
tc-filter-add src-mac-filt 6 ffffffffffff 0009C0101010

tc-filter-attach eth1_0 ingress 0 broadcast-filt broadcast-pol
tc-filter-attach eth2_0 ingress 0 broadcast-filt broadcast-pol
tc-filter-attach eth1_0 ingress 1 src-mac-filt src-mac-pol
tc-filter-attach eth2_0 ingress 1 src-mac-filt src-mac-pol
tc-iface-bind eth1_0 ingress default-pol
tc-iface-bind eth2_0 ingress default-pol
```

- Display filter `broadcast-filt`:

```
tc-filter broadcast-filt
```

- Display statistics for filter `broadcast-filt`:

```
tc-filter-stats broadcast-filt
```

- Display shared filter configurations and their statistics in json format:

```
tc-filter-json
```

- Display filtered traffic conditioner for interfaces `eth1_0` and `eth2_0`:


```
tc-filter-iface eth1_0
tc-filter-iface eth2_0
```

- Unbind `eth1_0` ingress from filters and policer:

```
tc-filter-detach eth1_0 ingress 0
tc-filter-detach eth1_0 ingress 1
tc-iface-del eth1_0 ingress
```

`eth2_0` remains attached to the filtered policers `broadcast-pol` and `src-mac-pol`, and to its default policer `default-pol`.

Configuring a flow-based traffic conditioner rule in the fast path

The rate limiting can be applied for packets which match an ip flow. The ip flow is defined by interface, source ip, destination ip and ip protocol.

For physical interfaces, it is possible to rate limit traffic going through an interface and all logical interfaces created on top. It can be used for example to provide a global rate limiter for different VLAN declared on top of the same physical interface. An option is also given to rate-limit traffic without taking into account logical interfaces built on top of a physical interface (e.g. rate limit untagged traffic on a port where some VLANs are defined).

tc-flow-add

Description

Add a flow-based traffic conditioner rule.

Synopsis

```
tc-flow-add <iface> ingress|egress <addr src> <addr dst> <tos>|any <ip proto>|any
            <CIR> <CBS> <EIR> <EBS> [G|M|K]pps|bps <priority>
```

Parameters

<iface> interface name, physical interface or virtual interface.

ingress|ingress_all|egress Direction. The ingress_all direction can be set only on a physical interface to apply the rate limiter to the physical interface and all virtual interfaces built on top. It is not possible, for the same flow, to add an ingress and an ingress_all with different values of rate limiting.

<addr src> Source ip address, a host ip or a subnet(ADDRESS/MASK), 0.0.0.0/0 means any ip address. For example: 192.168.1.2 or 192.168.1.0/24.

<addr dst> Destination ip address, a host ip or a subnet(ADDRESS/MASK), 0.0.0.0/0 means any ip address. For example: 192.168.1.2 or 192.168.1.0/24.

<tos>|any ToS value, "any" means any DSCP value.

<ip proto>|any IP protocol number, "any" means any IP protocol.

<CIR> CIR. Expressed in:

- multiples of BPS
- multiples of PPS

<CBS> CBS. Expressed in:

- bytes
- packets

A committed depth of 0 disables a flow-based traffic conditioner rule.

<EIR> EIR. Expressed in the same unit as CIR.

<EBS> EBS. Expressed in the same unit as CBS.

[G|M|K]pps|bps

Unit and multiplier used for CIR, CBS, EIR and EBS.

- pps means that values are expressed in terms of packets:
  - rates are multiples of PPS (CIR and EIR)
  - burst sizes are in packets (CBS and EBS)
- bps means that values are expressed in terms of bits.
  - rates are multiples of BPS (CIR and EIR)
  - burst sizes are in bytes (CBS and EBS)
- G, M and K multipliers apply to rates (CIR and EIR). They do not apply to burst sizes (CBS and EBS).
  - multipliers are powers of 1000 (K=1000, M=1000^2, G=1000^3)

<priority>

Priority of the flow-based traffic conditioner rule among others, it is meant to allow priority to ensure more precise flow to be matched first, and the more generic flows after, for example:

```
tc-flow-add eth0 ingress 1.1.1.0/24 2.2.2.2 any any 10 10 10 10 pps 1
tc-flow-add eth0 ingress 1.1.1.0/24 2.2.2.0/24 any any 20 20 20 20 pps 2
```

Without priority second rule could be matched before the first one, even through we specifically try to reach the specified host.

Example

Limit the bandwidth of traffic received on `eth2_0` from 10.24.3.92 to 10.22.3.91 to 4000 KBPS with a maximum burst size of 512000 bytes:

```
<fp-0> tc-flow-add eth2_0 ingress 10.24.3.92 10.22.3.91 any any 4000 512000 0 0 Kbps 1
```

- The CIR is set to 4000*1000 BPS = 4000000 BPS.
- The CBS is set to 512000 bytes = 4096000 bits.
- The EIR is set to 0 bits.
- The EBS is set to 0 bytes.

tc-flow-del

Description

Delete the specified flow-based traffic conditioner, either by ID or by flow.

Synopsis

```
tc-flow-del id <id>
or
tc-flow-del <iface> ingress|egress <addr src> <addr dst> <tos>|any <ip proto>|any
```

Parameters

<id> Each tc flow rule has an ID. The id number can be shown by `tc-flow` command.

<iface> interface name, physical interface or virtual interface.

ingress|ingress_all|egress Direction.

<addr src> Source ip address, a host ip or a subnet(ADDRESS/MASK), 0.0.0.0/0 means any ip address. For example: 192.168.1.2 or 192.168.1.0/24.

<addr dst> Destination ip address, a host ip or a subnet(ADDRESS/MASK), 0.0.0.0/0 means any ip address. For example: 192.168.1.2 or 192.168.1.0/24.

<tos>|any ToS value, “any” means any DSCP value.

<ip proto>|any IP protocol number, “any” means any IP protocol.

tc-flow-add6

Description

Add an IPv6 flow-based traffic conditioner rule.

Synopsis

```
tc-flow-add6 <iface> ingress|ingress_all|egress <addr src> <addr dst> <tc>|any <next␣
↪header>|any
            <CIR> <CBS> <EIR> <EBS> [G|M|K]pps|bps <priority>
```

Parameters

<iface> interface name, physical or virtual interface.

ingress|egress Direction. The ingress_all direction can be set only on a physical interface to apply the rate limiter to the physical interface and all virtual interfaces built on top. It is not possible, for the same flow, to add an ingress and an ingress_all with different values of rate limiting.

<addr src> Source ip address, a host ip or a subnet(ADDRESS/MASK).

`::` means any IPv6 address.

Example: `2001:DB8:1::/48`.

<addr dst> Destination ip address, a host ip or a subnet(ADDRESS/MASK).

`::` means any IPv6 address.

Example: `2001:DB8:2::/48`.

<tc>|any Traffic Class of the flow, “any” meaning any class.

<next header>|any Next header protocol, “any” means any protocol.

<CIR> CIR. Expressed in:

- multiples of BPS
- multiples of PPS

<CBS> CBS. Expressed in:

- bytes
- packets

A committed depth of 0 disables a flow-based traffic conditioner rule.

<EIR> EIR. Expressed in the same unit as CIR.

<EBS> EBS. Expressed in the same unit as CBS.

[G|M|K]pps|bps

Unit and multiplier used for CIR, CBS, EIR and EBS.

- pps means that values are expressed in terms of packets:
  - rates are multiples of PPS (CIR and EIR)
  - burst sizes are in packets (CBS and EBS)
- bps means that values are expressed in terms of bits.
  - rates are multiples of BPS (CIR and EIR)
  - burst sizes are in bytes (CBS and EBS)
- G, M and K multipliers apply to rates (CIR and EIR). They do not apply to burst sizes (CBS and EBS).
  - multipliers are powers of 1000 (K=1000, M=1000^2, G=1000^3)

<priority> Priority of the flow-based traffic conditioner rule among others, it is meant to allow priority to ensure more precise flow to be matched first, and the more generic flows after.

Example

Limit the bandwidth of traffic received on `eth2_0` from `2001:DB8:1::1` to `2001:DB8:2::1` to 4000 KBPS with a maximum burst size of 512000 bytes:

```
<fp-0> tc-flow-add6 eth2_0 ingress 2001:DB8:1::1 2001:DB8:2::1 any any 4000 512000 0 0␣
↪Kbps 1
```

- The CIR is set to 4000*1000 BPS = 4000000 BPS.
- The CBS is set to 512000 bytes = 4096000 bits.
- The EIR is set to 0 bits.
- The EBS is set to 0 bytes.

tc-flow-del6

Description

Delete the specified IPv6 flow-based traffic conditioner, either by ID or by flow.

Synopsis

```
tc-flow-del6 id <id>
or
tc-flow-del6 <iface> ingress|egress <addr src> <addr dst> <tc>|any <next header>|any
```

Parameters

<id> Each tc flow rule has an ID. The id number can be shown by `tc-flow` command.

<iface> interface name, physical or virtual interface.

ingress|ingress_all|egress Direction.

<addr src> Source ip address, a host ip or a subnet(ADDRESS/MASK).

`::` means any IPv6 address.

Example: `2001:DB8:1::/48`.

<addr dst> Destination ip address, a host ip or a subnet(ADDRESS/MASK).

`::` means any IPv6 address.

Example: `2001:DB8:2::/48`.

<tc>|any Traffic Class of the flow, “any” meaning any class.

<next header>|any

Next header protocol, “any” means any protocol.

tc-flow

Description

List one or all configured flow-based traffic conditioner.

Synopsis

```
tc-flow [<iface>] [ingress|egress]
```

Parameters

<iface> Interface name. Optional. If interface is set, only rules on this interface are displayed.

ingress|egress Direction. Optional. If direction is set, only rules matching the direction are displayed.

Example

```
<fp-0> tc-flow
Ingress TC: 1 rules
1: eth2_0 ingress 10.24.3.92/32 10.22.3.91/32 any 1 priority 1 (IPv4)
     CIR 4 Mbps
     CBS 512000
     EIR 0 bps
     EBS 0
Egress  TC: 0 rules
```

tc-flow-stats

Description

Dump statistics of the specified flow-based traffic conditioner. The packet/byte number of the 3 marked colors(Green/Yellow/Red) are displayed.

Synopsis

```
tc-flow-stats id <id>
```

Parameters

ID Each tc flow rule has an ID. The id number can be shown by `tc-flow` command.

Example

```
<fp-0> tc-flow-stats id 1
Green  19940 packets 1674960 bytes
Yellow 0 packets 0 bytes
Red    114782 packets 9641688 bytes
```

tc-flow-stats-reset

Description

Reset the statistics of the specified flow-based traffic conditioner. The packet/byte number of the 3 marked colors will be set to 0.

Synopsis

```
tc-flow-stats-reset id <id>
```

Parameters

ID Each tc flow rule has an ID. The id number can be shown by `tc-flow` command.

Providing options

There are 3 parameters provided by this module:

- `hash-order` is the order of the hash table (1 << hash-order)
- `max-flows` is the maximum number of flows.
- `timeout` is the idle duration (in seconds) before one flow hash node is deleted.

You can dynamically set the 3 parameters when starting the fast path:

- Hash order with option `--mod-opt=tc-flow:--hash-order=<hash-order>`. Default is 10.
- Maximum number of flows with option `--mod-opt=tc-flow:--max-flows=<flow-number>`. Default is 10000.
- Timeout with option `--mod-opt=tc-flow:--timeout=<timeout-value>`. Default is 5.

Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

---

Configuring the global exception rate limitation

Setting the global exception rate limitation

Synopsis

```
tc-erl-add <CIR> <CBS> <EIR> <EBS> [G|M|K]pps|bps
```

Parameters

<CIR> CIR. Expressed in:

- multiples of BPS
- multiples of PPS

<CBS> CBS. Expressed in:

- bytes
- packets

A committed depth of 0 disables a traffic conditioner rule.

<EIR> EIR. Expressed in the same unit as CIR.

<EBS> EBS. Expressed in the same unit as CBS.

[G|M|K]pps|bps

Unit and multiplier used for CIR, CBS, EIR and EBS.

- pps means that values are expressed in terms of packets:
  - rates are multiples of PPS (CIR and EIR)
  - burst sizes are in packets (CBS and EBS)
- bps means that values are expressed in terms of bits.
  - rates are multiples of BPS (CIR and EIR)
  - burst sizes are in bytes (CBS and EBS)
- G, M and K multipliers apply to rates (CIR and EIR). They do not apply to burst sizes (CBS and EBS).
  - multipliers are powers of 1000 (K=1000, M=1000^2, G=1000^3)

Example

```
<fp-0> tc-erl-add 4000 512000 0 0 Kbps
```

Deleting the global exception rate limitation

Synopsis

```
tc-erl-del
```

Example

```
<fp-0> tc-erl-del
```

Displaying the global exception rate limitation

Synopsis

```
tc-erl
```

Example

```
<fp-0> tc-erl
tc-erl rule:
   CIR = 4095996 bps
   CBS = 512000
   EIR = 0 bps
   EBS = 0
```

Displaying the global exception rate limitation statistics

Synopsis

```
tc-erl-stats
```

Example

```
<fp-0> tc-erl-stats
tc-erl statistics:
   Green  0 packets 0 bytes
   Yellow 0 packets 0 bytes
   Red    0 packets 0 bytes
```

Resetting the global exception rate limitation statistics

Synopsis

```
tc-erl-stats-reset
```

Example

```
<fp-0> tc-erl-stats-reset
```

Configuring exception rate limitation per input port

ERL (Exception Rate Limitation) rules may be attached to an input port. The rate limitation is applied to exceptions based on their arrival network port (not on their current physical or logical interface).

Critical control plane traffic (ARP, ICMP, routing protocols, IKE...) bypasses this kind of ERL rule.

The global ERL rule, if any, is also verified and applied.

Setting an exception rate limitation on a port

Synopsis

```
tc-erl-port-add <port_name> <CIR> <CBS> <EIR> <EBS> [G|M|K]pps|bps
```

Parameters

<port_name> Port name (interface name given to the port). This interface name may reference a physical port or vport, but not a logical interface.

<CIR> CIR. Expressed in:

- multiples of BPS
- multiples of PPS

<CBS> CBS. Expressed in:

- bytes
- packets

A committed depth of 0 disables a traffic conditioner rule.

<EIR> EIR. Expressed in the same unit as CIR.

<EBS> EBS. Expressed in the same unit as CBS.

[G|M|K]pps|bps

Unit and multiplier used for CIR, CBS, EIR and EBS.

- pps means that values are expressed in terms of packets:
  - rates are multiples of PPS (CIR and EIR)
  - burst sizes are in packets (CBS and EBS)
- bps means that values are expressed in terms of bits.
  - rates are multiples of BPS (CIR and EIR)
  - burst sizes are in bytes (CBS and EBS)
- G, M and K multipliers apply to rates (CIR and EIR). They do not apply to burst sizes (CBS and EBS).
  - multipliers are powers of 1000 (K=1000, M=1000^2, G=1000^3)

Example

```
<fp-0> tc-erl-port-add eth0 4000 512000 0 0 Kbps
```

Note: Although the port is referenced by an interface name, the rule is attached to the port itself, and will remain attached even if the interface name or vrfid change.

Deleting an exception rate limitation on a port

Synopsis

```
tc-erl-port-del <port_name>
```

Parameters

<port_name> Port name.

Example

```
<fp-0> tc-erl-port-del eth0
```

Displaying per-port exception rate limitations

Synopsis

```
tc-erl-port [<port_name>]
```

Parameters

<port_name> Optional port name. All ports if unspecified.

Example

```
<fp-0>  tc-erl-port
eth0-vrf0:
   CIR = 4 Mbps
   CBS = 512000
   EIR = 0 bps
   EBS = 0
eth1-vrf0:
   CIR = 100 pps
   CBS = 4
   EIR = 20 pps
   EBS = 2
```

Displaying statistics of per-port exception rate limitation

Synopsis

```
tc-erl-port-stats [<port_name>]
```

Parameters

<port_name> Optional port name. All ports if unspecified.

Example

```
<fp-0> tc-erl-port-stats eth0
eth0-vrf0:
   Green  2543 packets 3838657 bytes
   Yellow 0 packets 0 bytes
   Red    488 packets 738832 bytes
```

```
<fp-0> tc-erl-port-stats
eth0-vrf0:
   Green  2543 packets 3838657 bytes
   Yellow 0 packets 0 bytes
   Red    488 packets 738832 bytes
eth1-vrf0:
   Green  3 packets 675 bytes
   Yellow 0 packets 0 bytes
   Red    0 packets 0 bytes
```

Resetting statistics of per-port exception rate limitation

Synopsis

```
tc-erl-port-stats-reset [<port_name>]
```

Parameters

<port_name> Optional port name. All ports if unspecified.

Example

```
<fp-0> tc-erl-port-stats-reset eth0
```

```
<fp-0> tc-erl-port-stats-reset
```

Configuring exception rate limitation DSCP classes

Setting an exception rate limitation DSCP class

Synopsis

```
tc-erl-dscp-class-set <dscp_value> <dscp_class>
```

Parameters

<dscp_value> TOS (Type of Service) value to be associated with a DSCP class.

Can be expressed in base 10 or 16, if preceded with '0x'.

<dscp_class> DSCP class of the value.

Example

```
<fp-0> tc-erl-dscp-class-set 0x20 3
<fp-0> tc-erl-dscp-class-set 34 3
```

Resetting an exception rate limitation DSCP class

Synopsis

```
tc-erl-dscp-class-reset [<dscp_class>]
```

Parameters

<dscp_class> Optional DSCP class to be reset.

If none is provided, all DSCP classes are reset.

Example

```
<fp-0> tc-erl-dscp-class-reset 3
```

Displaying an exception rate limitation DSCP class

Synopsis

```
tc-erl-dscp-class [<dscp_class>]
```

Parameters

<dscp_class> Optional DSCP class to be displayed.

If none is provided, all DSCP classes are displayed.

Example

```
<fp-0> tc-erl-dscp-class 3
<fp-0> tc-erl-dscp-class
```

Adding an exception rate limitation DSCP filter

Synopsis

```
tc-erl-dscp-cp-filter-add <if_name> <dscp_class>
```

Parameters

<if_name> Interface name, physical or virtual.

Each interface uses a filter slot, which are limited to 16.

<dscp_class> The DSCP class to be prioritized. Any packet having a DSCP value associated with this class and originating from this interface will be matched, marking this packet as high-priority.

The values any or all can be used to match all packets originating from this interface.

Example

```
<fp-0> tc-erl-dscp-cp-filter-add eth0 3
<fp-0> tc-erl-dscp-cp-filter-add eth1 any
```

Deleting an exception rate limitation DSCP filter

Synopsis

```
tc-erl-dscp-cp-filter-del <if_name> <dscp_class>
```

Parameters

<if_name> Interface name, physical or virtual.

<dscp_class> The DSCP class currently matched.

The values any or all can be used to completely remove the filter from this interface, freeing a slot for a new filter to be inserted eventually.

Example

```
<fp-0> tc-erl-dscp-cp-filter-del eth1 3
<fp-0> tc-erl-dscp-cp-filter-del eth0 all
```

Displaying an exception rate limitation DSCP filter

Synopsis

```
tc-erl-dscp-cp-filter [if_name]
```

Parameters

[if_name] Optional interface name, physical or virtual. If provided, only the CP filter associated with this interface is displayed.

Example

```
<fp-0> tc-erl-dscp-cp-filter eth1
<fp-0> tc-erl-dscp-cp-filter
```

Adding an exception rate limitation DSCP traffic conditioner rule

Synopsis

```
tc-erl-if-dscp-add <if_name> <dscp_class> <CIR> <CBS> <EIR> <EBS> [G|M|K]pps|bps
```

Parameters

Each traffic conditioner rule uses a slot, which are limited to 16.

<if_name> Interface name, physical or virtual.

<dscp_class> The DSCP class to be rate-limited. If set, only packets having DSCP value of this class will be matched by the associated traffic conditioner rule. If set, any packet having a DSCP value associated with this class and originating from this interface will be matched, and the traffic conditioner rule will be applied.

The values `any` or `all` can be used to signify that any DSCP value (even `0`) would match the traffic conditioner rule.

Several traffic conditioner rules can be set for the same interface, if their associated DSCP classes are different. In some cases, when the specified DSCP class is `any` or `all`, then the most specific rule applies first (the one with a specific DSCP class), then the catch-all rule will match (DSCP class is `any` or `all`).

If a new rule is given for an (if_name, DSCP class) pair, the old one is updated with the new parameters.

<CIR> CIR. Expressed in:

- multiples of BPS

- multiples of PPS

<CBS> CBS. Expressed in:

- bytes
- packets

A committed depth of 0 disables a traffic conditioner rule.

<EIR> EIR. Expressed in the same unit as CIR.

<EBS> EBS. Expressed in the same unit as CBS.

[G|M|K]pps|bps

Unit and multiplier used for CIR, CBS, EIR and EBS.

- pps means that values are expressed in terms of packets:
  - rates are multiples of PPS (CIR and EIR)
  - burst sizes are in packets (CBS and EBS)
- bps means that values are expressed in terms of bits.
  - rates are multiples of BPS (CIR and EIR)
  - burst sizes are in bytes (CBS and EBS)
- G, M and K multipliers apply to rates (CIR and EIR). They do not apply to burst sizes (CBS and EBS).
  - multipliers are powers of 1000 (K=1000, M=1000^2, G=1000^3)

Example

```
<fp-0> tc-erl-if-dscp-add eth0 any 152000 4000 0 0 Kbps
<fp-0> tc-erl-if-dscp-add eth1 3 152000 4000 0 0 Kbps
```

Deleting an exception rate limitation DSCP traffic conditioner rule

Synopsis

```
tc-erl-if-dscp-del <if_name> <dscp_class> [<CIR> <CBS> <EIR> <EBS> [G|M|K]pps|bps]
```

Parameters

<if_name> Interface name, physical or virtual.

<dscp_class> The DSCP class of the rule to be deleted.

Specifying `any` or `all` will only delete the traffic conditioner rule matching `any` or `all` DSCP class. All other rules with a specific DSCP class are kept.

All other parameters are purely optional and are only accepted for ease-of-use.

Example

```
<fp-0> tc-erl-if-dscp-del eth0 any 152000 4000 0 0 Kbps
<fp-0> tc-erl-if-dscp-del eth1 3
```

Displaying an exception rate limitation DSCP traffic conditioner rule

Synopsis

```
tc-erl-if-dscp [<if_name> [<dscp_class>]]
```

Parameters

<if_name> Interface name, physical or virtual.

If none provided, all DSCP traffic conditioner rules are displayed.

<dscp_class> The DSCP class of the rule to be displayed.

If none provided and `if_name` is given, all rules on this interface are shown.

Example

```
<fp-0> tc-erl-if-dscp
eth1-vrf0:
  DSCP = 4
   CIR = 24 Mpps
   CBS = 5000
   EIR = 0 pps
   EBS = 0
eth2-vrf0:
  DSCP = Any
```

(continues on next page)

(continued from previous page)

```
   CIR = 152 Mpps
   CBS = 4000
   EIR = 0 pps
   EBS = 0
eth2-vrf0:
  DSCP = 4
   CIR = 152 Mpps
   CBS = 4000
   EIR = 0 pps
   EBS = 0

<fp-0> tc-erl-if-dscp eth2
eth2-vrf0:
  DSCP = Any
   CIR = 152 Mpps
   CBS = 4000
   EIR = 0 pps
   EBS = 0
eth2-vrf0:
  DSCP = 4
   CIR = 152 Mpps
   CBS = 4000
   EIR = 0 pps
   EBS = 0

<fp-0> tc-erl-if-dscp eth2 any
eth2-vrf0:
  DSCP = Any
   CIR = 152 Mpps
   CBS = 4000
   EIR = 0 pps
   EBS = 0
```

Displaying exception rate limitation DSCP traffic conditioner rule statistics

Synopsis

```
tc-erl-if-dscp-stats [<if_name> [<dscp_class>]]
```

Parameters

<if_name> Interface name, physical or virtual.

If none provided, all DSCP traffic conditioner rule statistics are displayed.

<dscp_class> The DSCP class of the rule statistics to be displayed.

If none provided and `if_name` is given, all rule statistics on this interface are shown.

Example

```
<fp-0> tc-erl-if-dscp-stats
eth1-vrf0:
  DSCP = 4
   Green  0 packets 0 bytes
   Yellow 0 packets 0 bytes
   Red    0 packets 0 bytes
eth2-vrf0:
  DSCP = Any
   Green  0 packets 0 bytes
   Yellow 0 packets 0 bytes
   Red    0 packets 0 bytes
eth2-vrf0:
  DSCP = 4
   Green  0 packets 0 bytes
   Yellow 0 packets 0 bytes
   Red    0 packets 0 bytes

<fp-0> tc-erl-if-dscp-stats eth2
eth2-vrf0:
  DSCP = Any
   Green  0 packets 0 bytes
   Yellow 0 packets 0 bytes
   Red    0 packets 0 bytes
eth2-vrf0:
  DSCP = 4
   Green  0 packets 0 bytes
   Yellow 0 packets 0 bytes
   Red    0 packets 0 bytes

<fp-0> tc-erl-if-dscp-stats eth2 any
eth2-vrf0:
  DSCP = Any
   Green  0 packets 0 bytes
```

(continues on next page)

(continued from previous page)

```
   Yellow 0 packets 0 bytes
   Red    0 packets 0 bytes
```

Resetting exception rate limitation DSCP traffic conditioner rule statistics

Synopsis

```
tc-erl-if-dscp-stats-reset [<if_name> <dscp_class>]
```

Parameters

Either no parameters should be provided, or both `if_name` and DSCP class.

<if_name> Interface name, physical or virtual.

If none provided, all DSCP traffic conditioner rule statistics are reset.

<dscp_class> The DSCP class of the rule whose statistics will be reset

Example

```
<fp-0> tc-erl-if-dscp-stats-reset
<fp-0> tc-erl-if-dscp-stats-reset eth2 any
```

Configuring exception rate limitation rules per exception class

Mapping an exception class to a generic class

Synopsis

This function will map an exception class to a generic class.

A traffic conditioner rule is configured for each generic class, which is applied to any exception mapped to this generic class.

Several exception classes can be mapped to the same generic class.

```
tc-erl-class-exc-map <exception_class> <generic_class>
```

Parameters

<exception_class> The exception class to map. Acceptable values are any from:

```
FPTUN_EXC_SP_FUNC
FPTUN_EXC_ETHER_DST
FPTUN_EXC_IP_DST
FPTUN_EXC_ICMP_NEEDED
FPTUN_EXC_NDISC_NEEDED
FPTUN_EXC_IKE_NEEDED
FPTUN_EXC_FPC
FPTUN_EXC_NF_FUNC
FPTUN_EXC_TAP
FPTUN_EXC_REPLAYWIN
FPTUN_EXC_ECMP_NDISC_NEEDED
FPTUN_EXC_VNB_TO_VNB
FPTUN_EXC_SOCKET
FPTUN_EXC_IP_PMTU
```

The `FPTUN_EXC_` prefix can be left out. The input is case-insensitive.

`icmp_needed` is strictly equivalent to `FPTUN_EXC_ICMP_NEEDED`.

<generic_class>

The generic class identifying number. Must be comprised within `1` and `127` included.

The generic class `0` is the default one, which corresponds to the global ERL traffic conditioner rule.

Example

```
<fp-0> tc-erl-class-exc-map FPTUN_EXC_ICMP_NEEDED 3
<fp-0> tc-erl-class-exc-map NDISC_NEEDED 3
<fp-0> tc-erl-class-exc-map tap 3
```

Unmapping an exception class from a generic class

Synopsis

This function resets the map from an exception class to a generic class. Essentially, it will set the generic class of this exception to the `default` generic class, unbinding this exception class from the traffic conditioner rule configured for its previous generic class.

```
tc-erl-class-exc-unmap <exception_class>
```


Parameters

<exception_class> The exception class to unmap. Acceptable values are any from:

```
FPTUN_EXC_SP_FUNC
FPTUN_EXC_ETHER_DST
FPTUN_EXC_IP_DST
FPTUN_EXC_ICMP_NEEDED
FPTUN_EXC_NDISC_NEEDED
FPTUN_EXC_IKE_NEEDED
FPTUN_EXC_FPC
FPTUN_EXC_NF_FUNC
FPTUN_EXC_TAP
FPTUN_EXC_REPLAYWIN
FPTUN_EXC_ECMP_NDISC_NEEDED
FPTUN_EXC_VNB_TO_VNB
FPTUN_EXC_SOCKET
FPTUN_EXC_IP_PMTU
```

The `FPTUN_EXC_` prefix can be left out. The input is case-insensitive.

`icmp_needed` is strictly equivalent to `FPTUN_EXC_ICMP_NEEDED`.

Example

```
<fp-0> tc-erl-class-exc-unmap FPTUN_EXC_ICMP_NEEDED
<fp-0> tc-erl-class-exc-unmap NDISC_NEEDED
<fp-0> tc-erl-class-exc-unmap tap
```

Displaying the exception to generic class map

Synopsis

This function will show the current configured mapping from exception classes to generic classes.

```
tc-erl-class-exc [<exception_class>]
```

Parameters

<exception_class> The optional exception class to display. If not set, all exception classes are shown.

Acceptable values are any from:

```
FPTUN_EXC_SP_FUNC
FPTUN_EXC_ETHER_DST
FPTUN_EXC_IP_DST
FPTUN_EXC_ICMP_NEEDED
FPTUN_EXC_NDISC_NEEDED
FPTUN_EXC_IKE_NEEDED
FPTUN_EXC_FPC
FPTUN_EXC_NF_FUNC
FPTUN_EXC_TAP
FPTUN_EXC_REPLAYWIN
FPTUN_EXC_ECMP_NDISC_NEEDED
FPTUN_EXC_VNB_TO_VNB
FPTUN_EXC_SOCKET
FPTUN_EXC_IP_PMTU
```

The `FPTUN_EXC_` prefix can be left out. The input is case-insensitive.

`icmp_needed` is strictly equivalent to `FPTUN_EXC_ICMP_NEEDED`.

Example

```
<fp-0> tc-erl-class-exc
Exception:              Class:
FPTUN_EXC_ICMP_NEEDED : 003
FPTUN_EXC_NDISC_NEEDED: 003
FPTUN_EXC_TAP         : 003
```

Configuring a classful traffic conditioner rule

Synopsis

Each generic classes can be configured with its own traffic conditioner rule.

This traffic conditioner rule will be applied to any packets matching this generic class, such as exceptions of a class mapped to this generic class.

```
tc-erl-class-set <generic_class> <CIR> <CBS> <EIR> <EBS> [G|M|K]pps|bps
```

Parameters

<generic_class> The generic class of the traffic conditioner rule.

<CIR> CIR. Expressed in:

- multiples of BPS
- multiples of PPS

<CBS> CBS. Expressed in:

- bytes
- packets

A committed depth of 0 disables a traffic conditioner rule.

<EIR> EIR. Expressed in the same unit as CIR.

<EBS> EBS. Expressed in the same unit as CBS.

[G|M|K]pps|bps

Unit and multiplier used for CIR, CBS, EIR and EBS.

- pps means that values are expressed in terms of packets:
  - rates are multiples of PPS (CIR and EIR)
  - burst sizes are in packets (CBS and EBS)
- bps means that values are expressed in terms of bits.
  - rates are multiples of BPS (CIR and EIR)
  - burst sizes are in bytes (CBS and EBS)
- G, M and K multipliers apply to rates (CIR and EIR). They do not apply to burst sizes (CBS and EBS).
  - multipliers are powers of 1000 (K=1000, M=1000^2, G=1000^3)

Example

```
<fp-0> tc-erl-class-set 1 152000 4000 0 0 Kbps
<fp-0> tc-erl-class-set 2 152000 25600 0 0 Kbps
```

Disabling a classful traffic conditioner rule

Synopsis

A generic class traffic conditioner rule can be disabled, while the class mapping remains.

If so, the traffic will be conditioned by the global tc-erl traffic conditioner rule if it is configured, and won't be rate-limited at all otherwise.

```
tc-erl-class-reset <generic_class> [<CIR> <CBS> <EIR> <EBS> [G|M|K]pps|bps]
```

Parameters

<generic_class> The generic class of the traffic conditioner rule to disable.

All other parameters are optional and will be discarded.

Example

```
<fp-0> tc-erl-class-reset 1 152000 4000 0 0 Kbps
<fp-0> tc-erl-class-reset 2
```

Displaying one or more exception rate limitation generic traffic conditioner rule

Synopsis

```
tc-erl-class [<generic_class>]
```

Parameters

<generic_class> The (Optional) generic class of the traffic conditioner rule to display. If not set, the traffic conditioner rule of all generic classes are shown.

Example

```
<fp-0> tc-erl-class
Class 001:
   CIR = 152 Mbps
   CBS = 4000
   EIR = 0 bps
```

(continues on next page)

(continued from previous page)

```
   EBS = 0
Class 002:
   CIR = 152 Mbps
   CBS = 25600
   EIR = 0 bps
   EBS = 0
<fp-0> tc-erl-class 2
Class 002:
   CIR = 152 Mbps
   CBS = 25600
   EIR = 0 bps
   EBS = 0
```

Displaying the statistics of one or more classful traffic conditioner rule

Synopsis

```
tc-erl-class-stats [<generic_class>]
```

Parameters

<generic_class> The (Optional) generic class of the traffic conditioner rule statistics to display. If not set, the statistics of all generic classes traffic conditioner rules are shown.

Example

```
<fp-0> tc-erl-class-stats
Class 001:
   Green  0 packets 0 bytes
   Yellow 0 packets 0 bytes
   Red    0 packets 0 bytes
Class 002:
   Green  0 packets 0 bytes
   Yellow 0 packets 0 bytes
   Red    0 packets 0 bytes
<fp-0> tc-erl-class-stats 2
Class 002:
   Green  0 packets 0 bytes
   Yellow 0 packets 0 bytes
   Red    0 packets 0 bytes
```

Resetting the statistics of one or more classful traffic conditioner rule

Synopsis

```
tc-erl-class-stats-reset [<generic_class>]
```

Parameters

<generic_class> The (Optional) generic class of the traffic conditioner rule statistics to reset. If not set, the statistics of all generic classes traffic conditioner rules are reset.

Example

```
<fp-0> tc-erl-class-stats-reset 2
<fp-0> tc-erl-class-stats-reset
```

2.3.20 Fast Path QoS Advanced

Overview

Fast Path QoS Advanced provides ingress and egress Quality of Service, including:

- Classification and marking
- Metering
- Policing
- Shaping
- Scheduling
- Congestion management

Fast Path QoS Advanced implements a pipeline model with customizable distribution of worker and scheduler threads.

Fast Path QoS Advanced does not synchronize Linux TC configuration from Linux kernel; it has its own API to configure QoS classification and scheduling rules.

Features

- Pipeline model with customizable distribution of worker and scheduler threads
- Number of traffic classes only depends on memory available
- iptables-based classification (mangle) supporting Netfilter matches
- Marking: fwmark, ToS/IPP/DSCP
- Metering: token bucket, trTCM, mark ToS/IPP/DSCP
- Policing per interface (egress)[1]
- Shaping per interface (egress)
- Scheduling: Priority Queuing
- Congestion management: Taildrop

Packets are classified by netfilter rules, using any match method supported by the Fast Path Filtering IPv4 and Fast Path Filtering IPv6 modules, and labeled with a netfilter mark. A QoS classification module then maps the netfilter mark to a QoS class.

Traffic metering evaluates the traffic conformance to a traffic profile via a trTCM algorithm (**RFC 4115** (https://tools.ietf.org/html/rfc4115.html)), and takes an action according to the packet color (enqueue with optional TOS remarking, drop).

Packet scheduling: packets are submitted for output on the network interface according to a strict priority scheduling algorithm: packets queued in higher priority queues are submitted before packets of lower priority queues.

Dependencies

Hardware support

- Intel or Arm-based servers
- Physical ports

6WINDGate modules

- *Fast Path QoS*
- *Fast Path Filtering IPv4*

and optionally:

- *Fast Path Filtering IPv6*

[1] Ingress policing per interface provided by the Fast Path QoS module.

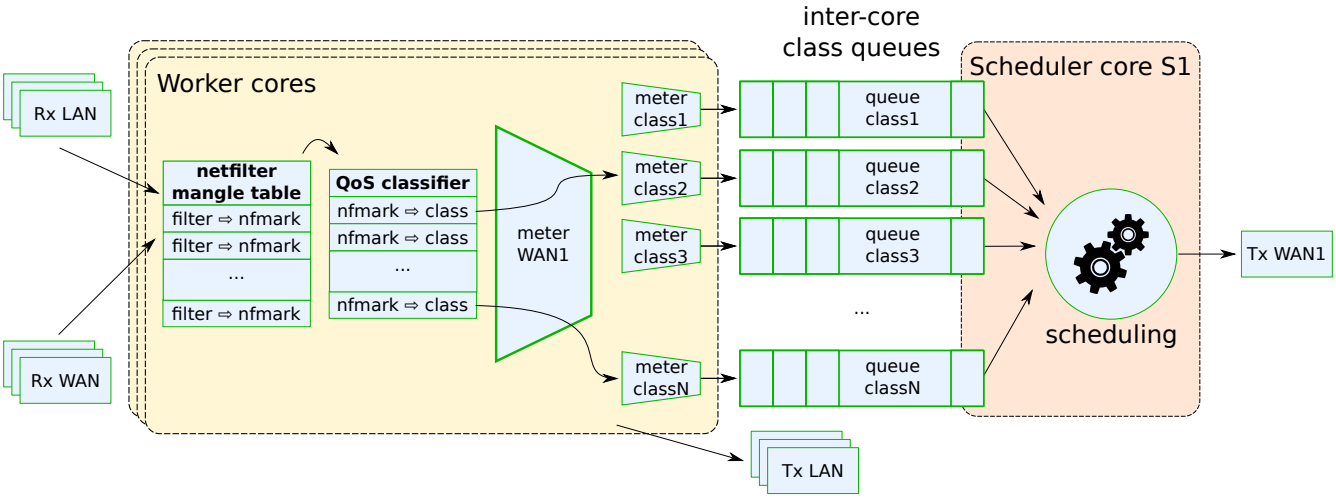
Design

Software architecture

Fast Path QoS Advanced uses computer cores dedicated to scheduling. All usual fast path processing (packet receive, routing, filtering, encapsulation, IPsec, QoS classification…) is processed by worker cores, following a run-to-completion model.

Then, if a packet needs to be sent out a physical interface with QoS enabled, it is enqueued in intercore packet queues, one per QoS class, after an optional per-interface and per-class metering stage.

The QoS class queues are polled by the scheduler cores, and packets are send on the physical network interface according to a scheduling algorithm.



Usage

The classification of packets for QoS uses the netfilter framework, which is configured via `iptables` and is synchronized from Linux to the *Fast Path Filtering IPv4* and *Fast Path Filtering IPv6* modules.

All other QoS operations are implemented in the fast path only and configured via `fp-cli`.

Fast Path startup options

Enabling QoS on a physical interface is done by configuring a scheduler on this interface. The scheduler is attached to a single fast path core.

By default, in addition to scheduling output packets, fast path cores may poll network port rx queues, perform crypto offloading for other cores, process packets originating from Linux, etc. These cores are named *worker cores*.

However, it is possible to dedicate some of the fast path cores to only perform scheduling. These cores are named *scheduler cores*. This is optional, but provides more accurate QoS guarantees.

The assignment of cores dedicated to QoS scheduling can be performed by editing the fast path configuration file.

QoS cores are a subset of the fast path cores (FP_MASK variable). Their list can be specified by setting the `-Q` option in `FPNSDK_OPTIONS` or by using the option : ${QOS_SCHEDULER_MASK:=<value>}.

Example

For example, the fast path runs on cores 1 to 6, and cores 2 and 4 are dedicated to QoS scheduling:

```
# fast-path.sh stop
# vi /etc/fast-path.env
  [...]
  FP_MASK="1-6"
  QOS_SCHEDULER_MASK="2,4"
  [...]
# fast-path.sh start
```

Module initialization

The QoS module initialization supports the following options.

`--sched-max` set maximum number of QoS schedulers (interfaces with QoS enabled)

`--filter-max` set maximum number of QoS filter rules

`--class-max` set maximum number of QoS classes

`--meter-max` set maximum number of QoS meters (max 1 per class + 1 per scheduler)

The default values for the maximum are set in the configuration. Eg. `CONFIG_MCORE_QOS_SCHED_MAX` is the default value for `--sched-max`.

Display QoS global parameters

Display worker and scheduler cores, cores with schedulers attached, and free QoS objects.

```
<fp-0> qos-global
worker cores: 1 3 5 6
scheduler cores: 2 4
cores with configured schedulers: 2:1
allocated objects (current/max):
- sched: 1/32
- class: 4/512
- filter: 0/2048
- meter: 0/512
```


`worker cores` lists cores that poll network interfaces. They also perform packet classification (netfilter rules, QoS filters), and metering. They can optionally perform scheduling if schedulers are configured on them.

`scheduler cores` lists cores that only perform scheduling.

`cores with configured schedulers` lists worker and scheduler cores on which schedulers are configured. The first number is the core id, the second one after `:` is the number of schedulers.

`allocated objects` counts the number of allocated QoS objects (schedulers, classes, filters, meters). These objects are allocated from a pool. The first number counts the allocated objects. The second one, after `/` is the pool size.

Memory considerations

The objects considered are allocated during the fast path initialization and are pre-allocated, so the memory footprint depends only on the maximum for each pool.

Below are the unitary sizes for the objects allocated for each pool. For the total memory footprint, multiply the unitary size by the maximum object number.

Unitary sizes for the object allocated in shared memory

- `sched` 76 Bytes (72 Bytes for 32bits architectures)
- `class` 68 Bytes
- `filter` 32 Bytes
- `meter` 60 Bytes

Unitary sizes for the objects allocated in the fast path memory

Without CONFIG_MCORE_FPN_GC:

- `sched` 32 Bytes (16 Bytes for 32bits architectures)
- `class` 8 Bytes (4 Bytes for 32bits architectures)
- `filter` 0 Bytes
- `meter` 8 Bytes (4 Bytes for 32bits architectures)

With CONFIG_MCORE_FPN_GC:

- `sched` 48 Bytes (24 Bytes for 32bits architectures)
- `class` 24 Bytes (12 Bytes for 32bits architectures)
- `filter` 16 Bytes (8 Bytes for 32bits architectures)
- `meter` 24 Bytes (12 Bytes for 32bits architectures)

Configure packet classification and marking

The netfilter framework is used to classify packets, optionally mark their DSCP, then set a netfilter mark. The mark will be used by the QoS filter stage to enqueue paquets in the right class queue.

All netfilter match methods and targets supported by the *Fast Path Filtering IPv4* and *Fast Path Filtering IPv6* modules can be used. The mangle table is typically used for QoS packet classification.

Netfilter rules are configured with the `iptables` Linux command.

There are basically 2 models when configuring packet classification for QoS:

- trusted: we trust the DSCP marking performed by packet originators. The typical match method used for QoS is then `dscp`.
- untrusted: we do not trust the DSCP marking performed by packet originators. The DSCP of packets must then be reset, and all match methods are liable to be used.

The typical targets used for QoS are `DSCP` and `MARK`.

Example

All packets with the EF (Expedited Forwarding) DSCP must have their netfilter mark set to `0x1`. This mark will then be used by the QoS filter stage to steer packets to the right class:

```
# iptables -t mangle -I POSTROUTING -m dscp --dscp-class EF -j MARK --set-mark 0x1
```

All UDP packets destined to 10.99.0.1 must have their DSCP set to EF:

```
# iptables -t mangle -I POSTROUTING -p udp -d 10.99.0.1 -j DSCP --set-dscp-class EF
```

Display netfilter rules in the Linux kernel:

```
# iptables -t mangle -L -v
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 1 packets, 76 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 1 packets, 76 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

(continues on next page)

(continued from previous page)

```
    0     0 DSCP       udp  --  any    any     anywhere             10.99.0.1          ␣
↪  DSCP set 0x2e
    1    76 MARK       all  --  any    any     anywhere             anywhere           ␣
↪  DSCP match 0x2e MARK set 0x1
```

Display netfilter rules in the fast path:

```
<fp-0> nf4-rules mangle
Chain PREROUTING (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source               destination

Chain INPUT (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source               destination

Chain FORWARD (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source               destination

Chain OUTPUT (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source               destination

Chain POSTROUTING (policy ACCEPT 0 packets 0 bytes)
    pkts      bytes target    prot opt  in     out    source               destination
       0          0 DSCP      udp  --  any    any    anywhere             10.99.0.1    ␣
↪         DSCP set 0xb8
       0          0 MARK      all  --  any    any    anywhere             anywhere     ␣
↪         MARK set 0x1 DSCP match 0x2e
```

Configure QoS schedulers

A scheduler may be configured for each physical network interface. Each scheduler runs on a single fast path core.

Enable and configure scheduling

Enable QoS on a physical interface and configure its scheduler.

```
qos-sched-add <ifname> (prio|dwrr) <classes> [rate <rate>] [burst <burst>] \
              [l2overhead <l2overhead>] [qsize <pkts>[,<pkts>[,...]]] \
              [priority <priority>[,<priority>[,...]]] \
              [weight <weight>[,<weight>[,...]]] [cpu <coreid>]
```

```
qos-sched-add <ifname> fuzz <classes> [rate <rate>] [burst <burst>] \
              [l2overhead <l2overhead>] [qsize <pkts>[,<pkts>[,...]]] \
```

(continues on next page)

(continued from previous page)

```
            [qrate <rate>[,<rate>[,...]]] [qburst <burst>[,<burst>[,...]]] \
            [drop <x/y>[,<x/y>[,...]]] delay <delay>[,<us>[,...]] \
            [lsize <nb_pkts>[,<nb_pkts>[,...]]] [cpu <coreid>]
```

Parameters

<ifname> physical interface on which the scheduler is configured.

prio selects the strict priority scheduling algorithm. Class identifiers and priorities range from 1 (highest priority) to <classes> (lowest priority).

dwrr selects deficit weighted round robin (DWRR) scheduling algorithm. Unlike `prio`, all classes have the same priority by default, but are given more or less scheduling attention based on their `weight`.

Since weight is used directly as a quantum, it represents the number of packet bytes of a class to process during each scheduling turn.

Specifying weight values lower than the mean packet size provides higher accuracy but may negatively affect performance as more scheduling turns are needed to achieve the desired distribution, while very large weights will cause rough scheduling patterns at the small scale, unfairness and possibly weight/priority inversion due to packet drop on the remaining classes.

When unsure about the mean packet size, a MTU-sized weight should guarantee that at least one packet will be processed during a scheduling turn.

The default weight (quantum) for DWRR classes is 1500 bytes.

A higher priority level can be given to at most one class through `priority`. Such a class is not affected by `weight` since its traffic is always processed first. This class should be limited to low volume traffic where latency matters (such as voice or control data) as it can monopolize all the available bandwidth.

Note: quantum only reflects packet data; `l2overhead` is separately taken into account during transmission and has no impact on class deficit.

To put this into perspective, 15 packets of 100 bytes each are considered equal to a single packet of 1500 bytes, although more bandwidth will typically be consumed by a higher number of packets due to `l2overhead`.

fuzz selects the fuzzer scheduling algorithm. All classes have equal priorities and a fixed weight which cannot be configured and thus these options are not supported by the fuzzer scheduler. The classes are scheduled in a round robin manner and are entertained until their fixed weight is consumed.

<classes> number of classes (input queues) to allocate for scheduler. While possible values depend on the algorithm specified by the previous parameter, minimum is typically 1 and there is usually no upper bound other than memory constraints (see `CONFIG_MCORE_QOS_CLASS_MAX`).

rate <rate> (optional) link rate in BPS, with an optional multiplier prefix (K/M/G). If set, the scheduler will implement a token bucket algorithm with parameters (rate, burst) to limit the sending of packets on the wire. If unset, the scheduler will try to submit as many packets as possible up to the burst size, regardless of the link rate.

burst <burst> (optional) maximum bytes sent in a scheduler round, with an optional multiplier prefix (K/M/G). Default 48K.

l2overhead <l2overhead> (optional) layer 2 overhead in bytes. Bytes added to the frame size to enforce `rate` and `burst`. Default 24 (ethernet CRC (Cyclic Redundancy Code) + IFG (InterFrame Gap) + preamble).

qsize <pkts>[,<pkts>[,...]] (optional) size of each class queue, in packets. Default 256.

qrate <rate>[,<rate>[,...]] (optional) configures traffic shaping on a list of classes, that is, the maximum rate in BPS at which each of them can be dequeued by the scheduler.

A zero <`rate`> disables traffic shaping. This is the default.

See `rate` parameter for more information.

qburst <burst>[,<burst>[,...]] (optional) maximum number of bytes dequeued from each class during a scheduler round. Only relevant to classes whose `qrate` is nonzero.

If zero or unspecified, a default value is inherited from `burst`.

priority <priority>[,<priority>[,...]] (optional) force priority level of each class instead of relying on the default behavior of the chosen scheduler.

weight <weight>[,<weight>[,...]] (optional) weight of each class relative to others for a given priority level. Some scheduling algorithms such as DWRR rely on that to balance bandwidth in case of congestion.

drop <x/y>[,<x/y>[,...]] (optional) 'x' packets will be forcefully dropped per 'y' packets for each class in the fuzz scheduler.

delay <delay>[,<us>[,...]] (optional) delay of each class introduces latencies before sending the packets in the fuzz scheduler.

lsize <nb_pkts>[,<nb_pkts>[,...]] (optional) lsize helps to configure the fuzz scheduler to dynamically set list sizes per class that will be used to cache packets in the scenario when the class queue is full due to delay introduced by the user.

cpu <coreid> (optional) core that will schedule packets for this interface. Default: scheduler core with the least schedulers attached, or, in the absence of scheduler core, the worker core with the least schedulers attached.

Example

Configure a strict priority scheduler on interface `ntfp2`, with 4 classes, and a rate of 1 GBPS (gigabits/sec = 1000^3 bits/second).

```
qos-sched-add ntfp2 prio 4 rate 1g
```

Display scheduling configuration

Display all or a subset of QoS schedulers.

```
qos-sched [iface <ifname>|cpu <cpu>|index <index>|per-iface|raw]
```

By default, display all schedulers by scanning fast path cores.

Parameters

iface <ifname> display the scheduler attached to the specified interface.

cpu <cpu> display schedulers handled by the specified fast path core.

index <index> display the scheduler with this index.

per-iface display all schedulers by scanning interfaces.

raw display all schedulers by scanning the scheduler table.

Example

```
<fp-0> qos-sched
=== core 2:
[1] ntfp2-vrf0 core=2 algo=prio classes=4 def-classid=0x4 rate=1G burst=48K␣
↪l2overhead=24 status=active
```

Disable scheduling on a physical interface

Disable QoS on a physical interface and release all attached resources (classes, filters, meters…).

```
qos-sched-del iface <ifname>
```

or

```
qos-sched-del index <index>
```

Parameters

<ifname> physical interface on which the scheduler is configured.

<index> index in the table of scheduler object.

Example

Disable QoS on interface `ntfp2`:

```
qos-sched-del iface ntfp2
```

Delete scheduler 1, and disable QoS on the interface to which it was attached.

```
qos-sched-del index 1
```

Configure QoS classes

QoS classes are automatically create when adding a strict priority scheduler. They cannot be individually added or deleted.

Display classes

Display all or a subset of QoS classes.

```
qos-class [iface <ifname>|cpu <cpu>|index <index>|per-iface|raw]
```

By default, display all classes by scanning fast path cores.

Parameters

iface <ifname> display the class attached to the specified interface.

cpu <cpu> display classes handled by the specified class core.

index <index> display the class with this index.

per-iface display all classes by scanning interfaces.

raw display all classes by scanning the class table.

Example

```
<fp-0> qos-class
=== core 2:
--- ntfp2-vrf0:
[1] classid=0x1 prio=1 weight=1 qsize=256 rate=0 burst=48K status=active
[2] classid=0x2 prio=2 weight=1 qsize=256 rate=0 burst=48K status=active
[3] classid=0x3 prio=3 weight=1 qsize=256 rate=0 burst=48K status=active
[4] classid=0x4 prio=4 weight=1 qsize=256 rate=0 burst=48K status=active
```

Configure QoS filters

An ordered table of QoS filter rules may be configured for each physical network interface with scheduling enabled. This table maps netfilter marks to class IDs.

A scheduler must be configured on the interface.

The default behavior when a packet matches no filter rule is to use the low order 16 bits of the mark as the packet classid. If this value does not match an existing class, then the packet is sent to the scheduler default class (lowest priority in the case of a strict priority scheduler).

The reserved class ID 0xffff references the *direct queue*, it indicates that the traffic will bypass QoS metering, scheduling and shaping, it will be directly sent over the wire. The direct queue must be reserved to sporadic traffic that is not likely to disturb the scheduled traffic.

Add a QoS filter rule

```
qos-filter-add <ifname> <prio> <mark>[/<mask>] [flag <flag>] <mark-to-classid>
where <flag> is:
    cp        (critical control plane traffic)
    nocp      (any packet except critical control plane traffic)
where <mark-to-classid> is:
   set-classid <val>
   and-classid <val>
   or-classid <val>
   xor-classid <val>
   xset-classid <val>[/<mask>]
```

Example

<ifname> interface on which the scheduler is configured.

<prio> priority of the filtering rule.

<mark>[/<mask>] netfilter mark filter. The rule applies to all packets matching this mark filter.

<flag> optional flag. `cp` matches critical control plane traffic (ARP, ICMP, routing protocols, IKE...). `nocp` matches data traffic and non-critical control plane traffic.

If a flag option is set, the filter matches traffic that statisfies both mark and flag conditions.

The filtering action <mark-to-classid> is similar to the netfilter `MARK` target. It can be one of the following, knowing that <val> is an integer between 0 and 255. First the classid is initialized to the low order 16 bits of the mark. Then the classid is calculated as follows:

set-classid <val> the classid is set to <val>.

and-classid <val> a logical `AND` is done between the classid and <val>.

or-classid <val> a logical `OR` is done between the classid and <val>.

xor-classid <val> a logical `XOR` is done between the classid and <val>.

xset-classid <val>[/<mask>] first the bits of the classid in <mask> are reset, then a logical `XOR` is done between the classid and <val>.

Example

Send all packets with netfilter mark ending with 0x0001 to class 0x2:

```
<fp-0> qos-filter-add ntfp2 10 1/0xffff set-classid 0x2
```

Send all packets with netfilter mark first byte equal to 0x80 to the class whose classid is the mark last byte:

```
<fp-0> qos-filter-add ntfp2 10 0x80000000/0xf0000000 and-classid 0xff
```

Delete a QoS filter rule

```
qos-filter-del iface <ifname> [prio <prio>]
```

or:

```
qos-filter-del index <index>
```

Parameters

<ifname> interface on which the QoS filter rule is configured.

<prio> priority of the filter rule.

<index> index in the table of filter rule objects.

Example

Delete the first QoS filter rule on interface ntfp2:

```
<fp-0> qos-filter-del iface ntfp2
```

Delete the QoS filter entry with index 2:

```
<fp-0> qos-filter-del index 2
```

Display QoS filter rules

```
qos-filter [iface <ifname>|cpu <cpu>|index <index>|per-iface|raw] [detail]
```

Parameters

iface <ifname> display the QoS filter rules attached to the specified interface.

cpu <cpu> display the QoS filter rules handled by the specified fast path core.

index <index> display the QoS filter rule with this index.

per-iface display all QoS filter rules by scanning interfaces.

raw display all QoS filter rules by scanning the QoS filter rule table.

detail display more details

Example

```
<fp-0> qos-filter iface ntfp2
=== core 2:
--- ntfp2-vrf0:
[1]    10: mark=0x1/0xff set-classid=0x1 status=active
[2]    10: mark=0x80000000/0xf0000000 and-classid=0xff status=active
```

Configure QoS metering and policing

A meter may be attached to each QoS scheduler and to each QoS class.

A meter implements a color-blind trTCM conformant to **RFC 4115** (https://tools.ietf.org/html/rfc4115.html), in order to evaluate the conformance of traffic to a configured profile.

Packets that must be enqueued in a QoS class queue are submitted to the meter attached to the scheduler (if any), then to the meter attached to the class (if any). Depending on the color assigned by the meter (green, yellow or red), the packet may be accepted (with an optional TOS remarking) or dropped.

Configure a meter

Attach a meter to a scheduler or to a class.

```
qos-meter-add <ifname> <classid> <CIR> <CBS> <EIR> <EBS> pps|bps
              [green <action>] [yellow <action>] [red <action>]
where <action> is:
   set-tos <tos>
   and-tos <tos>
   or-tos <tos>
   xor-tos <tos>
   xset-tos <tos>[/<mask>]
   accept
   drop
```

Parameters

The metering action <action> for each color consists in dropping or accepting the packet, with an optional TOS remarking.

<ifname> interface on which the meter is defined.

<classid> classid of the class to which the meter is attached. If 0, then the meter is attached to the scheduler itself (common to all classes).

<CIR> CIR, expressed in bps or pps, with an optional multiplier (K/M/G).

<CBS> CBS, expressed in bytes or packets, with an optional multiplier (K/M/G).

<EIR> EIR, expressed in bps or pps, with an optional multiplier (K/M/G).

<EBS> EBS, expressed in bytes or packets, with an optional multiplier (K/M/G).

pps|bps Unit used for CIR, CBS, EIR and EBS.

- pps means that values are expressed in terms of packets:
  - rates are in pps (CIR and EIR)

  - burst sizes are in packets (CBS and EBS)
- bps means that values are expressed in terms of bits.
  - rates are in bps (CIR and EIR)
  - burst sizes are in bytes (CBS and EBS)

By default, the actions are:

```
green accept yellow accept red drop
```

Example

Attach a simple policer for all traffic sent to `ntfp2` scheduler: drop traffic that exceeds 500 Mbps with a burst size of 48 Kbytes:

```
<fp-0> qos-meter-add ntfp2 0 500m 48k 0 0 bps
```

Attach a 2-level policer on class 2 of interface `ntfp2`, that remarks the TOS of yellow packets and drops red packets:

```
<fp-0> qos-meter-add ntfp2 2 200m 48k 20m 15k bps yellow set-tos 0x48
```

Delete a meter

Delete a meter from a scheduler or to a class.

```
qos-meter-del iface <ifname> <classid>
```

or

```
qos-meter-del index <index>
```

Parameters

<ifname> interface on which the meter is defined.

<classid> classid of the class to which the meter is attached. If 0, then the meter is attached to the scheduler itself (common to all classes).

<index> index in the table of meter objects.

Example

```
```

Delete the meter attached to interface ntfp2:

```
<fp-0> qos-meter-del iface ntfp2 0
```

Delete the meter entry with index 2:

```
<fp-0> qos-meter-del index 2
```

Display meters

Display all or a subset of QoS meters.

```
qos-meter [iface <ifname>|cpu <cpu>|index <index>|per-iface|raw] [detail]
```

Parameters

By default, display all meters by scanning fast path cores.

iface <ifname> display the meters attached to the specified interface.

cpu <cpu> display the meters handled by the specified fast path core.

index <index> display the meter with this index.

per-iface display all meters by scanning interfaces.

raw display all meters by scanning the meter table.

detail display more details

Example

```
<fp-0> qos-meter
=== core 2:
--- ntfp2-vrf0:
[1] class=0 cir='500M bps' cbs='48K B' eir='0 bps' ebs='0 B' green='accept' yellow=
↪'accept' red='drop' status=active
[2] class=2 cir='200M bps' cbs='48K B' eir='20M bps' ebs='15K B' green='accept' yellow=
↪'set-tos 0x48' red='drop' status=active
```

QoS statistics

Display QoS statistics

Display all QoS statistics in a human readable form, except QoS filter statistics.

```
qos-stats [percore] [all]
```

Parameters

percore display statistics per-core, when applicable. Note that metering and transmit statistics are not maintained per core.

all Display all statistics (even those that are null).

Example

```
<fp-0> qos-stats all
sched iface=ntfp2-vrf0 [1]:
| enq_ok_pkts:199214
| enq_drop_noclass_pkts:0
| enq_drop_meter_pkts:0
| enq_drop_qfull_pkts:0
| xmit_ok_pkts:199214
| xmit_drop_pkts:0
| meter [1]:
| | green packets:199214 bytes:19522076
| | yellow packets:0 bytes:0
| | red packets:0 bytes:0
| class classid=0x1 [1]:
| | enq_ok_pkts:11
| | enq_drop_meter_pkts:0
| | enq_drop_qfull_pkts:0
| | xmit_ok_pkts:11
| | xmit_drop_pkts:0
| class classid=0x2 [2]:
| | enq_ok_pkts:141008
| | enq_drop_meter_pkts:0
| | enq_drop_qfull_pkts:0
| | xmit_ok_pkts:141008
| | xmit_drop_pkts:0
| | meter [2]:
| | | green packets:141008 bytes:13818784
```

(continues on next page)

(continued from previous page)

```
| | | yellow packets:0 bytes:0
| | | red packets:0 bytes:0
| class classid=0x3 [3]:
| | enq_ok_pkts:0
| | enq_drop_meter_pkts:0
| | enq_drop_qfull_pkts:0
| | xmit_ok_pkts:0
| | xmit_drop_pkts:0
| class classid=0x4 [4]:
| | enq_ok_pkts:58195
| | enq_drop_meter_pkts:0
| | enq_drop_qfull_pkts:0
| | xmit_ok_pkts:58195
| | xmit_drop_pkts:0
```

This command navigates through all schedulers and for each scheduler dumps:

- the scheduler statistics (packet enqueuing and transmission counters),
- the optional scheduler meter statistics (green/yellow/red packet counters)
- the class statistics (packet enqueuing and transmission counters)
- the optional class meter statistics (green/yellow/red packet counters)

Display QoS statistics in json format

Display all QoS statistics in JSON (JavaScript Object Notation) format, except QoS filter statistics.

```
qos-stats-json
```

Example

```
<fp-0> qos-stats-json
[
  {
    "ifname": "ntfp2",
    "vrfid": 0,
    "index": 1,
    "enq_ok_pkts": 199214,
    "enq_drop_noclass_pkts": 0,
    "enq_drop_meter_pkts": 0,
    "enq_drop_qfull_pkts": 0,
    "xmit_ok_pkts": 199214,
```

(continues on next page)

(continued from previous page)


```
    "xmit_drop_pkts": 0,
    "meter": {
      "green": {
        "packets": 199214,
        "bytes": 19522076
      },
      "yellow": {
        "packets": 0,
        "bytes": 0
      },
      "red": {
        "packets": 0,
        "bytes": 0
      }
    },
    "classes": [
      {
        "classid": 1,
        "index": 1,
        "enq_ok_pkts": 11,
        "enq_drop_meter_pkts": 0,
        "enq_drop_qfull_pkts": 0,
        "xmit_ok_pkts": 11,
        "xmit_drop_pkts": 0
      },
      {
        "classid": 2,
        "index": 2,
        "enq_ok_pkts": 141008,
        "enq_drop_meter_pkts": 0,
        "enq_drop_qfull_pkts": 0,
        "xmit_ok_pkts": 141008,
        "xmit_drop_pkts": 0,
        "meter": {
          "green": {
            "packets": 141008,
            "bytes": 13818784
          },
          "yellow": {
            "packets": 0,
            "bytes": 0
          },
          "red": {
            "packets": 0,
```


(continues on next page)

(continued from previous page)


```
            "bytes": 0
          }
        }
      },
      {
        "classid": 3,
        "index": 3,
        "enq_ok_pkts": 0,
        "enq_drop_meter_pkts": 0,
        "enq_drop_qfull_pkts": 0,
        "xmit_ok_pkts": 0,
        "xmit_drop_pkts": 0
      },
      {
        "classid": 4,
        "index": 4,
        "enq_ok_pkts": 58195,
        "enq_drop_meter_pkts": 0,
        "enq_drop_qfull_pkts": 0,
        "xmit_ok_pkts": 58195,
        "xmit_drop_pkts": 0
      }
    ]
  }
]
```


This command displays the same information as `qos-stats` in the JSON format. This format enables later parsing by various applications.

Display QoS filter rules statistics

Display QoS filter statistics in a human readable form.

```
qos-filter-stats [iface <ifname>|cpu <cpu>|index <index>|per-iface|raw]
                 [detail] [percore] [all]
```

Parameters

iface <ifname> display the QoS filter rules attached to the specified interface.

cpu <cpu> display the QoS filter rules handled by the specified fast path core.

index <index> display the QoS filter rule with this index.

per-iface display all QoS filter rules by scanning interfaces.

raw display all QoS filter rules by scanning the QoS filter rule table.

detail display more details

percore display statistics per-core.

all display all statistics (even those that are null).

Example

```
<fp-0> qos-filter iface ntfp2
=== core 2:
--- ntfp2-vrf0:
[1]    10: mark=0x1/0xff set-classid=0x1 status=active
  match_pkts:10
[2]    10: mark=0x80000000/0xf0000000 and-classid=0xff status=active
  match_pkts:8
```

```
<fp-0> qos-filter-stats index 2 percore all
[2] sched=1 prio=10 mark=0x80000000/0xf0000000 and-classid=0xff status=active
  match_pkts:
    match_pkts[1]:5
    match_pkts[2]:2
    match_pkts[3]:1
    Total:0
```

Display QoS and filters configuration and statistics in json format

Display all QoS configuration and statistics with filter statistics in JSON format.

```
qos-dump-json
```


Example

```
<fp-0> qos-dump-json
{
  "ntfp2-vr0": {
    "core": 1,
    "nb-classes": 3,
    "def-classid": 3,
    "rate": 8000000000,
    "burst": 12000,
    "l2overhead": 24,
    "algo": "dwrr",
    "status": "active",
    "enq_ok_pkts": 0,
    "enq_drop_noclass_pkts": 0,
    "enq_drop_meter_pkts": 0,
    "enq_drop_qfull_pkts": 0,
    "xmit_ok_pkts": 0,
    "xmit_drop_pkts": 0,
    "classes": {
      "1": {
        "prio": 1,
        "weight": 1500,
        "qsize": 512,
        "rate": 0,
        "burst": 0,
        "enq_ok_pkts": 0,
        "enq_drop_noclass_pkts": 0,
        "enq_drop_meter_pkts": 0,
        "enq_drop_qfull_pkts": 0,
        "xmit_ok_pkts": 0,
        "xmit_drop_pkts": 0,
        "set-filters": {
          "0": {
            "mark": "0x00000004",
            "mask": "0xffffffff",
            "classid_val": "0x0001",
            "classid_mask": "0xffff",
            "match_pkts": 0
          },
          "1": {
            "mark": "0x00000002",
            "mask": "0xffffffff",
            "classid_val": "0x0001",
            "classid_mask": "0xffff",
```

(continues on next page)

(continued from previous page)


```
            "match_pkts": 0
          }
        }
      },
      "2": {
        "prio": 2,
        "weight": 1500,
        "qsize": 256,
        "rate": 4000000000,
        "burst": 48000,
        "enq_ok_pkts": 0,
        "enq_drop_noclass_pkts": 0,
        "enq_drop_meter_pkts": 0,
        "enq_drop_qfull_pkts": 0,
        "xmit_ok_pkts": 0,
        "xmit_drop_pkts": 0,
        "set-filters": {
          "2": {
            "mark": "0x00000010",
            "mask": "0xffffffff",
            "classid_val": "0x0002",
            "classid_mask": "0xffff",
            "match_pkts": 0
          }
        }
      },
      "3": {
        "prio": 2,
        "weight": 2000,
        "qsize": 256,
        "rate": 0,
        "burst": 0,
        "enq_ok_pkts": 0,
        "enq_drop_noclass_pkts": 0,
        "enq_drop_meter_pkts": 0,
        "enq_drop_qfull_pkts": 0,
        "xmit_ok_pkts": 0,
        "xmit_drop_pkts": 0,
        "meter": {
          "cir": 6000000000,
          "cbs": 1500,
          "eir": 500000000,
          "ebs": 1500,
          "unit": "bps",
```


(continues on next page)

(continued from previous page)

```
          "green-packets": 0,
          "green-bytes": 0,
          "yellow-packets": 0,
          "yellow-bytes": 0,
          "red-packets": 0,
          "red-bytes": 0
        },
        "set-filters": {
          "-1": {
            "mark": "0x00000000",
            "mask": "0x00000000",
            "classid_val": "0x0003",
            "classid_mask": "0xffff",
            "match_pkts": 0
          }
        }
      }
    },
    "other-filters": {
      "5": {
        "mark": "0x80000000",
        "mask": "0xf0000000",
        "classid_val": "0x0000",
        "classid_mask": "0xff00",
        "match_pkts": 0
      }
    }
  }
}
```

Reset QoS statistics

Reset all QoS statistics.

```
qos-stats-reset
```

Example of simple shaping with control plane protection

Here is a simple example where the output traffic on interface `eth2` is shaped at 100Mbps, while protecting the control plane critical traffic.

A Priority Queueing scheduler is configured with 2 classes, one for critical control plane traffic, one for other traffic. All traffic with the `cp` flag is directed to class 1, with the highest priority:

```
qos-sched-add eth2 prio 2 rate 100M
qos-filter-add eth2 10 0/0 flag cp set-classid 1
```

By default, all other traffic is sent to class 2, the class with the lowest priority. An explicit filter may however be added if marks are used by other fast path modules, in order to send all other traffic to class 2 regardless of its mark:

```
qos-filter-add eth2 10 0/0 set-classid 2
```

2.3.21 Fast Path Reassembly IPv4

Overview

Fast Path Reassembly IPv4 provides IPv4 packet reassembly in the fast path.

Features

- IPv4 reassembly

Dependencies

6WINDGate modules

- *Fast Path Forwarding IPv4*

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

Setting the force reassembly flag for IPv4

Synopsis

```
reass4-force-set <ifname> on|off
```

<ifname> Name of the interface in human reading form, must be unique.

on or off Enable or disable reassembly.

Example

```
<fp-0> reass4-force-set eth0 on
```

Displaying the maximum queue length

Synopsis

```
reass4-maxqlen
```

Example

```
<fp-0> reass4-maxqlen
IPv4 reass. max queue length: 10
```

Setting the maximum queue length

Synopsis

```
reass4-maxqlen-set <val>
```

<val> Maximum queue length for IPv4 reassembly.

Example

```
<fp-0> reass4-maxqlen
IPv4 reass. max queue length: 10
```

```
<fp-0> reass4-maxqlen-set 13
IPv4 reass. max queue length: 13
```

2.3.22 Fast Path Reassembly IPv6

Overview

Fast Path Reassembly IPv6 provides IPv6 fragmented packets reassembly in the fast path.

Features

- IPv6 reassembly

Dependencies

Modules

- *Fast Path Forwarding IPv6*

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

Setting the force reassembly flag for IPv6

Synopsis

```
reass6-force-set <ifname> on|off
```

<ifname> Name of the interface in human reading form, must be unique.

on or off Enable or disable reassembly.

Example

```
<fp-0> reass6-force-set eth0 on
```

Displaying the maximum queue length

Synopsis

```
reass6-maxqlen
```

Example

```
<fp-0> reass6-maxqlen
IPv6 reass. max queue length: 10
```

Setting the maximum queue length

Synopsis

```
reass6-maxqlen-set <val>
```

<val> Maximum queue length for IPv6 reassembly.

Example

```
<fp-0> reass6-maxqlen
IPv6 reass. max queue length: 10
```

```
<fp-0> reass6-maxqlen-set 16
IPv6 reass. max queue length: 16
```

2.3.23 Fast Path Tunnelling (IPinIP)

Overview

Fast Path Tunnelling (IPinIP) provides IPinIP tunnelling in the fast path.

Features

- IPv4 in IPv4 tunnelling
- IPv6 in IPv4 tunnelling
- IPv4 in IPv6 tunnelling
- IPv6 in IPv6 tunnelling

Dependencies

6WINDGate modules

- *Fast Path Forwarding IPv4*
- *Fast Path Forwarding IPv6*

Linux

- IPv4 over IPv4 and IPv6 over IPv4 on the same interface is a kernel patch (upstream 3.11).
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=32b8a8e59c9c
- Full Linux synchronization is a kernel patch (upstream 3.8).
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=0974658da47c
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=ba3e3f50a0e5
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=c075b13098b3
- Cross-VRF support for IPv4/IPv6 over IPv4 is a kernel patch (upstream 3.11).
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=5e6700b3bf98
- Cross-VRF support for IPv4/IPv6 over IPv6 is a kernel patch (upstream 3.12).
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=0bd8762824e7

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

```
# modprobe ipip
# modprobe sit
# modprobe ip6_tunnel
```

Example

```
# echo 1 > /proc/sys/net/ipv4/conf/all/forwarding
# ip a a 10.22.4.104/24 dev eth1
# ip a a 10.23.4.104/24 dev eth3
# ip link set up dev eth1
# ip link set up dev eth3

# ip tu ad ipiptun mode ipip local 10.23.4.104 remote 10.23.4.204 ttl 64 dev eth3
# ip ad ad dev ipiptun 192.168.1.1 peer 192.168.1.2/32
# ip li se dev ipiptun up
# ip ro ad 10.24.4.119/32 via 192.168.1.1

# fp-cli
```

```
<fp-0> iface
1:lo [VR-0] ifid=1 (virtual) <UP|RUNNING|FWD4|FWD6> (0x63)
     type=loop mac=00:00:00:00:00:00 mtu=16436 tcp4mss=0 tcp6mss=0
     IPv4 routes=0  IPv6 routes=0
     if_ops: rx_dev=none tx_dev=none ip_output=none
6:eth2 [VR-0] ifid=6 (virtual) <UP|RUNNING|FWD4|FWD6> (0x63)
     type=ether mac=00:1b:21:c5:7f:75 mtu=1500 tcp4mss=0 tcp6mss=0
     IPv4 routes=1  IPv6 routes=0
     if_ops: rx_dev=none tx_dev=none ip_output=none
7:eth3 [VR-0] ifid=7 (port 1) <UP|RUNNING|FWD4|FWD6> (0x63)
     type=ether mac=00:1b:21:c5:7f:76 mtu=1500 tcp4mss=0 tcp6mss=0
     IPv4 routes=0  IPv6 routes=0
     if_ops: rx_dev=none tx_dev=none ip_output=none
8:eth1 [VR-0] ifid=8 (port 0) <UP|RUNNING|FWD4|FWD6> (0x63)
     type=ether mac=00:1b:21:c5:7f:74 mtu=1500 tcp4mss=0 tcp6mss=0
     IPv4 routes=0  IPv6 routes=0
     if_ops: rx_dev=none tx_dev=none ip_output=none
9:eth0 [VR-0] ifid=9 (virtual) <FWD4|FWD6> (0x60)
     type=ether mac=00:21:85:c1:82:58 mtu=1500 tcp4mss=0 tcp6mss=0
```

(continues on next page)

(continued from previous page)

```
     IPv4 routes=0  IPv6 routes=0
     if_ops: rx_dev=none tx_dev=none ip_output=none
10:eth4 [VR-0] ifid=10 (port 2) <FWD4|FWD6> (0x60)
     type=ether mac=00:1b:21:c5:7f:77 mtu=1500 tcp4mss=0 tcp6mss=0
     IPv4 routes=0  IPv6 routes=0
     if_ops: rx_dev=none tx_dev=none ip_output=none
15:ipiptun [VR-0] ifid=15 (virtual) <UP|RUNNING|FWD4|FWD6> (0x63)
     type=Xin4 (6to4) mac=00:00:00:00:00:00 mtu=1480 tcp4mss=0 tcp6mss=0
     IPv4 routes=0  IPv6 routes=0
     ttl=0 local=10.23.4.104 remote=10.23.4.204 link-vrfid=0
     if_ops: rx_dev=none tx_dev=none ip_output=none
<fp-0> route4 type all
# - Preferred, * - Active, > - selected
0.0.0.0/32  [5001]  LOCAL (1)
10.22.4.104/32  [02]  ADDRESS via eth1-vr0 (6)
10.22.4.118/32  [06]  NEIGH gw 10.22.4.118 (N) via eth1-vr0 (10)
10.23.4.104/32  [03]  ADDRESS via eth3-vr0 (7)
10.23.4.204/32  [05]  NEIGH gw 10.23.4.204 (N) via eth3-vr0 (9)
10.24.4.119/32  [09]  ROUTE gw 192.168.1.1 via ipiptun-vr0 (12)
10.81.0.100/32  [08]  NEIGH gw 10.81.0.100 (N) via eth2-vr0 (11)
10.81.0.141/32  [01]  ADDRESS via eth2-vr0 (5)
127.0.0.0/8  [5002]  BLACKHOLE (4)
192.168.1.1/32  [04]  ADDRESS via ipiptun-vr0 (8)
224.0.0.0/4  [5001]  LOCAL (3)
255.255.255.255/32  [5001]  LOCAL (2)
```

Example IPv4 and IPv6 in IPv6

```
# echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
# ip link set eth1 up
# ip addr add fd00:100::2/64 dev eth1
# ip link set eth2 up
# ip addr add fd00:125::1/64 dev eth2
# ip link set eth3 up
# ip addr add fd00:175::1/64 dev eth3
# ip link add tun1 type ip6tnl ttl 64 local fd00:125::1 remote fd00:125::2 dev eth2
# ip addr add fd00:cafe:cafe::1/128 peer fd00:cafe:cafe::2/128 dev tun1
# ip link set dev tun1 up
# ip route add fd00:200::/64 dev tun1
# ip addr add dev tun1 192.168.1.1 remote 192.168.1.2
```

(continues on next page)

(continued from previous page)

```
# fp-cli
```

```
<fp-0> iface
1:lo [VR-0] ifid=1 (virtual) <UP|RUNNING|FWD4|FWD6> (0x1b)
     type=loop mac=00:00:00:00:00:00 mtu=0 tcp4mss=0 tcp6mss=0
     IPv4 routes=0  IPv6 routes=0
     if_ops: rx_dev=none rx_early=none tx_dev=none ip_output=none
2:mgmt0 [VR-0] ifid=2 (virtual) <UP|RUNNING|FWD4|FWD6> (0x1b)
     type=ether mac=de:ad:de:01:02:03 mtu=1500 tcp4mss=0 tcp6mss=0
     IPv4 routes=4  IPv6 routes=0
     if_ops: rx_dev=none rx_early=none tx_dev=none ip_output=none
7:eth1 [VR-0] ifid=7 (port 0) <UP|RUNNING|FWD4|FWD6> (0x1b)
     type=ether mac=de:ed:01:34:ee:0e mtu=1500 tcp4mss=0 tcp6mss=0
     IPv4 routes=0  IPv6 routes=1
     if_ops: rx_dev=none rx_early=none tx_dev=none ip_output=none
8:eth2 [VR-0] ifid=8 (port 1) <UP|RUNNING|FWD4|FWD6> (0x1b)
     type=ether mac=de:ed:02:ed:fb:f6 mtu=1500 tcp4mss=0 tcp6mss=0
     IPv4 routes=0  IPv6 routes=1
     if_ops: rx_dev=none rx_early=none tx_dev=none ip_output=none
9:eth3 [VR-0] ifid=9 (port 2) <UP|RUNNING|FWD4|FWD6> (0x1b)
     type=ether mac=de:ed:03:0b:2a:c6 mtu=1500 tcp4mss=0 tcp6mss=0
     IPv4 routes=0  IPv6 routes=1
     if_ops: rx_dev=none rx_early=none tx_dev=none ip_output=none
12:tun1 [VR-0] ifid=12 (virtual) <UP|RUNNING|FWD4|FWD6> (0x1b)
     type=Xin6 mac=00:00:00:00:00:00 mtu=1452 tcp4mss=0 tcp6mss=0
     IPv4 routes=1  IPv6 routes=1
     if_ops: rx_dev=none rx_early=none tx_dev=none ip_output=none
     hlim=64 local=fd00:125::1 remote=fd00:125::2 link-vrfid=0

 <fp-0> route6 type all
 # - Preferred, * - Active, > - selected
 (255) ::/80 [5002]  BLACKHOLE (3)
 (255) fd00:100::2/128 [14]  ADDRESS via eth1-vr0 (14)
 (255) fd00:125::1/128 [16]  ADDRESS via eth2-vr0 (16)
 (255) fd00:175::1/128 [18]  ADDRESS via eth3-vr0 (18)
 (255) fd00:cafe:cafe::1/128 [19]  ADDRESS via tun1-vr0 (19)
 (255) fe80::/10 [5001]  LOCAL (1)
 (255) ff00::/8 [5001]  LOCAL (2)
 (254) fd00:100::/64 metric 256 [13]  CONNECTED via eth1-vr0 (13)
 (254) fd00:125::/64 metric 256 [15]  CONNECTED via eth2-vr0 (15)
 (254) fd00:175::/64 metric 256 [17]  CONNECTED via eth3-vr0 (17)
 (254) fd00:200::/64 metric 1024 [20]  CONNECTED via tun1-vr0 (20)
```

(continues on next page)

(continued from previous page)

```
<fp-0> route4 type all
# - Preferred, * - Active, > - selected
(255) 0.0.0.0/32 [5001]  LOCAL (1)
(255) 10.0.2.15/32 [01]  ADDRESS via mgmt0-vr0 (5)
(255) 127.0.0.0/8 [5002]  BLACKHOLE (4)
(255) 192.168.1.1/32 [05]  ADDRESS via tun1-vr0 (10)
(255) 224.0.0.0/4 [5001]  LOCAL (3)
(255) 255.255.255.255/32 [5001]  LOCAL (2)
(254) 0.0.0.0/0 [03]  NEIGH gw 10.0.2.2 via mgmt0-vr0 (8)
(254) 10.0.2.0/24 [04]  CONNECTED via mgmt0-vr0 (9)
(254) 192.168.1.2/32 [06]  CONNECTED via tun1-vr0 (11)
(0) 10.0.2.2/32 [03]  NEIGH gw 10.0.2.2 (N) via mgmt0-vr0 (7)
(0) 10.0.2.3/32 [02]  NEIGH gw 10.0.2.3 (N) via mgmt0-vr0 (6)
```

2.3.24 Fast Path VLAN

Overview

Fast Path VLAN provides VLAN devices in the fast path.

Features

- IEEE 802.1Q VLAN support
- Automatic strip of vlan 0
- IEEE 802.1ad QinQ support
- Ingress QoS mapping
- Egress QoS mapping DSCP to VLAN priority

Dependencies

6WINDGate modules

- *Fast Path Baseline*

Linux

- Netlink notifications to put the lower interface in `promiscuous` and `allmulticast` modes is a kernel patch (upstream 3.13).

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=991fb3f74c14

- Support for 802.1ad VLAN devices (QinQ) is a kernel patch (upstream 3.10).

  http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commitdiff;h=8ad227ff89a7e

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

You can manage VLANs either under Linux or in the fast path.

```
# modprobe vlan
```

Linux

You can manage VLAN devices via the Linux commands below.

- If you do not receive netlink notifications from the lower interface, set the lower interface in `promiscuous` and `allmulticast` modes:

  ```
  $ sudo ip link set eth2 promisc on
  $ sudo ip link set eth2 allmulticast on
  ```

  Interface eth2 should now have the flags ALL_MULTI and PROMISC:

  ```
  $ fpcmd iface
  ...
  6:eth2 [VR-0] ifid=6 (port 1) <UP|RUNNING|PROMISC|FWD4|FWD6> (0x73)
          type=ether mac=00:02:02:00:00:21 mtu=1500 tcp4mss=0 tcp6mss=0
          IPv4 routes=0  IPv6 routes=0
          if_ops: rx_dev=none tx_dev=none ip_output=none
  ...
  ```

- Create a VLAN device:

  ```
  $ sudo ip link add link eth2 name eth2.1000 type vlan id 1000
  ```

  This adds the VLAN (802.1Q) device number `1000` to the link interface `eth2`. The new VLAN interface name is `eth2.1000`. A VLAN tag is appended to frames traversing this VLAN device.

- Display information about the VLAN device:

```
$ ip -d link show eth2.1000
```

- Delete the VLAN device:

```
$ sudo ip link delete eth2.1000
```

Egress QoS mapping

This feature allows setting the VLAN priority field of outgoing packets based on the priority of input packets.

In Linux, it depends on both the TOS to skb priority mapping table and the skb priority to VLAN priority mapping table. The former is hardcoded and the later can be configured via iproute2. Besides that, skb priority can be forced by an iptables rule with CLASSIFY target(–set-class).

The TOS octet in the IP header has an interesting history having been redefined several times. The TOS to skb priority table is based on **RFC 1349** (https://tools.ietf.org/html/rfc1349.html):

```
3.  Specification of the Type of Service Octet

               0     1     2     3     4     5     6     7
            +-----+-----+-----+-----+-----+-----+-----+-----+
            |                 |                       |     |
            |   PRECEDENCE    |          TOS          | MBZ |
            |                 |                       |     |
            +-----+-----+-----+-----+-----+-----+-----+-----+

4.  Specification of the TOS Field

                   1000   --   minimize delay
                   0100   --   maximize throughput
                   0010   --   maximize reliability
                   0001   --   minimize monetary cost
                   0000   --   normal service
```

In fact, Linux uses only the bit 3 and 4. Here is the mapping:

bit #3	bit# 4	skb priority
0	0	0
0	1	2
1	0	6
1	1	4

Here is an example on how to configure the skb priority to VLAN priority mapping table.

```
$ sudo ip link add link eth2 name eth2.1000 type vlan id 1000 egress-qos-map 2:6 4:5
```

The resulting mapping for the previous example is described in the following table:

bit #3	bit# 4	skb priority	vlan priority
0	0	0	0
0	1	2	6
1	0	6	0
1	1	4	5

As a result, incoming packets with tos 0x18 will get the vlan priority field set to 5.

Note:

```
$ sudo ip link add link eth2 name eth2.1000 type vlan help
Usage: ... vlan [ protocol VLANPROTO ] id VLANID                [ FLAG-LIST ]
               [ ingress-qos-map QOS-MAP ] [ egress-qos-map QOS-MAP ]
VLANPROTO: [ 802.1Q / 802.1ad ]
VLANID := 0-4095
FLAG-LIST := [ FLAG-LIST ] FLAG
FLAG := [ reorder_hdr { on | off } ] [ gvrp { on | off } ] [ mvrp { on | off } ]
        [ loose_binding { on | off } ]
QOS-MAP := [ QOS-MAP ] QOS-MAPPING
QOS-MAPPING := FROM:TO
```

Linux packet priority can also be set by iptables rules. For example:

```
$ iptables -t mangle -A POSTROUTING -p udp -o eth2.1000 -j CLASSIFY --set-class 0:2
```

By the rule, the skb priority of an udp packet to be sent on eth2.1000 will be set to 2. For the previous egress qos mapping example, these udp packets will get the vlan priority field set to 6.

Ingress QoS mapping

This feature allows mapping VLAN header prio field to the Linux internal packet priority(skb->priority) on incoming VLAN packets.

The ingress-qos-map parameters can be configured by the above `ip link add` command.

Fast path

You can manage VLAN devices via the `fp-cli` commands below.

Displaying VLAN devices

Synopsis

```
vlan
```

Example

```
<fp-0> vlan
eth2-vr0:
        eth2.1000-vr0: vlanid: 1000 vlanproto: 0x8100
                Ingress QOS mappings:
                        0:2 1:6 2:4 3:0 4:0 5:0 6:0 7:0
                Egress QOS mappings:
                        2:6 4:5
```

Note: The `fp-cli vlan` command displays the egress mapping between mbuf's priority value and the three priority bit values set in the VLAN tag.

Displaying VLAN information

Synopsis

```
iface
```

Example

```
<fp-0> iface
...
6:eth2.1000 [VR-0] ifid=6 (virtual) <FWD4|FWD6> (0x60)
        type=vlan mac=00:02:02:00:00:21 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
        if_ops: rx_dev=none tx_dev=vlan ip_output=none
```

(continues on next page)

(continued from previous page)

```
        vlan id 1000 vlan proto 0x8100 link eth2 <REORDER_HDR>
...
```

Providing options

Some capabilities can be tuned for this module.

`--ifaces`
: Maximum number of VLAN interfaces

  Default value 127

  Memory footprint per VLAN interfaces 150 B

  Range 0 .. 50000

Example

```
FP_OPTIONS="--mod-opt=vlan:--ifaces=512"
```

Then fast path can manage up to 512 VLAN interfaces.

`--hash-order`
: Size order of VLAN interfaces hash table. Value automatically updated if *`--ifaces`* is changed.

  Default value 8

  Range 1 .. 31

Example

```
FP_OPTIONS="--mod-opt=vlan:--hash-order=10"
```

Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

---

Tip: To get optimal performance, apply the following ratios to the two parameters:

Parameter	Value
–hash-order	`N`
–ifaces	`2 ** N`

2.3.25 Fast Path VXLAN

Overview

Fast Path VXLAN provides VXLAN functionalities in the fast path.

VXLAN is defined by **RFC 7348** (https://tools.ietf.org/html/rfc7348.html).

Features

- IPv4 and IPv6 transport support
- Multiple VTEP (VXLAN Tunnel End Point) support
- Multiple UDP ports support
- Static management of VXLAN FDB
- VXLAN GBP support (defined by https://tools.ietf.org/html/draft-smith-vxlan-group-policy-01)

Dependencies

6WINDGate modules

- *Fast Path Baseline*
- *Fast Path Forwarding IPv4*

Linux

- VXLAN is a kernel patch (upstream 3.7).
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=d342894c5d2f8
- VXLAN FDB multicast entries synchronization is a kernel patch (upstream 3.15).
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=9e4b93f905bc
- IPv6 UDP Zero Checksums (for IPv6 VXLAN) is a kernel patch (upstream 3.16).
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=4068579e1e09
- Cross-VRF support for VXLAN is a kernel patch (upstream 3.16).
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=f01ec1c017de
- VXLAN GBP support is a kernel patch (upstream 4.0).
  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=3511494ce2f3

Usage

In this section, it is assumed that Virtual Accelerator has been properly installed and configured. See *Getting Started* for more details.

With *Linux - Fast Path Synchronization* you can manage VXLANs (Virtual eXtensible Local Area Networks) via standard Linux commands.

Synchronizing Linux and the fast path

1. Load required modules:

```
# modprobe vxlan
```

Managing VXLANs from Linux

The Linux commands below allow you to manage VXLAN interfaces.

Creating a VXLAN interface

```
# ip link add IFNAME type vxlan id ID group GROUPADDR dev LINK_IFNAME ageing LIFETIME␣
↪dstport DSTPORT
```

IFNAME VXLAN interface's name.

ID VXLAN interface ID.

GROUPADDR Multicast group address.

LINK_IFNAME Name of the link interface.

DSTPORT Port number of the VXLAN interface (IANA has assigned port 4789, but the Linux default value is 8472).

LIFETIME Lifetime of FDB entries learnt.

Example

Create a new device (`vxlan10`). The device uses the multicast group `239.0.0.10` over `eth1` to handle packets when no forwarding table entry matches. Lifetime of FDB entries learnt is 3600 seconds.

```
# ip link add vxlan10 type vxlan id 10 group 239.0.0.10 dev eth1 ageing 3600 dstport␣
↪4789
```


Creating a VXLAN GBP interface

```
# ip link add IFNAME type vxlan id ID group GROUPADDR dev LINK_IFNAME ageing LIFETIME␣
↪dstport DSTPORT gbp
```

Example

Create a new VXLAN GBP device and set a mark in gbp field with iptables while forwarding.

```
# ip link add vxlan10 type vxlan id 10 group 239.0.0.10 dev eth1 ageing 3600 dstport␣
↪4789 gbp
# iptables -t mangle -A PREROUTING -j MARK --set-mark 0xabcd
```

Displaying information about a VXLAN interface

```
# ip -d link show vxlan10
```

Deleting a VXLAN device

```
# ip link delete vxlan10
```

Displaying the forwarding table

```
# bridge fdb show dev vxlan10
```

Creating a forwarding table entry

```
# bridge fdb add to 00:17:42:8a:b4:05 dst 192.168.1.1 dev vxlan10 permanent
```

Deleting a forwarding table entry

```
# bridge fdb delete 00:17:42:8a:b4:05 dev vxlan10
```

Managing VXLANs from the fast path

The `fp-cli` command below allows you to manage VXLAN instances from the fast path.

Displaying VXLAN interfaces

```
vxlan
```

Example

```
<fp-0> vxlan
VXLAN: check of rx checksum is off

vrfid: 0 dstport: 4789
        vxlan10-vr0 vni: 10
vrfid: 0 dstport: 8474 master: ovs
```

Note: VXLAN interfaces displaying 'master: ovs' will be managed by the *Fast Path OVS Acceleration* module. With kernel >= 4.4, a VXLAN interface will be created to represent Open vSwitch VXLAN ports.

Displaying a VXLAN interface's FDB entries

```
vxlan-fdb <ifname>
```

ifname Name of the VXLAN interface.

Example

```
<fp-0> vxlan-fdb vxlan10
vxlan10: vni=10 dstport=4789 ttl=255 srcminport=32768 srcmaxport=61000
FDB entries:
00:17:42:8a:b4:05 192.168.1.1 vni: 10 port: 4789
00:00:00:00:00:00 239.0.0.10 vni: 10 port: 4789 eth1-vr0
```

Enabling the verification of the UDP checksum on the input path

```
vxlan-check-rx-csum-set on|off
```

on|off Enable or disable the verification of the UDP checksum on the input path.

Example

```
<fp-0> vxlan-check-rx-csum-set on
VXLAN: check of rx checksum is on (was off)
```

Statistics

vxlan-stats

Description

Display vxlan statistics.

Synopsis

```
vxlan-stats [percore] [all]
```

Parameters

percore Display all statistics per core running the fast path.

all Display all statistics (even those that are null).

Example

```
<fp-0> vxlan-stats all
 VxlanDroppedUnknownIface:0
 VxlanDroppedUnknownVNI:0
 VxlanDroppedHeaderTooShort:0
 VxlanDroppedInvalidIPv4Header:0
 VxlanDroppedInvalidIPv6Header:0
 VxlanDroppedInvalidIPv4Csum:0
 VxlanDroppedInvalidIPv6Csum:0
 VxlanDroppedOvsNoDst:0
 VxlanDroppedIPv4NoDst:0
 VxlanDroppedIPv6NoDst:0
 VxlanDroppedPrependOvsFailure:0
 VxlanDroppedPrependIPv4Failure:0
 VxlanDroppedPrependIPv6Failure:0
 VxlanDroppedInvalidIpFamily:0
 VxlanFdbForwDuplicateError:0
 VxlanDroppedAddGBPFailure:0
 VxlanExceptionIFlagNotSet:0
 VxlanExceptionOvsMtuExceeded:0
 VxlanExceptionIPv4MtuExceeded:0
 VxlanExceptionIPv6MtuExceeded:0
 VxlanExceptionNoInputFdb:0
 VxlanExceptionNoOutputFdb:0
 VxlanExceptionIPv4NoMcastSrc:0
 VxlanExceptionIPv6NoMcastSrc:0
 VxlanExceptionNoRemote:0
 VxlanExceptionOvsRoute:0
 VxlanExceptionIPv4Route:0
 VxlanExceptionIPv6Route:0
 VxlanExceptionTooManyFlags:0
```

vxlan-stats-json

Description

Display vxlan statistics in json format.

Synopsis

```
vxlan-stats-json
```

Example

```
<fp-0> vxlan-stats-json
 {
   "VxlanDroppedUnknownIface": 0,
   "VxlanDroppedInvalidIpFamily": 0,
   "VxlanDroppedUnknownVNI": 0,
   "VxlanDroppedPrependOvsFailure": 0,
   "VxlanDroppedPrependIPv4Failure": 0,
   "VxlanDroppedPrependIPv6Failure": 0,
   "VxlanDroppedOvsNoDst": 0,
   "VxlanDroppedIPv4NoDst": 0,
   "VxlanDroppedIPv6NoDst": 0,
   "VxlanDroppedHeaderTooShort": 0,
   "VxlanDroppedInvalidIPv4Header": 0,
   "VxlanDroppedInvalidIPv6Header": 0,
   "VxlanDroppedInvalidIPv4Csum": 0,
   "VxlanDroppedInvalidIPv6Csum": 0,
   "VxlanFdbForwDuplicateError": 0,
   "VxlanDroppedAddGBPFailure": 0,
   "VxlanExceptionNoInputFdb": 0,
   "VxlanExceptionNoOutputFdb": 0,
   "VxlanExceptionIPv4NoMcastSrc": 0,
   "VxlanExceptionIPv6NoMcastSrc": 0,
   "VxlanExceptionIFlagNotSet": 0,
   "VxlanExceptionTooManyFlags": 0,
   "VxlanExceptionOvsMtuExceeded": 0,
   "VxlanExceptionIPv4MtuExceeded": 0,
   "VxlanExceptionIPv6MtuExceeded": 0,
   "VxlanExceptionOvsRoute": 0,
   "VxlanExceptionIPv4Route": 0,
   "VxlanExceptionIPv6Route": 0,
   "VxlanExceptionNoRemote": 0,
 }
```

Providing options

`--max-port`
Maximum number of (VXLAN destination port, VR) pairs

Default value 15

Memory footprint per (VXLAN destination port, VR) pair 2 KB

Range 0 .. 128

`--max-if`
Maximum number of VXLAN interfaces

Default value 127

Memory footprint per VXLAN interface 66 KB

Range 0 .. 50K

`--max-fdb`
Maximum number of VXLAN forwarding database entries

Default value 5000

Memory footprint per IPv6 Netfilter conntrack 100 B

Range 0 .. 50K

`--port-hash-order`
Size order of (VXLAN destination port, VR) tuples. Value automatically updated if *`--max-port`* is changed.

Default value 4

Range 1 .. 31

`--if-hash-order`
Size order of VXLAN interfaces hash table Value automatically updated if *`--max-if`* is changed.

Default value 8

Range 1 .. 31

`--fdb-hash-order`
Size order of VXLAN forwarding database hash table Value automatically updated if *`--max-fdb`* is changed.

Default value 13

Range 1 .. 31

Note: See *Fast Path Capabilities* documentation for impact of the available memory on the default value of configurable capabilities

---

2.4 Control Plane

2.4.1 Control Plane Security - IKEv1 and IKEv2

This module requires a Virtual Accelerator IPsec Application License.

Overview

Control Plane Security - IKEv1 and IKEv2 implements the Internet Key Exchange protocol in the Linux control plane.

Control Plane Security - IKEv1 and IKEv2 is an IKE implementation based on the open source *strongSwan* distribution, version 5.9.1.

It implements protocols IKEv1 and IKEv2 through a daemon named `charon`. It allows to negotiate keying material (IPsec SAs) for the use of IPsec VPNs (Virtual Private Networks).

Features

Control Plane Security - IKEv1 and IKEv2 supports all *strongSwan* features.

Supported Algorithms

The supported encryption and authentication algorithms for IKE phase 2 are listed in the fast path IPsec module *Supported Algorithms* section.

The supported Diffie-Hellman groups and Pseudo-random Functions for IKE phase 1 and 2 and the supported encryption and authentication algorithms for IKE phase 1 are listed below:

- for IKEv1 (https://wiki.strongswan.org/projects/strongswan/wiki/IKEv1CipherSuites)
- for IKEv2 (https://wiki.strongswan.org/projects/strongswan/wiki/IKEv2CipherSuites)

Note: As stated in the Overview section, our Control Plane Security - IKEv1 and IKEv2 is based on the open source *strongSwan* distribution, version 5.9.1.

Dependencies

6WINDGate modules

- *Fast Path IPsec IPv4*
- *Fast Path IPsec IPv6*

Linux

- Control Plane Security - IKEv1 and IKEv2 relies on your Linux distribution's support of IPsec and cryptographic algorithms.

  Check that the following variables are set to yes (`y`) in the kernel configuration:

  - `CONFIG_XFRM`
  - `CONFIG_XFRM_ALGO`
  - `CONFIG_XFRM_USER`
  - `CONFIG_INET_XFRM_MODE_TUNNEL`
  - `CONFIG_CRYPTO_AES`
  - `CONFIG_CRYPTO_CBC`
  - `CONFIG_CRYPTO_HMAC`
  - `CONFIG_CRYPTO_SHA1`

Usage

1. Configuration files/directories used by strongSwan:
   - `ipsec.conf`
   - `ipsec.secrets`
   - `ipsec.d/*/*` (optional)
   - `strongswan.conf` (optional)
   - `strongswan.d/*/*` (optional)
2. strongSwan's configuration base directory:

```
# ipsec --confdir
/etc/ike
```

3. strongSwan's PID directory:

```
# ipsec --piddir
/etc/ike/ipsec.d/run
```

Depending on your configuration, IKE negotiations may be:

- initiated at startup,
- triggered by dataplane traffic requesting IPsec SAs, or
- triggered by an IKE negotiation request received from a remote IKE host.

See also:

For more information, see the reference guide and configuration examples (http://wiki.strongswan.org/projects/strongswan/wiki/UserDocumentation)

Configuration Examples

Configuration example without VRF

We will configure a VPN gateway on a network that interconnects two sites via a public network.

Network topology:

```
10.22.1.0/24 ================ 10.23.1.0/24 +--------------+ 10.24.1.0/24
-------------| VPN gateway  |==============|   remote     |-------------
   private   |     with     |    public    |     VPN      |  private
   network   | cp-ipsec-ike |   network    |   gateway    |  network
 (plaintext) =================   (IPsec)    +--------------+ (plaintext)
                  .101                          .201
```

The network interfaces belong to the default network namespace.

1. Configure IP addresses and routes:

```
# ip link set eth0 up
# ip addr add 10.22.1.101/24 dev eth0
# ip link set eth2 up
# ip addr add 10.23.1.101/24 dev eth2
# ip route add default via 10.23.1.201
```

2. Create the strongSwan PID directory:

```
# mkdir -p /etc/ike/ipsec.d/run
```

3. Edit the `/etc/ike/ipsec.conf` configuration file as follows:

```
config setup

conn %default
        keyexchange=ikev2
        keyingtries=1
        mobike=no
        ikelifetime=57600s
        rekeymargin=5760s
        keylife=28800s

conn myconnection
        auto=route
        left=10.23.1.101
        right=10.23.1.201
        leftsubnet=10.22.1.0/24
        rightsubnet=10.24.1.0/24
        type=tunnel
        ike=aes-sha1-modp1024!
        esp=aes-sha1-modp1024!
        authby=psk
```

This file defines the SPs and the negotiation parameters of the SAs.

4. Add the following line to the `/etc/ike/ipsec.secrets` file:

```
10.23.1.101 10.23.1.201 : PSK 0x12345678
```

This file defines authentication secrets, such as pre-shared keys or certificates.

5. Add the following lines to the `/etc/ike/strongswan.conf` file:

```
charon {
        install_routes = no
}
```

This file defines global configuration options for IKE daemons.

6. Start *strongSwan* daemons:

```
# ipsec start
```

7. Start IKE daemons:

```
# ipsec start
Starting strongSwan 5.4.0 IPsec [starter]...
no netkey IPsec stack detected
```

(continues on next page)

(continued from previous page)

```
no KLIPS IPsec stack detected
no known IPsec stack detected, ignoring!
```

Once IKE daemons are started, the IPsec SPs are configured in the kernel and in the fast path.

A negotiation is automatically initiated the next time the IPsec gateway must forward a dataplane packet from the private network *10.22.1.0/24* to the remote private network *10.24.1.0/24*. It also possible to initiate the negotiation manually with the command 'ipsec up myconnection'.

8. Display the IKE daemons' state:

   Before a negotiation:

```
# ipsec statusall
Status of IKE charon daemon (strongSwan 5.4.0, Linux 4.4.6, x86_64):
  uptime: 70 seconds, since Jun 09 12:31:28 2016
  malloc: sbrk 1613824, mmap 0, used 456864, free 1156960
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 0
  loaded plugins: charon aes des rc2 sha2 sha1 md5 random nonce x509
revocation constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnskey sshkey
pem openssl fips-prf gmp xcbc cmac hmac curl attr kernel-netlink resolve
socket-default stroke vici updown eap-identity eap-aka eap-aka-3gpp2
eap-simaka-pseudonym eap-simaka-reauth eap-md5 eap-radius xauth-generic
Listening IP addresses:
  10.0.2.15
  10.22.1.101
  10.23.1.101
Connections:
myconnection:  10.23.1.101...10.23.1.201  IKEv2
myconnection:   local:  [10.23.1.101] uses pre-shared key authentication
myconnection:   remote: [10.23.1.201] uses pre-shared key authentication
myconnection:   child:  10.22.1.0/24 === 10.24.1.0/24 TUNNEL
Routed Connections:
myconnection{1}:  ROUTED, TUNNEL, reqid 1
myconnection{1}:   10.22.1.0/24 === 10.24.1.0/24
Security Associations (0 up, 0 connecting):
  none
```

   After a successful negotiation:

```
# ipsec statusall
Status of IKE charon daemon (strongSwan 5.4.0, Linux 4.4.6, x86_64):
  uptime: 4 minutes, since Jun 09 12:31:27 2016
  malloc: sbrk 2154496, mmap 0, used 482624, free 1671872
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 3
  loaded plugins: charon aes des rc2 sha2 sha1 md5 random nonce x509
```

(continues on next page)

(continued from previous page)

```
revocation constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnskey sshkey
pem openssl fips-prf gmp xcbc cmac hmac curl attr kernel-netlink resolve
socket-default stroke vici updown eap-identity eap-aka eap-aka-3gpp2
eap-simaka-pseudonym eap-simaka-reauth eap-md5 eap-radius xauth-generic
Listening IP addresses:
  10.0.2.15
  10.22.1.101
  10.23.1.101
Connections:
myconnection:  10.23.1.101...10.23.1.201  IKEv2
myconnection:   local:  [10.23.1.101] uses pre-shared key authentication
myconnection:   remote: [10.23.1.201] uses pre-shared key authentication
myconnection:   child:  10.22.1.0/24 === 10.24.1.0/24 TUNNEL
Routed Connections:
myconnection{1}:  ROUTED, TUNNEL, reqid 1
myconnection{1}:   10.22.1.0/24 === 10.24.1.0/24
Security Associations (1 up, 0 connecting):
myconnection[1]: ESTABLISHED 2 minutes ago, 10.23.1.101[10.23.1.101]...10.23.1.
↪201[10.23.1.201]
myconnection[1]: IKEv2 SPIs: 9217d43e81075119_i* a7a2a27f487e7598_r, pre-shared␣
↪key reauthentication in 11 hours
myconnection[1]: IKE proposal: AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024
myconnection{2}:  INSTALLED, TUNNEL, reqid 1, ESP SPIs: c83bbb1d_i ca86380f_o
myconnection{2}:  AES_CBC_128/HMAC_SHA1_96, 0 bytes_i, 0 bytes_o, rekeying in 5␣
↪hours
myconnection{2}:   10.22.1.0/24 === 10.24.1.0/24
```

9. Display the Linux IPsec SPs:

```
# ip xfrm policy show
src 10.24.1.0/24 dst 10.22.1.0/24
        dir fwd priority 2883 ptype main
        tmpl src 10.23.1.201 dst 10.23.1.101
                proto esp reqid 1 mode tunnel
src 10.24.1.0/24 dst 10.22.1.0/24
        dir in priority 2883 ptype main
        tmpl src 10.23.1.201 dst 10.23.1.101
                proto esp reqid 1 mode tunnel
src 10.22.1.0/24 dst 10.24.1.0/24
        dir out priority 2883 ptype main
        tmpl src 10.23.1.101 dst 10.23.1.201
                proto esp reqid 1 mode tunnel
src 0.0.0.0/0 dst 0.0.0.0/0
        socket in priority 0 ptype main
```

(continues on next page)

(continued from previous page)

```
src 0.0.0.0/0 dst 0.0.0.0/0
        socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
        socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
        socket out priority 0 ptype main
src ::/0 dst ::/0
        socket in priority 0 ptype main
src ::/0 dst ::/0
        socket out priority 0 ptype main
src ::/0 dst ::/0
        socket in priority 0 ptype main
src ::/0 dst ::/0
        socket out priority 0 ptype main
```

10. Display the negotiated Linux IPsec SAs:

```
# ip xfrm state show
src 10.23.1.101 dst 10.23.1.201
        proto esp spi 0xca86380f reqid 1 mode tunnel
        replay-window 32 flag af-unspec
        auth-trunc hmac(sha1) 0x8239c2887f263f346ff555a72355ff7146b8f240 96
        enc cbc(aes) 0x1f25429035e5d86455ce7af4ba0755c9
        anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
src 10.23.1.201 dst 10.23.1.101
        proto esp spi 0xc83bbb1d reqid 1 mode tunnel
        replay-window 32 flag af-unspec
        auth-trunc hmac(sha1) 0xe028f32ecc1a53d80940fca70d49351bb26d035e 96
        enc cbc(aes) 0x35d05466e993ff1793da4d4e80d3773b
        anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
```

11. Display the fast path IPsec SPs:

```
<fp-0> ipsec4-spd all
Inbound SPD: 1 rules
1: 10.24.1.0/24 10.22.1.0/24 proto any vr0 protect prio 2883
     link-vr0
     ESP tunnel 10.23.1.201 - 10.23.1.101 reqid=1
     sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
Outbound SPD: 1 rules
1: 10.22.1.0/24 10.24.1.0/24 proto any vr0 protect prio 2883
     link-vr0 cached-SA 0 (genid 0)
     ESP tunnel 10.23.1.101 - 10.23.1.201 reqid=1
     sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
```

12. Display the negotiated fast path IPsec SAs:

```
<fp-0> ipsec4-sad all
SAD 2 SA.
1: 10.23.1.201 - 10.23.1.101 vr0 spi 0xc83bbb1d ESP tunnel
     x-vr0 reqid=1 counter 1 (genid 1)
     AES-CBC HMAC-SHA1
     key enc:35d05466e993ff1793da4d4e80d3773b
     digest length: 12
     key auth:e028f32ecc1a53d80940fca70d49351bb26d035e
     sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
     sa_replay_errors=0 sa_selector_errors=0
     replay width=32 seq=0x0 - oseq=0x0
     00000000
2: 10.23.1.101 - 10.23.1.201 vr0 spi 0xca86380f ESP tunnel
     x-vr0 reqid=1 counter 1 (genid 2)
     AES-CBC HMAC-SHA1
     key enc:1f25429035e5d86455ce7af4ba0755c9
     digest length: 12
     key auth:8239c2887f263f346ff555a72355ff7146b8f240
     sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
     sa_replay_errors=0 sa_selector_errors=0
     replay width=32 seq=0x0 - oseq=0x0
     00000000
```

13. Stop IKE daemons:

```
# ipsec stop
```

This command stops IKE daemons and removes all SPs and SAs.

Configuration example with VRF

We will configure a VPN gateway on a network that interconnects two sites via a public network.

Network topology:

```
10.22.1.0/24 ================ 10.23.1.0/24 +--------------+ 10.24.1.0/24
-------------| VPN gateway  |==============|   remote     |-------------
   private   |     with     |    public    |     VPN      |  private
   network   | cp-ipsec-ike |   network    |   gateway    |  network
 (plaintext) ================   (IPsec)    +--------------+ (plaintext)
                  .101                          .201
```

The network interfaces are in the network namespace *vrf1*.

To launch a strongSwan instance in a specific VRF its configuration must be stored in '/etc/netns/<netns name>/ike' directory.

1. Configure IP addresses and routes:

   In this example, interfaces *eth0* and *eth2* are moved to network namespace *vrf1*.

```
# vrfctl add 1
# ip link set eth0 down netns vrf1
# ip link set eth2 down netns vrf1
# ip netns exec vrf1 ip link set eth0 up
# ip netns exec vrf1 ip addr add 10.22.1.101/24 dev eth0
# ip netns exec vrf1 ip link set eth2 up
# ip netns exec vrf1 ip addr add 10.23.1.101/24 dev eth2
# ip netns exec vrf1 ip route add 0.0.0.0/0 via 10.23.1.201
```

2. Create strongSwan working directories:

```
# mkdir -p /etc/ike/ipsec.d/run
# mkdir -p /etc/netns/vrf1/ike/ipsec.d/run
```

   See also:

   For more information, see strongSwan netns wiki (https://wiki.strongswan.org/projects/strongswan/wiki/Netns) or ip netns man page (http://man7.org/linux/man-pages/man8/ip-netns.8.html).

3. Edit the `/etc/netns/vrf1/ike/ipsec.conf` file as follows:

```
config setup

conn %default
        keyexchange=ikev2
        keyingtries=1
        mobike=no
        ikelifetime=57600s
        rekeymargin=5760s
        keylife=28800s

conn myconnection
        auto=route
        left=10.23.1.101
        right=10.23.1.201
        leftsubnet=10.22.1.0/24
        rightsubnet=10.24.1.0/24
        type=tunnel
        ike=aes-sha1-modp1024!
        esp=aes-sha1-modp1024!
        authby=psk
```

This file defines the SPs and the negotiation parameters of the SAs.

4. Add the following line to the `/etc/netns/vrf1/ike/ipsec.secrets` file.

```
10.23.1.101 10.23.1.201 : PSK 0x12345678
```

This file defines authentication secrets, such as pre-shared keys or certificates.

5. Add the following lines to the `/etc/netns/vrf1/ike/strongswan.conf` file:

```
charon {
        install_routes = no
}
```

This file defines global configuration options for IKE daemons.

6. Start IKE daemons:

```
# ip netns exec vrf1 ipsec start
Starting strongSwan 5.4.0 IPsec [starter]...
no netkey IPsec stack detected
no KLIPS IPsec stack detected
no known IPsec stack detected, ignoring!
```

Once IKE daemons are started, the IPsec SPs are configured in the kernel and in the fast path.

A negotiation is automatically initiated the next time the IPsec gateway must forward a dataplane packet from the private network *10.22.1.0/24* to the remote private network *10.24.1.0/24*.

7. Display the IKE daemons' state:

Before a negotiation:

```
# ip netns exec vrf1 ipsec statusall
Status of IKE charon daemon (strongSwan 5.4.0, Linux 4.4.6, x86_64):
  uptime: 6 seconds, since Jun 09 13:59:13 2016
  malloc: sbrk 1486848, mmap 0, used 451904, free 1034944
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 0
  loaded plugins: charon aes des rc2 sha2 sha1 md5 random nonce x509
revocation constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnskey sshkey
pem openssl fips-prf gmp xcbc cmac hmac curl attr kernel-netlink resolve
socket-default stroke vici updown eap-identity eap-aka eap-aka-3gpp2
eap-simaka-pseudonym eap-simaka-reauth eap-md5 eap-radius xauth-generic
Listening IP addresses:
  10.22.1.101
  10.23.1.101
Connections:
myconnection:  10.23.1.101...10.23.1.201  IKEv2
myconnection:   local:  [10.23.1.101] uses pre-shared key authentication
```

(continues on next page)

(continued from previous page)

```
myconnection:   remote: [10.23.1.201] uses pre-shared key authentication
myconnection:   child:  10.22.1.0/24 === 10.24.1.0/24 TUNNEL
Routed Connections:
myconnection{1}:  ROUTED, TUNNEL, reqid 1
myconnection{1}:   10.22.1.0/24 === 10.24.1.0/24
Security Associations (0 up, 0 connecting):
  none
```

After a successful negotiation:

```
# ip netns exec vrf1 ipsec statusall
Status of IKE charon daemon (strongSwan 5.4.0, Linux 4.4.6, x86_64):
  uptime: 2 minutes, since Jun 09 13:59:13 2016
  malloc: sbrk 2027520, mmap 0, used 477696, free 1549824
  worker threads: 11 of 16 idle, 5/0/0/0 working, job queue: 0/0/0/0, scheduled: 3
  loaded plugins: charon aes des rc2 sha2 sha1 md5 random nonce x509
revocation constraints pubkey pkcs1 pkcs7 pkcs8 pkcs12 pgp dnskey sshkey
pem openssl fips-prf gmp xcbc cmac hmac curl attr kernel-netlink resolve
socket-default stroke vici updown eap-identity eap-aka eap-aka-3gpp2
eap-simaka-pseudonym eap-simaka-reauth eap-md5 eap-radius xauth-generic
Listening IP addresses:
  10.22.1.101
  10.23.1.101
Connections:
myconnection:  10.23.1.101...10.23.1.201  IKEv2
myconnection:   local:  [10.23.1.101] uses pre-shared key authentication
myconnection:   remote: [10.23.1.201] uses pre-shared key authentication
myconnection:   child:  10.22.1.0/24 === 10.24.1.0/24 TUNNEL
Routed Connections:
myconnection{1}:  ROUTED, TUNNEL, reqid 1
myconnection{1}:   10.22.1.0/24 === 10.24.1.0/24
Security Associations (1 up, 0 connecting):
myconnection[1]: ESTABLISHED 54 seconds ago, 10.23.1.101[10.23.1.101]...10.23.1.
↪201[10.23.1.201]
myconnection[1]: IKEv2 SPIs: 465f61cc6e4c6d25_i* fbea95b6b8eaa14e_r, pre-shared␣
↪key reauthentication in 11 hours
myconnection[1]: IKE proposal: AES_CBC_128/HMAC_SHA1_96/PRF_HMAC_SHA1/MODP_1024
myconnection{2}:  INSTALLED, TUNNEL, reqid 1, ESP SPIs: c6fefb62_i ce610510_o
myconnection{2}:  AES_CBC_128/HMAC_SHA1_96, 0 bytes_i, 0 bytes_o, rekeying in 5␣
↪hours
myconnection{2}:   10.22.1.0/24 === 10.24.1.0/24
```

8. Display the Linux IPsec SPs:

```
# ip netns exec vrf1 ip xfrm policy show
src 10.24.1.0/24 dst 10.22.1.0/24
        dir fwd priority 2883 ptype main
        tmpl src 10.23.1.201 dst 10.23.1.101
                proto esp reqid 1 mode tunnel
src 10.24.1.0/24 dst 10.22.1.0/24
        dir in priority 2883 ptype main
        tmpl src 10.23.1.201 dst 10.23.1.101
                proto esp reqid 1 mode tunnel
src 10.22.1.0/24 dst 10.24.1.0/24
        dir out priority 2883 ptype main
        tmpl src 10.23.1.101 dst 10.23.1.201
                proto esp reqid 1 mode tunnel
src 0.0.0.0/0 dst 0.0.0.0/0
        socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
        socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
        socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
        socket out priority 0 ptype main
src ::/0 dst ::/0
        socket in priority 0 ptype main
src ::/0 dst ::/0
        socket out priority 0 ptype main
src ::/0 dst ::/0
        socket in priority 0 ptype main
src ::/0 dst ::/0
        socket out priority 0 ptype main
```

9. Display the negotiated Linux IPsec SAs:

```
# ip netns exec vrf1 ip xfrm state show
src 10.23.1.101 dst 10.23.1.201
        proto esp spi 0xce610510 reqid 1 mode tunnel
        replay-window 32 flag af-unspec
        auth-trunc hmac(sha1) 0x926c0f31eaf115743e3c22521157407442f8e093 96
        enc cbc(aes) 0x3afd5f64605ee03e645cd77fac8b4f30
        anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
src 10.23.1.201 dst 10.23.1.101
        proto esp spi 0xc6fefb62 reqid 1 mode tunnel
        replay-window 32 flag af-unspec
        auth-trunc hmac(sha1) 0x1f0d9a43c813cc8aff0e5b505aaf71682111e216 96
        enc cbc(aes) 0x528ef1cff76437f66660d468dedc9eb6
        anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
```

10. Display the fast path IPsec SPs:

```
<fp-0> vrf-exec 1 ipsec4-spd all
vrf1:
Inbound SPD: 1 rules
1: 10.24.1.0/24 10.22.1.0/24 proto any vr1 protect prio 2883
     link-vr1
     ESP tunnel 10.23.1.201 - 10.23.1.101 reqid=1
     sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
Outbound SPD: 1 rules
1: 10.22.1.0/24 10.24.1.0/24 proto any vr1 protect prio 2883
     link-vr1 cached-SA 0 (genid 0)
     ESP tunnel 10.23.1.101 - 10.23.1.201 reqid=1
     sp_packets=0 sp_bytes=0 sp_exceptions=0 sp_errors=0
```

11. Display the negotiated fast path IPsec SAs:

```
<fp-0> vrf-exec 1 ipsec4-sad all
vrf1:
SAD 2 SA.
1: 10.23.1.201 - 10.23.1.101 vr1 spi 0xc6fefb62 ESP tunnel
     x-vr1 reqid=1 counter 1 (genid 1)
     AES-CBC HMAC-SHA1
     key enc:528ef1cff76437f66660d468dedc9eb6
     digest length: 12
     key auth:1f0d9a43c813cc8aff0e5b505aaf71682111e216
     sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
     sa_replay_errors=0 sa_selector_errors=0
     replay width=32 seq=0x0 - oseq=0x0
     00000000
2: 10.23.1.101 - 10.23.1.201 vr1 spi 0xce610510 ESP tunnel
     x-vr1 reqid=1 counter 1 (genid 2)
     AES-CBC HMAC-SHA1
     key enc:3afd5f64605ee03e645cd77fac8b4f30
     digest length: 12
     key auth:926c0f31eaf115743e3c22521157407442f8e093
     sa_packets=0 sa_bytes=0 sa_auth_errors=0 sa_decrypt_errors=0
     sa_replay_errors=0 sa_selector_errors=0
     replay width=32 seq=0x0 - oseq=0x0
     00000000
```

12. Stop IKE daemons:

```
# ip netns exec vrf1 ipsec stop
```

This command stops IKE daemons and removes all SPs and SAs.

2.4.2 Control Plane OVS

Overview

Control Plane OVS is the 6WIND implementation of Open vSwitch (http://www.openvswitch.org/), based on Open vSwitch v2.9.3.

Open vSwitch provides virtual switching, flow matching, and packet manipulation, configured via an OpenFlow controller or the command line.

Features

Control Plane OVS provides the following capabilities:

- Support for statistics synchronization with the fast path
- Support for hitflags (a flow not hit by the kernel datapath, but hit by the fast path datapath, stays alive)
- Support for direct addition/deletion of flows in the fast path shared memory

Dependencies

6WINDGate modules

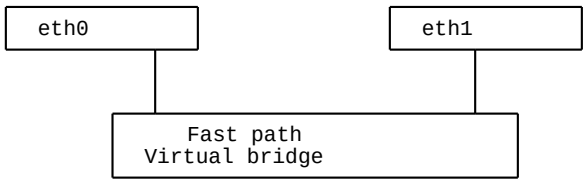
- *Fast Path OVS Acceleration*
- *Linux - Fast Path Synchronization*

Linux

- Open vSwitch kernel version from 2.6.32 to 4.3.

Usage

We will configure a fast path and a virtual bridge between two physical ports as follows:



Note: The output is from a Fedora Core 20 machine.

1. Configure and start the fast path.
2. Start the Linux synchronization.
3. Start the Open vSwitch control plane:

```
# /etc/init.d/openvswitch-switch stop
# /etc/init.d/openvswitch-switch start
```

4. Set the interfaces up and in `promiscuous` mode:

```
# ip link set eth0 up
# ip link set eth0 promisc on
# ip link set eth1 up
# ip link set eth1 promisc on
# ip link set br0 up
```

Note: Setting promiscuous mode on the interfaces is needed only if adding the port in a bridge does not trigger a netlink promiscuous notification. It has to be done before adding the port to the bridge.

---

5. Configure a bridge between two ports:

```
# ovs-vsctl add-br br0
# ovs-vsctl add-port br0 eth0
# ovs-vsctl add-port br0 eth1
```

Note: Depending on your kernel version, a `dmesg` warning may appear when adding a bridge:

```
openvswitch: netlink: Key attribute has unexpected length (type=62, length=4,␣
↪expected=0)
```

This is an expected behavior and does not prevent Open vSwitch from working correctly. It is due to an attempt of `ovs-vswitchd` to detect the kernel's supported features.

See also:

http://openvswitch.org/pipermail/dev/2014-May/040780.html

---

6. **[Optional]** Add an *OpenFlow* controller:

```
# ovs-vsctl set-controller br0 tcp:192.168.0.27:6633
```

The two physical ports `eth0` and `eth1` can now communicate via a fast path and the virtual bridge `br0`.

See also:

- The *Fast Path Baseline* documentation

- The *Linux - Fast Path Synchronization* documentation
- Your *OpenFlow* controller documentation
- The *OpenStack* documentation
- The Open vSwitch documentation
- The *Fast Path OVS Acceleration* documentation

2.4.3 Control Plane Routing

Overview

Control Plane Routing enables route management over a wide variety of routing protocols. It is supported by the `zebra` daemon, from the open source FRR (https://frrouting.org/) project.

You can manage routes via configuration files, or interactively in a specific CLI.

See also:

For more information on FRR (https://frrouting.org/), see the project's online documentation (https://frrouting.org/user-guide/).

Features

- BGP, BGP4+
- OSPFv2, OSPFv3
- RIP, RIPng
- Cross-VRF
- Static Routes
- ECMP
- PBR
- MPLS LDP
- BGP L3VPN
- BGP Flowspec
- Point to Multipoint GRE interfaces
- NHRP
- DMVPN with IPsec

Dependencies

6WINDGate modules

- *Fast Path Forwarding IPv4*
- *Fast Path Forwarding IPv6* (for IPv6)
- *Fast Path Policy-Based Routing* (for PBR and BGP Flowspec)
- *Fast Path MPLS* (for MPLS LDP and BGP L3VPN)
- *Linux - Fast Path Synchronization*

Usage

Starting Control Plane Routing

`frrinit.sh` or `systemctl`

Description

`frrinit.sh` is the startup script in charge of starting the underlying daemons of FRR.

- starts the relevant daemon with their default options
- stops the relevant daemon

By default, `frrinit.sh` will start the following 7 routing protocol daemons: `zebra`, `ospf`, `ospf6`, `rip`, `ripng`, `bgp`, `ldp`.

This script can be used, if you want to run FRR in standalone mode, that is to say, without the nc-cli framework. However, if you already have the nc-cli started, you should ensure that the nc-cli configuration does not interfere with the routing protocol configuration: for instance, the daemons should not have been launched by nc-cli. This script is also used by `systemctl` to start FRR in standalone mode.

Synopsis

When using `frrinit.sh`:

```
/usr/bin/frrinit.sh start|stop
```

When using `systemctl`:

```
systemctl start|stop frr
```

Parameters

`start`
: Start the above mentioned daemons

`stop`
: Stop the above mentioned daemons

`start frr.service`
: Start the above mentioned daemons

`stop frr.service`
: Stop the above mentioned daemons

  For more information about FRR (https://frrouting.org/) daemons startup, see the online documentation (https://frrouting.org/user-guide/).

Configuration

Prior to starting the daemons, you can provide a configuration file for each of the associated daemons. The configuration file must be located under `/etc/frr/` path. The startup procedure depicted above does not take into account vty integrated configuration. That is to say that as many files as daemons will have to be populated. The configuration files are the following ones:

- zebra.conf for `zebra`
- ospfd.conf for `ospf`
- ospf6d.conf for `ospf6`
- ripd.conf for `rip`
- ripng.conf for `ripng`
- bgpd.conf for `bgp`
- ldpd.conf for `ldp`.

When the daemons are started, you can alter the running configuration interactively by passing commands to the relevant daemon via CLI. The CLI syntax and the configuration file syntax are quite similar.

systemctl start frr.service

`systemd`

Accessing the CLI for runtime configuration

You can access each daemon's CLI for runtime configuration via `vtysh` program. It allows configuring the started daemons: `zebra`, `ospf`, `ospf6`, `rip`, `ripng`, `bgp`, `ldp`.

```
$ vtysh
```

See also:

For more information about the CLI modes, see the online documentation (https://frrouting.org/user-guide/Virtual-Terminal-Interfaces.html#Virtual-Terminal-Interface).

2.5 High Availability

2.5.1 HA VRRP

Overview

HA VRRP is the 6WIND implementation of VRRP. It is based on the *Keepalived* open source project.

VRRP provides a way, for a set of routers, to control a virtual IP address and MAC address, and to provide automatic failover. This address may be used by hosts to access specific services, a static default gateway, for instance. VRRP provides higher service availability without requiring automatic reconfiguration of end hosts.

Features

- VRRP v2 **RFC 3768** (https://tools.ietf.org/html/rfc3768.html) (IPv4 only).
  - Automatic election of master router.
  - Preemption when a backup server with higher priority than master is present.
  - Advertisement interval (to indicate that master is still in service) from 1 to 255 seconds.
- VRRP v3 **RFC 5798** (https://tools.ietf.org/html/rfc5798.html) (IPv4 and IPv6).
  - Advertisement interval (to indicate that master is still in service) starting from 100 milliseconds.
- Ability to create several VRRP groups and to synchronize them: a router, belonging to different groups, has the same state (master or backup) in any group.
- Authentication (Simple Text Password or IP Authentication Header) as defined in VRRP v1 **RFC 2338** (https://tools.ietf.org/html/rfc2338.html).
- Remote management with SNMP (Simple Network Management Protocol).

Dependencies

6WINDGate modules

- *Fast Path Baseline* (MACVLAN feature)
- *Linux - Fast Path Synchronization*

Linux

Linux >= 3.2 to have commit 729e72a10930 (http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=729e72a109 ("macvlan: receive multicast with local address"). Otherwise, HA VRRP is not compliant with **RFC 3768** (https://tools.ietf.org/html/rfc3768.html) (VRRP routers are not able to share the same MAC address without this commit).

Usage

You can manage HA VRRP devices under Linux.

1. To synchronize Linux and the fast path, start the cache manager and the fast path manager:

```
# fpmd
# cmgrd
```

2. Start `keepalived` with appropriate options:

```
# keepalived [options]
```

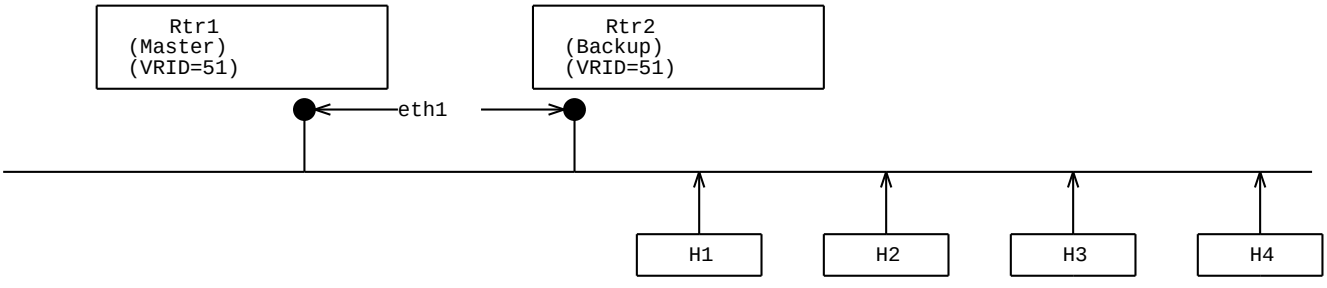
Note: Available options are displayed with the command: `keepalived -h`

Example: implementing one virtual router with two VRRP routers

Here is a very simple example that implements `Sample Configuration 1`, provided in **RFC 3768** (https://tools.ietf.org/html/rfc3768.html) section 4.1, with a VRRP router with MAC address `00:00:5e:00:01:33` and IP address `10.22.0.1`.

Network architecture

In the architecture below, *Rtr1* is the master VRRP router, and *Rtr2* is the backup VRRP router.



Launching keepalived on the master router

Perform the steps below on the master router.

1. Create the `Rtr1.conf` `keepalived` configuration file with the following content:

```
global_defs {
 router_id DEMO_VRRP_ROUTER
 dynamic_interfaces # to be notified of interface changes
 # wait for 30s at startup before starting election to avoid bug
 # with some interfaces (LACP for instance) on which Rx is off and
 # Tx is on at startup.
 # It avoids sending master state from the starting router because it does
 # not receive any VRRP master from other member
 vrrp_startup_delay 30
 disable_local_igmp
}
vrrp_instance VI_1 {
 state BACKUP           # start as backup
 interface eth1                    # ethernet link where hosts and VRRP routers␣
↪are connected
 use_vmac                          # to use macvlan
 virtual_router_id 51              # to use the 00:00:5e:00:01:{virtual_router_id}␣
↪mac address (33 hexadecimal = 51 decimal)
 priority 200                      # priority of the router
 advert_int 1                      # VRRP advertisement interval
 virtual_ipaddress {
  10.22.0.1/24                     # VRRP ip address
 }
}
```

See also:

For more information, see keepalived.conf SYNOPSIS (https://github.com/acassen/keepalived/blob/v2.0.10/doc/keepalive
and keepalived.conf samples (https://github.com/acassen/keepalived/tree/v2.0.10/doc/samples).

2. Launch the `keepalived` daemon for VRRP with the `Rtr1.conf` configuration file:

```
# keepalived -f /path/to/Rtr1.conf
```

Note: If you do not specify a custom configuration file, `keepalived` tries to read the default configuration file `/etc/keepalived/keepalived.conf`.

Launching keepalived on the backup router

Perform the steps below on the backup router.

1. Create the Rtr2.conf keepalived configuration file with the following content:

```
global_defs {
 router_id DEMO_VRRP_ROUTER
 dynamic_interfaces # to be notified of interface changes
 vrrp_startup_delay 30
 disable_local_igmp
}
vrrp_instance VI_1 {
 state BACKUP
 interface eth1
 use_vmac
 virtual_router_id 51
 priority 100                    # Set to 200 in *Rtr1.conf*
 advert_int 1
 virtual_ipaddress {
  10.22.0.1/24
 }
}
```

2. Launch the keepalived daemon for VRRP with the Rtr2.conf configuration file:

```
# keepalived -f /path/to/Rtr2.conf
```

Configuring the two routers

Perform the steps below on each of the two routers.

1. Check that the VRRP interface exists:

```
# fp-cli iface
...
849:vrrp.51 [VR-0] ifuid=0x5153311a (virtual) <UP|RUNNING|FWD4|FWD6> (0x63)
        type=macvlan mac=00:00:5e:00:01:33 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
        mode private, link eth1
...
```

2. Allow IP forwarding:

```
# sysctl -w net.ipv4.ip_forward=1
```

3. Disable reverse path filtering on interfaces to which the VRRP routers are linked:

```
# sysctl -w net.ipv4.conf.eth1.rp_filter=0
```

4. Disable `iptables` rules that block multicast traffic:

   To clear all `iptables` rules:

```
# iptables -F
```

5. Enable `arp_ignore` on interfaces to which the VRRP routers are linked:

```
# sysctl -w net.ipv4.conf.eth1.arp_ignore=1
```

   Otherwise, the linked interface answers ARP requests related to the VRRP MAC address.

6. If you do not receive *Netlink* notifications from the lower interface to which the VRRP routers are linked, set this interface in `promiscuous` mode:

```
# ip link set eth1 promisc on
```

7. Check that interface `eth1` has the flag `PROMISC`:

```
# fp-cli iface
...
96:eth1 [VR-0] ifuid=0x6008c1d2 (port 1) <UP|RUNNING|PROMISC|FWD4|FWD6> (0x73)
        type=ether mac=00:02:02:00:00:21 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
...
```

HA VRRP is now properly configured. The master VRRP router *Rtr1*, and the backup VRRP router *Rtr2* now implement one virtual router on a simple network, as specified in **RFC 3768** (https://tools.ietf.org/html/rfc3768.html), section 4.1.

Using VRRP with IKE/IPsec HA or/and Firewall/NAT HA

In order to make IKE/IPsec HA or/and Firewall/NAT HA synchronization work properly:

- dataplane and some control-plane trafic must be only processed on the master router.
- IKE/IPsec HA needs to know what is the state of VRRP.

This section describes how to set up a keepalived script to do so.

1. Create a keepalived notify script `/etc/keepalived/notify.sh`:

```
cat>/etc/keepalived/notify.sh <<\EOF
#!/bin/sh
```

(continues on next page)

(continued from previous page)

```
FILE=/run/ha-groups/ikeha
STATE="$3"

mkdir -p $(dirname "$FILE")
case "$STATE" in
   "MASTER")
      echo "master" > "$FILE"
      iptables-save -t filter | grep -v __internal__:keepalived | iptables-
↪restore -T filter
      ip6tables-save -t filter | grep -v __internal__:keepalived | ip6tables-
↪restore -T filter
      ;;
   *)
      iptables -I INPUT -p esp -m comment --comment "__internal__:keepalived" -j␣
↪DROP
      iptables -I INPUT -p ah -m comment --comment "__internal__:keepalived" -j␣
↪DROP
      iptables -I INPUT -p udp -m multiport --dports 500,4500 -m comment --
↪comment "__internal__:keepalived" -j DROP
      iptables -I FORWARD -m comment --comment "__internal__:keepalived" -j DROP
      ip6tables -I INPUT -p esp -m comment --comment "__internal__:keepalived" -j␣
↪DROP
      ip6tables -I INPUT -m ah -m comment --comment "__internal__:keepalived" -j␣
↪DROP
      ip6tables -I INPUT -p udp -m multiport --dports 500,4500 -m comment --
↪comment "__internal__:keepalived" -j DROP
      ip6tables -I FORWARD -m comment --comment "__internal__:keepalived" -j DROP
      echo "backup" > "$FILE"
      ;;
esac
EOF
```

2. Give the root execution permission to the script:

```
# chown root:root /etc/keepalived/notify.sh
# chmod u+x /etc/keepalived/notify.sh
```

3. Create an unique VRRP group on both routers that contains all the VRRP instances:

```
vrrp_sync_group mygroup {
  group {
    VI_1
    VI_2
```

(continues on next page)

(continued from previous page)

```
  }
  notify /etc/keepalived/notify.sh
}
```

4. Add the activity file to /etc/ike/strongswan.d/charon/ha.conf inside ha { } if necessary:

```
activity_file = /run/ha-groups/ikeha
```

2.6 Linux - Fast Path Synchronization

2.6.1 Linux - Fast Path Synchronization

Overview

Linux - Fast Path Synchronization provides transparent synchronization of fast path protocols with Linux.

It provides the cache manager, which is responsible for listening to Linux events and transmitting them to the fast path manager (provided as part of *Fast Path Baseline*) to configure the fast path.

Statistics are synchronized through the fast path statistics feature (provided as part of *Fast Path Baseline*).

Entry aging (to prevent the expiration of dynamic entries such as ARP, NDP, conntrack, etc. when they are used in the fast path) is handled by the hit flags synchronization feature (provided as part of *Fast Path Baseline*).

Features

- Queuing mechanism to cope with frequent changes
- Graceful restart
- Monitoring console access via a UNIX socket
- Plugins support
- Supported features list is automaticaly synchronized with the fast path manager

Dependencies

6WINDGate modules

- *Fast Path Baseline*

Before starting Linux - Fast Path Synchronization, you must start the fast path.

Linux

- iptables-devel
- libnl3 (https://www.infradead.org/~tgr/libnl/files/libnl-3.2.25.tar.gz) >= 3.2.25

Plugins

To monitor additional modules, the Linux - Fast Path Synchronization daemon can be extended with plugins (shared libraries loaded at startup.)

By default, the daemon loads all shared libraries that match the pattern */usr/lib/cmgr/*.so*. You can set the CMGR_PLUGINS environment variable to change the default plugin location pattern.

Available plugins:

- **VNB**
- **OVS acceleration**
- **IPsec output delegation and SA migration**

Usage

The `linux-fp-sync.sh` script starts:

- the fast path manager,
- the fast path statistics feature,
- the hit flags synchronization feature,
- the cache manager.

You can also launch manually the cache manager.

Starting Linux - Fast Path Synchronization

1. Adapt the Linux - Fast Path Synchronization default configuration file (`/etc/linux-fp-sync.env`) to your needs.
2. Start Linux - Fast Path Synchronization:

```
# linux-fp-sync.sh start
```

Note: To use a custom configuration file, use the CONF_FILE_linux_fp_sync environment variable. For instance:

```
# CONF_FILE_linux_fp_sync=/path/to/conf/conf_file linux-fp-sync.sh start
```

Stopping Linux - Fast Path Synchronization

- To stop Linux - Fast Path Synchronization:

```
#  linux-fp-sync.sh stop
```

Restarting Linux - Fast Path Synchronization

- To restart Linux - Fast Path Synchronization:

```
#  linux-fp-sync.sh restart
```

Displaying the Linux - Fast Path Synchronization status

- To display the current status of running Linux - Fast Path Synchronization threads:

```
# linux-fp-sync.sh status
```

- To display the current status of running Linux - Fast Path Synchronization threads and of the current installation (inserted `.ko`, for example):

```
# linux-fp-sync.sh status complete
```

Starting manually the cache manager

The cache manager daemon:

- listens to network changes in Linux, and,
- forwards network changes to the fast path manager daemon via the FPC API.

To start the cache manager, enter:

```
# cmgr.sh start
```

Providing options

The `cmgr.sh` script reads the `/etc/cmgr.env` default configuration file before actually starting the cache manager.

You can edit this file to customize the cache manager configuration.

Note: To use a custom configuration file, use the CONF_FILE_cmgr environment variable. For instance:

```
# CONF_FILE_cmgr=/path/to/conf/conf_file cmgr.sh start
```

If a variable specified in the configuration file already exists in the environment (for instance, by calling `HA=true cmgr.sh start`), the latter will be used.

Note: To have configuration file variables supersede global environment variables, specify them in the configuration file according to the following syntax:

```
HA=true
```

instead of:

```
: ${HA:=true}
```

You can set the most common options via a dedicated variable such as `DEBUG`.

To set common options, use the `CMGR_OPTIONS` variable and specify them using the appropriate option delimiter (`-b` for socket buffer size, `-I` for the cache manager identification number, etc.).

Parameters to disable the synchronization per feature

Default is to synchronize all features supported by the fast path. Runtime parameters can be set in `CMGR_OPTIONS` to disable some of them.

`--disable-sync-vxlan`

This will disable the synchronization of Linux VXLAN.

`--disable-sync-bridge`

This will disable the synchronization of Linux bridge.

`--disable-sync-ebtables`

This will disable the synchronization of Linux netfilter ebtables rules.

`--disable-sync-gre`

This will disable the synchronization of Linux GRE.

--disable-sync-vlan

This will disable the synchronization of Linux VLAN.

--disable-sync-macvlan

This will disable the synchronization of Linux macvlan.

--disable-sync-lag

This will disable the synchronization of Linux Bonding.

--disable-sync-tunnel

This will disable the synchronization of Linux tunnels.

--disable-sync-ipsec

This will disable the synchronization of Linux XFRM IPsec, including Linux VTI.

--disable-sync-svti

This will disable the synchronization of Linux VTI.

--disable-sync-nat

This will disable the synchronization of Linux netfilter NAT rules.

--disable-sync-netfilter

This will disable the synchronization of Linux netfilter IPv4 / IPv6 rules, including NAT and conntrack.

--disable-sync-bpf

This will disable the synchronization of Linux BPF.

--disable-sync-mcast

This will disable the synchronization of Linux IPv4 and IPv6 multicast.

--disable-sync-ipv6

This will disable the synchronization of Linux IPv6 routing.

--disable-sync-conntrack

This will disable the synchronization of Linux conntrack.

Alternatively, it is possible to specify the list of features to synchronize in a single option:

-S, --sync-mask <mask>
Specify the list of features to synchronize.

Feature	flag
VXLAN	0x00000001
bridge	0x00000002
VLAN	0x00000004
MACVLAN	0x00000008
LAG	0x00000010
ebtables	0x00000020
GRE/GREtap	0x00000040
tunnel	0x00000080
IPsec	0x00000100
SVTI	0x00000200
filter/audit	0x00000400
conntrack	0x00000800
NAT	0x00001000
BPF (tap)	0x00002000
tap pattern only	0x00004000
multicast routes	0x00008000
IPv6	0x00010000

Parameters to enable feature options

Some features support options, disabled by default. The following options can be enabled:

`-A, --sync-bpf-all`
This is an option of the BPF (tap) feature.

Default is to synchronize BPF only for the applications: tcpdump, ethereal, wireshark, tshark, nmap.

The option -A forces the BPF synchronization for any application.

Alternately, you can set `: ${BPF_OPT:=true}` in the configuration file.

`-R, --sync-ipsec-replay`
This is an option of the IPsec feature.

It enables the synchronization of IPsec replay information (SA input and output sequence numbers) from Linux to fast path. This option is used for IPsec HA (High Availability).

Alternately, you can set `: ${HA_IPSEC:=true}` in the configuration file.

This option may be dynamically enabled or disabled without restarting the cache manager. See *Configuring HA IPsec dynamically*.

Parameters for debug purpose

Here are parameters useful for debug purpose:

-d <mask>
: Debug mask value.

  Alternately, you can set `: ${DEBUG:=<mask>}` in the configuration file.

-F
: Foreground.

-b <val>
: Custom value of the socket buffer size, default is 2M.

-l <val>
: Custom value of the netlink socket buffer size, default is 64M.

-h
: Display the full list of options.

-o
: Display compilation options and exit.

-I <val>
: Specify a number to identify a cache manager instance (only when the control plane manages more than one fast path). The value is called the instance id of the cache manager instance.

  If a value is specified, syslog logs cmgrd<val>, and the console is at /tmp/.cmgrd<val>.

Dumping or changing current configuration

You can dump statistics such as netlink received messages and debug the queuing mechanism in a console.

Accessing the console

The cache manager console is reachable using the 6WIND `daemonctl` tool:

```
# daemonctl cmgrd <daemon_command>
```

The following example uses `daemonctl` to display available commands:

```
# daemonctl cmgrd help
help       - Show help
?          - Show help
quit       - Quit the shell
show       - show statistics
ipsec      - ipsec commands
```

(continues on next page)

(continued from previous page)

```
# daemonctl cmgrd show
pid        - show pid
netlink    - show netlink packets
queue      - show queued msg
conf       - show conf variables
modules    - show registered modules
interfaces - show registered interfaces
sync-features - show synchronized feartures
```

Dumping statistics and configuration

```
# daemonctl cmgrd show netlink
Dump netlink socket statistics:
netlink socket name                    packets received
netlink-route-listen-0                 16
    RTM_NEWLINK                        1
    RTM_NEWADDR                        2
    RTM_NEWROUTE                       9
    RTM_DELROUTE                       4

netlink-route-cmd-0                    47
    RTM_NEWLINK                        8
    RTM_NEWADDR                        3
    RTM_NEWROUTE                       15
    RTM_NEWNEIGH                       2
    RTM_[80]                           19

netlink-xfrm-listen-0                  0

netlink-xfrm-cmd-0                     0

netlink-vnb-listen-0                   3
    VNB_C_DUMP                         2
    VNB_C_NEW                          1

netlink-netfilter-conntrack-lis        3
    IPCTNL_MSG_CT_NEW                  3

netlink-audit-listen-0                 34
    AUDIT_[2]                          1
    AUDIT_[1300]                       11
    AUDIT_[1320]                       11
```

(continues on next page)

(continued from previous page)

```
    AUDIT_NETFILTER_CFG                 11
```

```
# daemonctl cmgrd show queue
Queue information
- sent: 98
- directly: 5
- in-queue: 0
- highest in-queue: 89
- has blocked: 0
- partially sent: 0
- errors: 0
- ev armed: 0

command_show_queue: address=0x1318920
        current=0x7f7e5aba4000
        chk_count=1
        chk_total_count=1
        obj_count=0
        obj_total_count=191
        obj_malloc_count=0
        obj_ignored_free=0
        next_free=0x7f7e5aba400c
```

```
# daemonctl cmgrd show modules
xfrm-migrate
vnb
```

```
# daemonctl cmgrd show interfaces
Interfaces list:
br0 vrfid 0 (ifindex: 15, ifuid: 0x42e9f282)
        type: 6, subtype: 5, flags: 0x60, mtu: 1500
        master_ifuid: 0x0, vnb_nodeid: 0x8
        in_l_bond: no
eth1 vrfid 0 (ifindex: 11, ifuid: 0x33117022)
        type: 6, subtype: 0, flags: 0x60, mtu: 1500
        master_ifuid: 0x82f2e942, vnb_nodeid: 0x4
        in_l_bond: no
fpn0 vrfid 0 (ifindex: 10, ifuid: 0x64247322)
        type: 6, subtype: 0, flags: 0x63, mtu: 1500
        master_ifuid: 0x0, vnb_nodeid: 0x3
        in_l_bond: no
eth0 vrfid 0 (ifindex: 2, ifuid: 0x61a1e72)
        type: 6, subtype: 0, flags: 0x63, mtu: 1500
```

(continues on next page)

(continued from previous page)

```
        master_ifuid: 0x0, vnb_nodeid: 0x2
        in_l_bond: no
lo vrfid 0 (ifindex: 1, ifuid: 0x754c6fa8)
        type: 24, subtype: 0, flags: 0x63, mtu: 65536
        master_ifuid: 0x0, vnb_nodeid: 0x0
        in_l_bond: no
Bridge interfaces list:
eth1 vrfid 0 (ifindex: 11, ifuid: 0x33117022)
        type: 249, subtype: 0, master_ifuid: 0x82f2e942
```

```
# daemonctl cmgrd show sync-features
Cache manager uses FPM configuration

Features             Status (enabled/disabled)
vxlan                enabled
bridge               enabled
gre                  enabled
vlan                 enabled
macvlan              enabled
lag                  enabled
ebtables             enabled
tunnel               enabled
ipsec                enabled
svti                 enabled
netfilter            enabled
conntrack            enabled
nat                  enabled
bpf                  disabled
multicast            enabled
ipv6                 enabled
```

Configuring HA IPsec dynamically

The synchronization of IPsec HA related information (IPsec SA sequence numbers) may be enabled or disabled dynamically without restarting the cache manager.

To display the current IPsec HA status, use the following command:

```
# daemonctl cmgrd ipsec ha show
IPsec High Availability disabled
```

To enable the support of IPsec HA:

```
# daemonctl cmgrd ipsec ha enable
```

To disable the support of IPsec HA:

```
# daemonctl cmgrd ipsec ha disable
```

2.6.2 FPTUN-eBPF (Fast Path Tunneling Protocol over eBPF)

Overview

FPTUN-eBPF uses the Linux eBPF architecture to drive packets between the fast path and the Linux kernel stack.

It provides functionalities that used to be implemented in FPTUN kernel module:

- special exception path to skip in Linux the work done in fast path: VLAN, GRE, VXLAN, ...
- metadata transfer like "skb->mark"
- "tcpdump" to capture packets processed at fast path level

Features

The following exception types are supported:

- FPTUN_BASIC_EXCEPT: exception used to inject a L2 packet in a fast path port interface at ingress.
- FPTUN_ETH_INPUT_EXCEPT: exception used to inject a L2 packet in an interface at ingress.
- FPTUN_IFACE_INPUT_EXCEPT: exception used to inject a L3 packet in an interface at ingress.
- FPTUN_IPV4_IPSECDONE_INPUT_EXCEPT: exception used to inject a deciphered IPv4 packet in an interface at ingress.
- FPTUN_IPV6_IPSECDONE_INPUT_EXCEPT: exception used to inject a deciphered IPv6 packet in an interface at ingress.
- FPTUN_TAP: exception used to inject a packet for tcpdump capture.
- FPTUN_ETH_SP_OUTPUT_REQ: used to inject packets from the Linux kernel stack to the fast path.

Dependencies

6WINDGate modules

- *Fast Path Baseline*

Linux

- tc eBPF support is a kernel patch (upstream 4.0).
  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=d23b8ad8ab23
- tc eBPF bpf_skb_store_bytes support is a kernel patch (upstream 4.1).
  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=608cd71a9c7c9
- tc eBPF bpf_redirect support is a kernel patch (upstream 4.2).
  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=3896d655f4d49
- tc eBPF BPF_F_INGRESS redirect flag is a kernel patch (upstream 4.5).
  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=781c53bc5d56
- tc eBPF bpf_redirect support from a L2 devive to a L3 device is a kernel patch (upstream 4.9).
  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=4e3264d21b909
- tc eBPF bpf_skb_load_bytes support is a kernel patch (upstream 4.5).
  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=05c74e5e53f6c
- tc eBPF bpf_skb_change_tail support is a kernel patch (upstream 4.9).
  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=5293efe62df81
- tc eBPF bpf_skb_adjust_room support is available since kernel version 5.2.
  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=14aa31929b72
  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=2278f6cc151a
  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=58dfc900faff

Design

Software architecture

FPTUN-eBPF uses the Linux eBPF architecture to drive packets between the fast path and the Linux kernel stack.

Path from the fast path to the Linux kernel stack

A dummy virtual interface (fptun0) is created in the current VRF and a tc eBPF filter is attached on egress. This filter is also attached at ingress on all fast path port interfaces. If the vrf of the packet is the same than the vrf of the input port, the fast path sends packets to the input port else it sends packets to this fptun0 interface using an AF_PACKET/SOCK_RAW socket. Statistics can be shown with the following commands:

```
# tc -s filter show dev eth0 ingress
# tc -s filter show dev fptun0 egress
```

The FPTUN-eBPF metadata (from the fast path) are stored in a trailer at the end of the packet. The eBPF program get them (exception type, targeted interface, etc.) from this trailer. It removes them before injecting the packet into the Linux stack.

The bpf_redirect() function is used to redirect packets from fast path port (ingress) or fptun0 (egress) to the targeted interface (at ingress or egress).

Below is a description of the supported exception types.

FPTUN_BASIC_EXCEPT

This exception is used to inject a L2 packet in an interface (a fast path port) at ingress.

Example of a packet flow for a datagram destinated to a local fast path port interface:

```
eth0 physical in-interface (FPTUN_BASIC_EXCEPT)
  --> fp_process_input (FPTUN_BASIC_EXCEPT)
  --> fp_ether_input (FPTUN_BASIC_EXCEPT)
  --> fp_ip_input (FPTUN_BASIC_EXCEPT)
  --> fp_ip_input_demux (FPTUN_BASIC_EXCEPT)
  --> fp_ip_bulk_exception (FPTUN_BASIC_EXCEPT)
  --> fp_send_fptunebpf (FPTUN_BASIC_EXCEPT)
  --> eth0/ingress
  --> _tc_fptun_ebpf (FPTUN_BASIC_EXCEPT)
  --> bpf_skb_change_tail (remove fptun-ebpf trailer)
  --> netif_rx_internal (eth0)
```


FPTUN_ETH_INPUT_EXCEPT

This exception is used to inject a L2 packet in an interface at ingress.

Example of a packet flow for a datagram destinated to a local VLAN interface:

```
eth0 physical in-interface (FPTUN_BASIC_EXCEPT)
  --> fp_process_input (FPTUN_BASIC_EXCEPT)
  --> fp_ether_input (FPTUN_BASIC_EXCEPT)
  --> fp_vlan_input (FPTUN_BASIC_EXCEPT)
  --> fp_vlan_bulk_process (FPTUN_BASIC_EXCEPT => FPTUN_ETH_INPUT_EXCEPT)
  --> fp_ether_input (FPTUN_ETH_INPUT_EXCEPT)
  --> fp_ip_input (FPTUN_ETH_INPUT_EXCEPT)
  --> fp_ip_input_demux (FPTUN_ETH_INPUT_EXCEPT)
  --> fp_ip_bulk_exception (FPTUN_ETH_INPUT_EXCEPT)
  --> fp_send_fptunebpf (FPTUN_ETH_INPUT_EXCEPT)
  --> fptun0/egress
  --> _tc_fptun_ebpf (FPTUN_ETH_INPUT_EXCEPT)
  --> bpf_skb_change_tail (remove fptun-ebpf trailer)
  --> bpf_redirect (eth0 / ingress)
  --> dev_forward_skb (eth0)
  --> netif_rx_internal (eth0)
```

FPTUN_IFACE_INPUT_EXCEPT

This exception is used to inject a L3 packet in an interface at ingress.

FPTUN_IPV4_IPSECDONE_INPUT_EXCEPT

This exception is used to inject a deciphered IPv4 packet in an interface at ingress.

FPTUN_IPV6_IPSECDONE_INPUT_EXCEPT

This exception is used to inject a deciphered IPv6 packet in an interface at ingress.

FPTUN_TAP

This exception is used to inject packets in opened AF_PACKET sockets on a specified interface and drop them before they enter the Linux stack. An eBPF program is setup at ingress of the tapped interface. This program drops all packets sent through this exception type. An optional trailer can be put ("FASTPATH OFFLOAD" in ascii) at the end of the packet. It helps to indentify if packets are tapped in the fast path or in the kernel.

Example of a packet flow for a datagram tapped on an input interface:

```
eth0 physical in-interface (FPTUN_BASIC_EXCEPT)
  --> fp_process_input (FPTUN_BASIC_EXCEPT)
  --> fp_tap_bulk (FPTUN_BASIC_EXCEPT)
  --> fp_prepare_tap_exception (m_dup / FPTUN_TAP)
  --> fp_sp_exception (FPTUN_TAP)
  --> fp_send_fptunebpf (FPTUN_TAP)
  --> fptun0/egress
  --> _tc_fptun_ebpf (FPTUN_TAP)
  --> bpf_skb_store_bytes (set trailer to "FASTPATH OFFLOAD")
  --> bpf_redirect (eth0 / ingress)
  --> dev_forward_skb (eth0)
  --> netif_rx_internal (eth0)
  --> __netif_receive_skb_core (eth0)
  --> deliver_skb (=> AF_PACKET sockets)
  --> sch_handle_ingress
  --> _tc_fptun_ebpf (drop)
```

Path from the Linux kernel stack to the fast path

A tc eBPF filter is attached on egress of all fast path port interfaces. It is used to add a FPTUN-eBPF header to carry metadata like skb->mark. Statistics can be shown with the following command:

```
# tc -s filter show dev eth0 egress
```

The FPTUN-eBPF metadata (from the Linux kernel stack) are stored in an FPTUN-eBPF header inserted between the L2 headers (Ethernet header and VLAN headers) and the L3 headers (IPv4, IPv6, MPLS, ...) of the packet. The fast path removes this header before processing the packet.

FPTUN_ETH_SP_OUTPUT_REQ

This exception is used to carry metadata to fast path for packet sent by the Linux kernel stack.

Example of a packet flow for a datagram sent by the Linux kernel stack to a fast path port interface:

```
eth0 tuntap out-interface
  --> __dev_queue_xmit
  --> sch_handle_egress
  --> _tc_fptun_mark
  --> bpf_skb_adjust_room (add fptun-ebpf header)
  --> __dev_xmit_skb()
  --> fpn_fpvi_vhost_input
  --> fp_process_soft_input
  --> fp_fptun_ebpf_header_input (remove fptun-ebpf header)
  --> fp_process_soft_output
  --> eth0 physical out-interface
```

2.6.3 Linux - Fast Path Synchronization - VRF

Overview

VRF is a Virtual Routing and Forwarding framework built on top of Linux network namespaces.

Virtual Routing and Forwarding (VRF) is an IP technology that allows multiple instances of a routing table to work simultaneously within the same router. Thus, overlapping IP addresses do not conflict with each other.

Linux - Fast Path Synchronization - VRF is a compatibility layer between network namespaces and 6WIND VRF. Routing or management daemons use it to configure the kernel and the cache manager to replicate the kernel configuration into the fast path.

Features

- Overall initialization of VRF instances
- API allowing the management of VRF instances at user space level and kernel module level

Dependencies

Linux

- Linux - Fast Path Synchronization - VRF relies on your Linux distribution's support of network namespaces. The *CONFIG_NET_NS* variable must be set to yes (*y*) in the kernel configuration.
- For Linux versions lower than 3.12, the following patches are required to allow cross-VRF through IP and IPv6 tunnels:

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=0bd8762

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=6c742e7

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=5e6700b
- Link-level VRF instances' identifiers are passed to the user space via the *IFLA_AF_SPEC* attribute, which requires the following patch:

  http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=f8ff182c716c
- libnl3 (https://www.infradead.org/~tgr/libnl/files/libnl-3.2.25.tar.gz) >= 3.2.25

API

Userspace API: *libvrf*

The *libvrf* library allows to manage and monitor VRF instances.

Initialize:

```
int libvrf_init(void);
```

Enter another VRF instance to run subsequent commands within this instance:

```
int libvrf_change(int vrfid);
```

Go back to the previous VRF instance:

```
int libvrf_back(void);
```

Return the file descriptor bound to a given VRF instance. The value can be used in the netlink API to fill the *IFLA_NET_NS_FD* attribute.

```
int libvrf_get_fd(int vrfid);
```

Iterate over all existing VRF instances and execute the callback function:

```
int libvrf_iterate(void( *callback)(int vrfid, void *data), void *data);
```

Initialize the monitoring system:

```
int libvrf_monitor_init(void);
```

Monitor adding and removing VRF instances, and call associated callbacks:

```
int libvrf_monitor_event(void( *add)(int vrfid, void *data),
                         void( *del)(int vrfid, void *data),
                         void  *data);
```

Usage

Initialization

1. Load and initalize the VRF module:

   ```
   # vrf.sh start
   ```

vrfctl

Basic functions

Description

Create, delete, list and monitor VRF instances.

Synopsis

```
vrfctl add < auto-vrfid | VRFID > [ vfrname NAME ] [ SCRIPT ]
vrfctl del < VRFID | vfrname NAME > [ SCRIPT ]
vrfctl list [ VRFID | vrfname NAME ]
vrfctl monitor
```

Parameters

auto-vrfid | VRFID ID of the VRF instance to create or remove. If 'auto-vrfid' is specified (only for the command 'add'), the highest existing vrfid plus one is used.

NAME Optional. Name of the VRF instance.

SCRIPT Optional. Custom script to execute when the VRF instance is created or deleted.

Note: By default, the script `linux-fp-sync-vrf.sh` must be used to manage *Linux - Fast Path Synchronization*. You can adapt it to your needs.

Advanced functions

vrfctl bind

Description

Bind an existing network namespace to a new VRF instance.

Note: Instances are deleted via 'vrfctl del', as for VRF instances created via 'vrfctl add'.

Synopsis

```
vrfctl bind < file FILE | ipnetns IPNETNSNAME | pid PID > < auto-vrfid | VRFID > [␣
↪vfrname NAME ] [ SCRIPT ]
```

Parameters

FILE A path to a network namespace file.

IPNETNSNAME An iproute2 netns name.

PID A PID.

auto-vrfid | VRFID ID of the VRF instance to create. If 'auto-vrfid' is specified, the highest existing vrfid plus one is used.

NAME Optional. Name of the VRF instance.

SCRIPT Optional. Custom script to execute when the VRF instance is created or deleted.

Note: By default, the script `linux-fp-sync-vrf.sh` must be used to manage *Linux - Fast Path Synchronization*. You can adapt it to your needs.

ip netns exec

Description

Execute a command in a VRF instance.

Synopsis

```
ip netns exec <vrf name> <cmd>
```

Parameters

vrf name Name of the VRF instance in which the command must be executed.

cmd Command to be executed.

Configuration example

We will configure a router with four interfaces:

- *eth9_0* and *eth9_1* are in the *vrf0* VRF instance, and
- *eth2_0* and *eth2_1* are in the *vrf1* VRF instance.

For Cross-VRF routing, we reserve a special network namespace called *xvrf*: it hosts a bridge which is used to enable communication between VRF instances. Special values for the *veth* endpoint MAC address in *vrfX* are used, embedding the VRF instances' identifiers: this is required by the fast path to help fast Cross-VRF routing. We also need to add private IP addresses on the *veth* endpoint in *vrfX*.

1. Create the *vrf1* VRF instance:

```
# vrfctl add 1 linux-fp-sync-vrf.sh
```

Note: `linux-fp-sync-vrf.sh` is used to initialize *Linux - Fast Path Synchronization*.

2. Create a network namespace which will host a bridge to perform Cross-VRF routing:

```
# ip netns add xvrf
# ip netns exec xvrf brctl addbr xvrf-bridge
# ip netns exec xvrf ip link set xvrf-bridge up
```

3. Create a tunnel (*veth*) from *vrf0* to *xvrf*:

```
# ip link add xvrf0 type veth peer name from-vrf0 netns xvrf
```

The last 2 bytes of the MAC address MUST match the VRF instance number:

```
# ip link set xvrf0 address 00:09:C0:00:00:00 up
# ip netns exec xvrf ip link set from-vrf0 up
```

We create a private address for Cross-VRF routing:

```
# ip address add 169.254.128.0/16 dev xvrf0
```

If VNB is loaded, we must destroy the *ng_ether* node:

```
# ngctl shutdown from-vrf0:
```

Link the *veth* tunnel to the *xvrf* bridge:

```
# ip netns exec xvrf brctl addif xvrf-bridge from-vrf0
```

4. Create a tunnel (*veth*) from *vrf1* to *xvrf*:

```
# ip netns exec vrf1 ip link add xvrf1 type veth peer name from-vrf1 netns xvrf
# ip netns exec vrf1 ip link set xvrf1 address 00:09:C0:00:00:01 up
# ip netns exec xvrf ip link set from-vrf1 up
# ip netns exec vrf1 ip address add 169.254.128.1/16 dev xvrf1
```

If VNB is loaded, we must destroy the *ng_ether* node (VNB is *netns* agnostic, no need to execute *ip netns exec*).

```
# ngctl shutdown from-vrf1:
```

Link the *veth* tunnel to the *xvrf* bridge:

```
# ip netns exec xvrf brctl addif xvrf-bridge from-vrf1
```

5. Check the bridge in netns xvrf:

```
# ip netns exec xvrf brctl show
# bridge name     bridge id               STP enabled     interfaces
# xvrf-bridge     8000.8e4459c37102       no              from-vrf0
                                                          from-vrf1
```

6. Configure interfaces:

   a. Set *eth2_0* and *eth2_1* in the *vrf1* VRF instance:

```
# ip link set eth2_0 netns vrf1
# ip link set eth2_1 netns vrf1
```

   b. Activate interfaces:

```
# ip netns exec vrf1 ip link set eth2_0 up
# ip netns exec vrf1 ip link set eth2_1 up
# ip link set eth9_0 up
# ip link set eth9_1 up
```

   c. Add addresses:

```
# ip netns exec vrf1 ip address add 2.0.0.1/24 dev eth2_0
# ip netns exec vrf1 ip address add 2.1.0.1/24 dev eth2_1
# ip address add 9.0.0.1/24 dev eth9_0
# ip address add 9.1.0.1/24 dev eth9_1
```

   d. Add a bidirectional route between 100.2.2.1 and 110.2.2.1 (*eth2_0* <-> *eth2_1*):

```
# netns exec vrf1 ip route add 100.2.2.1/32 via 2.0.0.5
# netns exec vrf1 ip route add 110.2.2.1/32 via 2.1.0.5
```

   e. Add a bidirectional route between 100.9.9.1 and 110.9.9.1 (eth9_0 <-> *eth9_1*):

```
# route add 100.9.9.1/32 via 9.0.0.5
# route add 110.9.9.1/32 via 9.1.0.5
```

   f. Add a route from 100.2.9.1 to 110.2.9.1 (*eth2_0* -> *xvrf1* -> *eth9_1*):

```
# netns exec vrf1 ip route add 110.2.9.1/32 via 169.254.128.0
# route add 110.2.9.1/32 via 9.1.0.5
```

   g. Add a route from 110.2.9.1 to 100.2.9.1 (*eth2_1* -> *xvrf1* -> eth9_0):

```
# netns exec vrf1 ip route add 100.2.9.1/32 via 169.254.128.0
# route add 100.2.9.1/32 via 9.0.0.5
```

   h. Add a route from 100.9.2.1 to 110.9.2.1 (eth9_0 -> xvrf0 -> *eth2_1*):

```
# route add 110.9.2.1/32 via 169.254.128.1
# netns exec vrf1 ip route add 110.9.2.1/32 via 2.1.0.5
```

   i. Add a route from 110.9.2.1 to 100.9.2.1 (*eth9_1* -> xvrf0 -> *eth2_0*):

```
# route add 100.9.2.1/32 via 169.254.128.1
# netns exec vrf1 ip route add 100.9.2.1/32 via 2.0.0.5
```

7. Check the route in the *vrf0* VRF instance:

```
# ip route
9.0.0.0/24 dev eth9_0  proto kernel  scope link  src 9.0.0.1
9.1.0.0/24 dev eth9_1  proto kernel  scope link  src 9.1.0.1
100.2.9.1 via 9.0.0.5 dev eth9_0
100.9.2.1 via 169.254.128.1 dev xvrf0
100.9.9.1 via 9.0.0.5 dev eth9_0
110.2.9.1 via 9.1.0.5 dev eth9_1
110.9.2.1 via 169.254.128.1 dev xvrf0
110.9.9.1 via 9.1.0.5 dev eth9_1
169.254.0.0/16 dev xvrf0  proto kernel  scope link  src 169.254.128.0
```

8. Check route in the *vrf1* VRF instance:

```
# ip netns exec vrf1 ip route
2.0.0.0/24 dev eth2_0  proto kernel  scope link  src 2.0.0.1
2.1.0.0/24 dev eth2_1  proto kernel  scope link  src 2.1.0.1
100.2.2.1 via 2.0.0.5 dev eth2_0
100.2.9.1 via 169.254.128.0 dev xvrf1
100.9.2.1 via 2.0.0.5 dev eth2_0
110.2.2.1 via 2.1.0.5 dev eth2_1
110.2.9.1 via 169.254.128.0 dev xvrf1
110.9.2.1 via 2.1.0.5 dev eth2_1
169.254.0.0/16 dev xvrf1  proto kernel  scope link  src 169.254.128.1
```

9. Check the fast path configuration:

```
<fp-0> iface
37:eth2_1 [VR-1] ifuid=0x25d8c282 (port 1) <UP|RUNNING|FWD4|FWD6> (0x63)
        type=ether mac=00:02:02:00:00:21 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=3  IPv6 routes=1
59:lo [VR-1] ifuid=0x3b1c7058 (virtual) <UP|RUNNING|FWD4|FWD6> (0x63)
        type=loop mac=00:00:00:00:00:00 mtu=16436 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
100:fpn0 [VR-0] ifuid=0x64247322 (virtual) <UP|RUNNING|FWD4|FWD6> (0x63)
        type=ether mac=00:00:46:50:4e:00 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
117:lo [VR-0] ifuid=0x754c6fa8 (virtual) <UP|RUNNING|FWD4|FWD6> (0x63)
        type=loop mac=00:00:00:00:00:00 mtu=16436 tcp4mss=0 tcp6mss=0
        IPv4 routes=0  IPv6 routes=0
141:eth9_0 [VR-0] ifuid=0x8d001382 (port 2) <UP|RUNNING|FWD4|FWD6> (0x63)
```

(continues on next page)

(continued from previous page)

```
        type=ether mac=00:02:02:00:00:90 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=3  IPv6 routes=1
440:xvrf1 [VR-1] ifuid=0xb8d9e470 (virtual) <UP|RUNNING|FWD4|FWD6|LOCAL_OUT>␣
↪(0x8063)
        type=xvrf mac=00:09:c0:00:00:01 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=3  IPv6 routes=0
504:eth2_0 [VR-1] ifuid=0xf8e170d2 (port 0) <UP|RUNNING|FWD4|FWD6> (0x63)
        type=ether mac=00:02:02:00:00:20 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=3  IPv6 routes=1
710:xvrf0 [VR-0] ifuid=0xc6129210 (virtual) <UP|RUNNING|FWD4|FWD6|LOCAL_OUT>␣
↪(0x8063)
        type=xvrf mac=00:09:c0:00:00:00 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=3  IPv6 routes=0
953:eth9_1 [VR-0] ifuid=0xb9f76532 (port 3) <UP|RUNNING|FWD4|FWD6> (0x63)
        type=ether mac=00:02:02:00:00:91 mtu=1500 tcp4mss=0 tcp6mss=0
        IPv4 routes=3  IPv6 routes=1
```

```
<fp-0> route4
# - Preferred, * - Active, > - selected
100.2.9.1/32  [27]  ROUTE gw 9.0.0.5 via eth9_0(0x8d001382) (39)
100.9.2.1/32  [30]  ROUTE gw 169.254.128.1 via xvrf0(0xc6129210) (42)
100.9.9.1/32  [27]  ROUTE gw 9.0.0.5 via eth9_0(0x8d001382) (34)
110.2.9.1/32  [28]  ROUTE gw 9.1.0.5 via eth9_1(0xb9f76532) (37)
110.9.2.1/32  [30]  ROUTE gw 169.254.128.1 via xvrf0(0xc6129210) (40)
110.9.9.1/32  [28]  ROUTE gw 9.1.0.5 via eth9_1(0xb9f76532) (35)
```

New reference for the VRF instance: 1.

```
<fp-0> vrf-set 1
```

```
<fp-1> route4
# - Preferred, * - Active, > - selected
100.2.2.1/32  [25]  ROUTE gw 2.0.0.5 via eth2_0(0xf8e170d2) (32)
100.2.9.1/32  [29]  ROUTE gw 169.254.128.0 via xvrf1(0xb8d9e470) (38)
100.9.2.1/32  [25]  ROUTE gw 2.0.0.5 via eth2_0(0xf8e170d2) (43)
110.2.2.1/32  [26]  ROUTE gw 2.1.0.5 via eth2_1(0x25d8c282) (33)
110.2.9.1/32  [29]  ROUTE gw 169.254.128.0 via xvrf1(0xb8d9e470) (36)
110.9.2.1/32  [26]  ROUTE gw 2.1.0.5 via eth2_1(0x25d8c282) (41)
```


vrfd

Description

This daemon monitors iproute2 netns creation and deletion and automatically binds or deletes corresponding VRF instances.

Synopsis

```
vrfd [-Fvh] [-p iproute2-netns-prefix] [-s vrf-init-script]
```

Parameters

-F
Optional. Foreground mode.

-v
Optional. Copy logs to standard output.

-h
Optional. Display help.

-p <iproute2-netns-prefix>
Optional. Prefix of the *iproute2 netns* instance to monitor. This option can be set up to 64 times. *vrfd* matches all *iproute2 netns* instances except *vrf0* and *xvrf*.

-s <vrf-init-script>
Optional. Custom script to execute when the VRF instance is bound.

Note: By default, the script `linux-fp-sync-vrf.sh` must be used to initialize *Linux - Fast Path Synchronization*. You can adapt it to your needs.

Userspace applications API usage examples

Opening a socket in the *VRF3* VRF instance

```
if (libvrf_init() < 0) {
  /* error path */
  return -1;
}
```

(continues on next page)

(continued from previous page)

```
if (libvrf_change(3) < 0) {
  /* error path */
  return -1;
}

/* now we are in netns vrf3 */

/* open /proc from vrf3 */
proc = open("/proc/sys/net/...", ..);

/* open a socket in vrf 3 */
socket = socket(x, y, z);

if (libvrf_back() < 0) {
  /* error path */
  return -1;
}

/* We are back to the initial network namespace, but proc and socket are still in␣
↪network namespace
 * vrf3. Don't forget to close them when you are over; otherwise, it will not be
 * possible to remove this network namespace.
 */
```

Move an interface from the *VRF2* to the *VRF3* VRF instance

```
if (libvrf_init() < 0) {
      /* error path */
      return -1;
}

if (libvrf_change(2) < 0) {
      /* error path */
      return -1;
}

/* Now we are in network namespace vrf2 */

/* Open a netlink socket in vrf2 */
socket = socket(AF_NETLINK, SOCK_RAW, NETLINK_ROUTE);

if (libvrf_back() < 0) {
```

(continues on next page)

(continued from previous page)

```
      /* error path */
      return -1;
}

/* Now, we get a ref to the new network namespace, this ref will be used in the␣
↪netlink
 * message.
 */
vrf3 = libvrf_get_fd(3);
if (vrf3 < 0) {
      /* error path */
      return -1;
}

/* Build a netlink message for the netdevice and then add the
 * information of the new network namespace.
 */
addattr_l(&req->n, sizeof(*req), IFLA_NET_NS_FD, &vrf3, 4);

/* Now, this message should be sent into a netlink socket of vrf2 */
sendmsg(socket, &req, 0);

/* Don't forget to close the fd at the end */
close(vrf3);
```

2.7 Management

2.7.1 Management KPIs

Overview

Streaming telemetry plays a critical role in traffic optimization, monitoring and troubleshooting.

6WIND monitoring solution is split to ease the integration with existing monitoring system, and also offers a complete solution for new deployment:

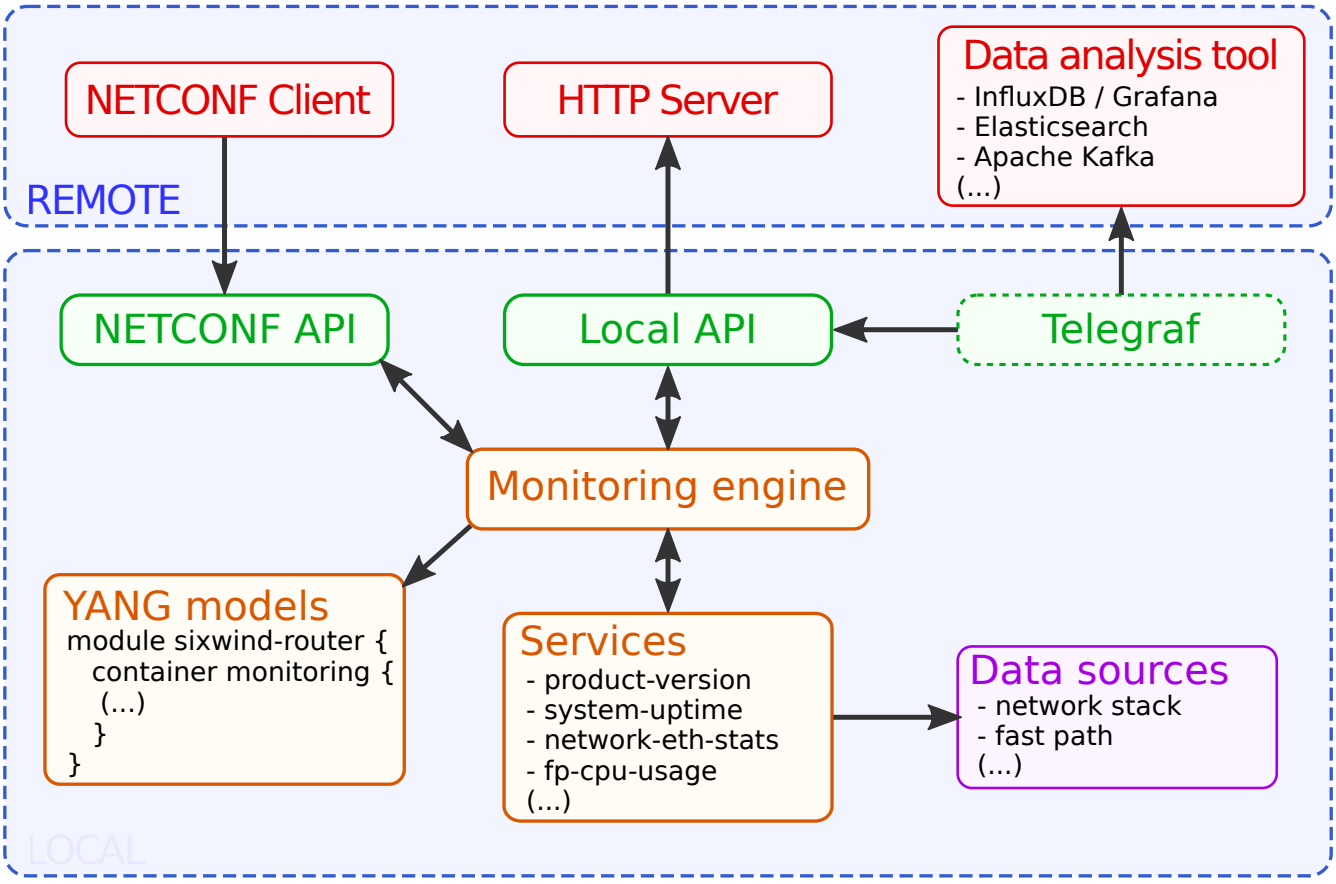
- The KPIs daemon, for both Linux system and fast path modules
- The monitoring engine, exposing the KPI (Key Performance Indicator) with local API and NETCONF/YANG API in a consistent way
- The tool to query the API for direct use by common existing tools (JSON, InfluxDB)

Features

- centralized API to query KPIs
- HTTP POST or local output
- Json and Influx format support
- NETCONF/Yang
- KPIs addition on a running system

Architecture

The 6WIND monitoring solution is organized as follows:



YANG models define the data that is available in the monitoring engine. The monitoring engine interacts with multiple services. Those services use data sources to provide monitoring data when requested by the monitoring engine.

The monitoring engine is exposed in two ways:

- A NETCONF API
- a local API

To support more monitoring solutions, the telegraf collector can make use of the local API to populate data analysis tools.

In term of software:

- the monitoring engine is provided by the `sysrepo` datastore
- the netconf API is provided by the `netopeer2-server` daemon
- the monitoring services are provided by the `kpid` daemon
- the local API can be queried by the `kpi-get` tool

Quickstart

A `monitoring.sh` script is available to start the monitoring module.

The start, status and stop commands are available:

```
# monitoring.sh start
Starting Monitoring...
Monitoring successfully started

# monitoring.sh status
* kpid.service - KPI Daemon
   Loaded: loaded (/lib/systemd/system/kpid.service; disabled; vendor preset: enabled)
   Active: active (running) since Tue 2020-06-16 09:31:53 UTC; 3s ago
 Main PID: 3526 (kpid)
    Tasks: 1 (limit: 1088)
   CGroup: /system.slice/kpid.service
           `-3526 /usr/bin/python3 /usr/bin/kpid

* netopeer2-server.service - NETCONF Server
   Loaded: loaded (/lib/systemd/system/netopeer2-server.service; enabled; vendor␣
↪preset: enabled)
   Active: active (running) since Mon 2020-06-16 09:31:52 UTC; 4s ago
 Main PID: 1067 (netopeer2-serve)
    Tasks: 7 (limit: 1088)
   CGroup: /system.slice/netopeer2-server.service
           `-1067 /usr/bin/netopeer2-server -U -g netconf -m 660

# monitoring.sh stop
Stopping Monitoring...
Monitoring successfully stopped
```

Netopeer options can be configured in `/etc/default/netopeer` and kpid options can be configured in `/etc/default/kpid`.

On the monitored device, fetch the fp-cpu-usage kpi:

```
# kpi-get fp-cpu-usage -o json
{
  "sixwind-router:monitoring": {
    "fp:cpu-usage": [
      {
        "cpu": "cpu1",
        "busy": 0
      },
      {
        "cpu": "cpu2",
        "busy": 0
      },
      {
        "cpu": "cpu3",
        "busy": 0
      }
    ]
  }
}
```

KPI definition

YANG

6WIND monitoring uses YANG to describe its KPIs. YANG is an IETF (Internet Engineering Task Force) data modeling language (RFC 6020 (https://tools.ietf.org/html/rfc6020)). This section will briefly introduce this format.

Here is an example of YANG model for 6WIND monitoring sixwind-router module and product group.

```
module sixwind-router {
  namespace "urn:6wind:router";
  prefix router;

  organization "6WIND";
  description
    "6WIND router data model";
  contact
    "support@6wind.com";

  revision "2017-12-04" {
```

(continues on next page)

(continued from previous page)

```
    description
      "Initial revision.";
  }

  container monitoring {
     config false;
     description
       "6WIND monitoring data model.";
  }
}
```

```
module product {
  namespace "urn:6wind:router:monitoring:product";
  prefix product;

  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This module provides support for showing product information.";

  revision "2017-11-14" {
    description
      "Initial revision.";
  }

  augment /sixwind-router:monitoring {

    leaf version {
      type string;
      description
        "The product version.";
    }

    container license {
      description
        "The license detailed info.";

      leaf enabled {
        type boolean;
        description
          "True if the license daemon is enabled on the system.";
```

(continues on next page)

(continued from previous page)

```
      }

      leaf valid {
        type boolean;
        description
          "True if the license is valid.";
      }

      leaf short-license-type {
        type enumeration {
          enum evaluation {
            description
              "The license is an evaluation.";
          }
          enum perpetual {
            description
              "The license is perpetual.";
          }
          enum subscription {
            description
              "The license is a subscription.";
          }
        }
        description
          "A shorter version of the license type.";
      }

      leaf support-end-date {
        type string;
        description
          "The support end date.";
      }

      leaf remaining-days {
        type union {
          type string;
          type enumeration {
            enum unset {
              description
                "The end date is not set.";
            }
            enum expired {
              description
                "The date has expired.";
```

(continues on next page)

(continued from previous page)

```
        }
      }
    }
    description
      "The number of days remaining.";
  }

  leaf throughput-allowed {
    description
      "The allowed throughput.";
    type decimal64 {
      fraction-digits 2;
    }
    units "Gb";
  }

  leaf throughput-used {
    description
      "The throughput currently in use.";
    type decimal64 {
      fraction-digits 2;
    }
    units "Gb";
  }

  leaf cgnat-conntracks-allowed {
    type uint32;
    description
      "The number of CG-NAT conntracks allowed.";
  }

  leaf cgnat-conntracks-used {
    type uint32;
    description
      "The number of CG-NAT conntracks currently in use.";
  }

  leaf ipsec-tunnels-allowed {
    type uint32;
    description
      "The number of IPsec tunnels allowed.";
  }

  leaf ipsec-tunnels-used {
```

(continues on next page)

(continued from previous page)

```
        type uint32;
        description
          "The number of IPsec tunnels currently in use.";
      }
    }
  }
}
```

Let's explain some of the YANG keywords:

- `module` is a self-contained top-level hierarchy of the following nodes.
- `leaf` is where values are put. Each leaf has a `type` (e.g: string, uint16_t, uint64_t, enumeration) and a `description`.
- `augment <path>` means that the content of the augment node should be inserted inside <path>.
- `list` defines a list, the `key` keyword defines its key.
- `container` is a way of grouping elements together.
- `description` is the current element description.
- `revision` is the version of the YANG model.
- `prefix` tells how the module elements should be prefixed.

KPI structure

The KPI list is broken into 4 groups:

- product, including version, or license state
- system, including load, uptime
- network, including interface statistics
- fp, for statistics about fast path

In each group, several services are available. A service is a group of statistics of the same kind.

For instance, the product group contains:

- product-version
- product-license

The fp group contains:

- fp-ip-stats
- fp-ip6-stats

- fp-cpu-usage
- ...

The list of services and their parameters is available using the `kpi-list-services` command on the device.

KPI identification

To identify a KPI in 6WIND monitoring module, two ways can be used.

The first one is to use a service name, as defined in the previous section:

service	data accessed
fp-ip-stats	the fast path IPv4 statistics
fp-cpu-usage	the fast path cpu usage
product-version	the version of the product

The second is to use a path, similar to xpath (XML path language). Paths can be deduced from the YANG model. Some examples follow:

xpath	data accessed
/sixwind-router:monitoring	everything
/sixwind-router:monitoring/fp:statistics	all the fast path statistics
/sixwind-router:monitoring/fp:statistics/fp:ip	the fast path IPv4 statistics
/sixwind-router:monitoring/system:cpu-usage[cpu='cpu3']	the system cpu load for cpu3

Depending on the product and the module, some statistics might not be available.

Fetching KPIs

This section explains how the data can be fetched from the device.

kpi-get

The kpi-get embedded tool can be used to query the monitoring data, using the local API or NETCONF API, and output in json or influx format. It can also make HTTP POST request to a remote server.

The tool -h option explains how it works.

```
# kpi-get -h
usage: kpi-get [-h] [-i FMT[:opts]] -o FMT[:opts] DATA

Tool to dump KPIs
```

(continues on next page)

(continued from previous page)

```
positional arguments:
  DATA                  either an xpath or a service. To get everything, use
                        /sixwind-router:monitoring

optional arguments:
  -h, --help            show this help message and exit
  -i FMT[:opts], --input-format FMT[:opts]
                        use "FMT:help" to get details about a format (default:
                        sysrepo)
  -o FMT[:opts], --output-format FMT[:opts]
                        use "FMT:help" to get details about a format

Available services:
    fp-context-switch-stats, fp-cpu-usage, fp-drop-stats, fp-exception-queue-stats,
    fp-exceptions-stats, fp-ip-stats, fp-ip6-stats, fp-ipsec-stats, fp-
    ipsec6-stats, fp-l2-stats, network-nic-eth-stats, network-nic-traffic-stats,
    product-license, product-version, system-cpu-usage, system-numa-stats,
    system-soft-interrupts-stats, system-uptime
Available input formats:
    netconf, sysrepo
Available output formats:
    http-influx, http-json, influx, json, raw
```

A few examples follow to explain how the tool works.

Get all the available data in json format:

```
# kpi-get /sixwind-router:monitoring -o json
{
  "sixwind-router:monitoring": {
    "system:numa-stats": [
      {
        "node": "node0",
        "other-node": 0,
        "numa-miss": 0,
        "numa-foreign": 0,
        "interleave-hit": 13969,
        "local-node": 3030760,
        "numa-hit": 3030760
      }
    ],
    "product:uptime": "5:12:15",
    (...)
}
```

Get all the fast path IP statistics in influx format:

```
# kpi-get fp-ip-stats -o influx
fp-ip-stats,path=/sixwind-router:monitoring/fp:statistics/fp:ip,host=dut-vm␣
↪IpDroppedBlackhole=0,IpDroppedForwarding=0,IpDroppedIPsec=0,
↪IpDroppedInvalidInterface=0,IpDroppedNetfilter=0,IpDroppedNoArp=0,
↪IpDroppedNoMemory=0,IpDroppedRouteException=0,IpForwDatagrams=0,IpFragCreates=0,
↪IpFragFails=0,IpFragOKs=0,IpInAddrErrors=0,IpInDelivers=0,IpInHdrErrors=0,
↪IpInReceives=0,IpReasmExceptions=0,IpReasmFails=0,IpReasmOKs=0,IpReasmReqds=0,
↪IpReasmTimeout=0
```

Post the fast path cpu usage in json format to a http://a.b.c.d:8000/write:

```
# kpi-get fp-cpu-usage -o http-json:host=a.b.c.d:port=8000:path=write
```

Telegraf

Telegraf (https://github.com/influxdata/telegraf) is an agent that collects and reports metrics. Coupled with 6WIND's tool `kpi-get`, it can report all the available statistics to a remote location.

Telegraf can report the monitoring data to multiple tools. The list of supported output plugins is maintained here (https://github.com/influxdata/telegraf/tree/1.4.4#output-plugins).

To run telegraf, at least an input plugin and an output plugin should be defined. The configuration is located in `/etc/telegraf/telegraf.conf` and `/etc/telegraf/telegraf.d/`.

In `/etc/telegraf/telegraf.conf`, you can configure the agent behavior. Here is an example:

```
[agent]
  debug = false
  flush_buffer_when_full = true
  flush_interval = "15s"
  flush_jitter = "0s"
  hostname = "myhostname"
  interval = "15s"
  round_interval = true
```

In the `/etc/telegraf/telegraf.d/` directory, you can add the plugins. Here is an example of making use of the input exec plugin, that will query into 6WIND monitoring using its local API, and output to influx format:

```
[[inputs.exec]]
   commands = [ "kpi-get /sixwind-router:monitoring -o influx"]
```

Here is an example of influxdb output plugin:

```
[[outputs.influxdb]]
  database = "telegraf"
  urls = [ "http://a.b.c.d:8086" ]
  username = "telegraf"
  password = "mypassword"
```

Note: We encourage the use of `/etc/telegraf/telegraf.d/` for plugins, but the configuration for those plugins can be put in `/etc/telegraf/telegraf.conf`. The plugins are self-documented in this file.

Configuration samples for input and output can be found in the `/usr/share/6WIND/telegraf` directory.

Once configured, telegraf is controlled by a service file.

It is disabled by default, because some configuration is needed to make it work (the output plugin). To enable telegraf on boot and start it:

```
# systemctl enable telegraf
# systemctl start telegraf
```

To disable it from boot and stop it:

```
# systemctl disable telegraf
# systemctl stop telegraf
```

NETCONF

NETCONF is an IETF standard used for configuration. It can also monitor the state of a system.

Any tool supporting NETCONF can get data from the device by connecting to port `830`. Any user configured on the machine can access the monitoring data. YANG models are available at the end of this document, and in the `/usr/share/6WIND/yang` directory on the device.

Use case: HTTP POST to remote server

In this section, we will explain how to send KPIs to a remote HTTP server using POST requests.

For the purpose of this use case, let's launch a python HTTP server that is able to process POST request on a linux host with IP address `a.b.c.d`.

1. Create a file named `dummy-server.py` with this content:

```
#!/usr/bin/env python
from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer
```

(continues on next page)

(continued from previous page)

```
class HTTPPostHandler(BaseHTTPRequestHandler):
    def do_POST(self):
        content_length = int(self.headers['Content-Length'])
        post_data = self.rfile.read(content_length)
        print(post_data)

        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()
        self.wfile.write("<html><body><h1>POST!</h1></body></html>")

def run():
    httpd = HTTPServer(('', 8000), HTTPPostHandler)
    print('Starting httpd...')
    httpd.serve_forever()

if __name__ == "__main__":
    run()
```

2. Start the server:

```
# python dummy-server.py
```

3. On a freshly booted system, start monitoring:

```
# monitoring.sh start
Starting Monitoring...
Monitoring successfully started
```

4. Check that the monitoring has been properly started

```
# monitoring.sh status
* kpid.service - KPI Daemon
   Loaded: loaded (/lib/systemd/system/kpid.service; disabled; vendor preset:␣
↪enabled)
   Active: active (running) since Tue 2020-06-16 09:31:53 UTC; 3s ago
 Main PID: 3526 (kpid)
    Tasks: 1 (limit: 1088)
   CGroup: /system.slice/kpid.service
           `-3526 /usr/bin/python3 /usr/bin/kpid

* netopeer2-server.service - NETCONF Server
   Loaded: loaded (/lib/systemd/system/netopeer2-server.service; enabled; vendor␣
↪preset: enabled)
```

(continues on next page)

(continued from previous page)

```
   Active: active (running) since Mon 2020-06-16 09:31:52 UTC; 4s ago
 Main PID: 1067 (netopeer2-serve)
    Tasks: 7 (limit: 1088)
   CGroup: /system.slice/netopeer2-server.service
           `-1067 /usr/bin/netopeer2-server -U -g netconf -m 660
```

5. Do an HTTP post of the fast path cpu usage in json format every 10 seconds to http://a.b.c.d:8000/write. Change `a.b.c.d` to match the remote host's address:

```
# while true; do
    kpi-get fp-cpu-usage -o http-json:host=a.b.c.d:port=8000:path=write
    sleep 10
  done
```

On the web server, you should see POST requests every 10 seconds or so:

```
{"sixwind-router:monitoring": {"fp:fp-cpu-usage": [{"cpu": "cpu1", "busy": 0}, {"cpu":
↪"cpu2", "busy": 0}, {"cpu": "cpu3", "busy": 0}]}}
127.0.0.1 - - [13/Dec/2017 16:04:37] "POST /write HTTP/1.1" 200 -
{"sixwind-router:monitoring": {"fp:fp-cpu-usage": [{"cpu": "cpu1", "busy": 0}, {"cpu":
↪"cpu2", "busy": 0}, {"cpu": "cpu3", "busy": 0}]}}
127.0.0.1 - - [13/Dec/2017 16:04:48] "POST /write HTTP/1.1" 200 -
{"sixwind-router:monitoring": {"fp:fp-cpu-usage": [{"cpu": "cpu1", "busy": 0}, {"cpu":
↪"cpu2", "busy": 0}, {"cpu": "cpu3", "busy": 0}]}}
127.0.0.1 - - [13/Dec/2017 16:04:59] "POST /write HTTP/1.1" 200 -
{"sixwind-router:monitoring": {"fp:fp-cpu-usage": [{"cpu": "cpu1", "busy": 0}, {"cpu":
↪"cpu2", "busy": 0}, {"cpu": "cpu3", "busy": 0}]}}
127.0.0.1 - - [13/Dec/2017 16:05:09] "POST /write HTTP/1.1" 200 -
```

Use case: integration with InfluxDB / Grafana

In this section, we will explain how to integrate 6WIND monitoring module with InfluxDB (https://docs.influxdata.com/influxdb) and Grafana (http://docs.grafana.org/) using the Telegraf (https://github.com/influxdata/telegraf) collector. This section assumes that InfluxDB and Grafana have been installed on a `A.B.C.D` host, and that InfluxDB is listening on its default `8086` port.

See also:

Grafana installation (http://docs.grafana.org/installation/) and InfluxDB installation (https://docs.influxdata.com/influxdb/v1.3/introduction/installation/).

1. On a freshly booted system, start monitoring

```
# monitoring.sh start
Starting Monitoring...
Monitoring successfully started
```

2. Check that the monitoring has been properly started

```
# monitoring.sh status
* kpid.service - KPI Daemon
   Loaded: loaded (/lib/systemd/system/kpid.service; disabled; vendor preset:␣
↪enabled)
   Active: active (running) since Tue 2020-06-16 09:31:53 UTC; 3s ago
 Main PID: 3526 (kpid)
    Tasks: 1 (limit: 1088)
   CGroup: /system.slice/kpid.service
           `-3526 /usr/bin/python3 /usr/bin/kpid

* netopeer2-server.service - NETCONF Server
   Loaded: loaded (/lib/systemd/system/netopeer2-server.service; enabled; vendor␣
↪preset: enabled)
   Active: active (running) since Mon 2020-06-16 09:31:52 UTC; 4s ago
 Main PID: 1067 (netopeer2-serve)
    Tasks: 7 (limit: 1088)
   CGroup: /system.slice/netopeer2-server.service
           `-1067 /usr/bin/netopeer2-server -U -g netconf -m 660
```

3. Copy the telegraf configuration file examples in telegraf configuration directory

```
# cp /usr/share/6WIND/telegraf/telegraf-input-kpi.conf /etc/telegraf/telegraf.d
# cp /usr/share/6WIND/telegraf/telegraf-output-influx.conf /etc/telegraf/telegraf.
↪d
```

4. Edit the `/etc/telegraf/telegraf.d/telegraf-output-influx.conf` file to put the A.B.C.D address, and the database information. If the database is not created on the InfluxDB server, it will be created by telegraf.

```
[[outputs.influxdb]]
  urls = ["http://A.B.D.C:8086"] # Put the InfluxDB host address here
  database = "telegraf" # Put the InfluxDB database name here
  # username = "user" # Optional, put a user to connect to the InfluxDB database
  # password = "password" # Optional, put a password to connect to the InfluxDB␣
↪database
```

5. Start and enable the telegraf service on boot:

```
# systemctl start telegraf
# systemctl enable telegraf
```

At this point, the InfluxDB `telegraf` database should start to be populated with statistics coming from the 6WIND monitoring module.

Using the InfluxDB client (installed influxdb-client package), we can check that some records have been received.

```
# influx -host A.B.C.D -port 8086 -database 'telegraf'
Visit https://enterprise.influxdata.com to register for updates, InfluxDB server␣
↪management, and monitoring.
Connected to http://127.0.0.1:8086 version 1.3.5
InfluxDB shell version: 1.0.2
> show measurements
name: measurements
------------------
fp-context-switch-stats
fp-cpu-usage
fp-drop-stats
fp-exception-queue-stats
fp-exceptions-stats
fp-ip-stats
fp-ip6-stats
fp-ipsec-stats
fp-ipsec6-stats
fp-l2-stats
linux-nic-eth-stats
linux-nic-traffic-stats
system-cpu-usage
system-numa-stats
system-memory
system-uptime
product-license
product-version

> SELECT * FROM linux-nic-traffic-stats LIMIT 4
name: linux-nic-traffic-stats
----------------------
time                    bytes-recv      bytes-sent      host    name    pkts-recv    ␣
↪   pkts-sent
1510672561000000000     127617994       1495014321      router  mgmt0   1529466      ␣
↪   1488143
1510672561000000000     41454147        41454147        router  lo      173106       ␣
↪   173106
1510672571000000000     127619161       1495027098      router  mgmt0   1529483      ␣
↪   1488159
1510672571000000000     41454147        41454147        router  lo      173106       ␣
↪   173106
```

(continues on next page)

(continued from previous page)

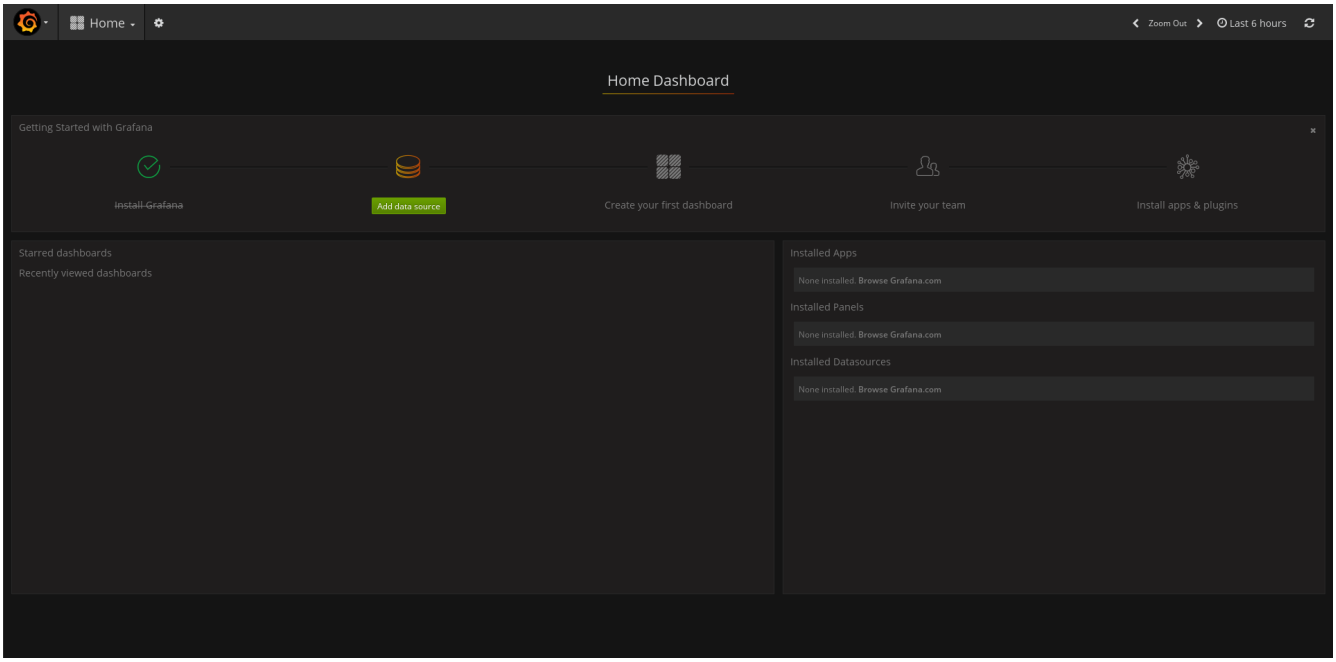
```
> SELECT IpForwDatagrams,IpInDelivers,IpInReceives FROM fp-ip-stats LIMIT 4
name: fp-ip-stats
----------------------
time             IpForwDatagrams    IpInDelivers    IpInReceives
1510748131000000000    0        0        0
1510748141000000000    0        0        0
1510748151000000000    0        0        0
1510748161000000000    0        0        0
```

Then, dashboards can be built by adding the InfluxDB database as data source and using the Query Editor to select metrics and tags in Grafana.

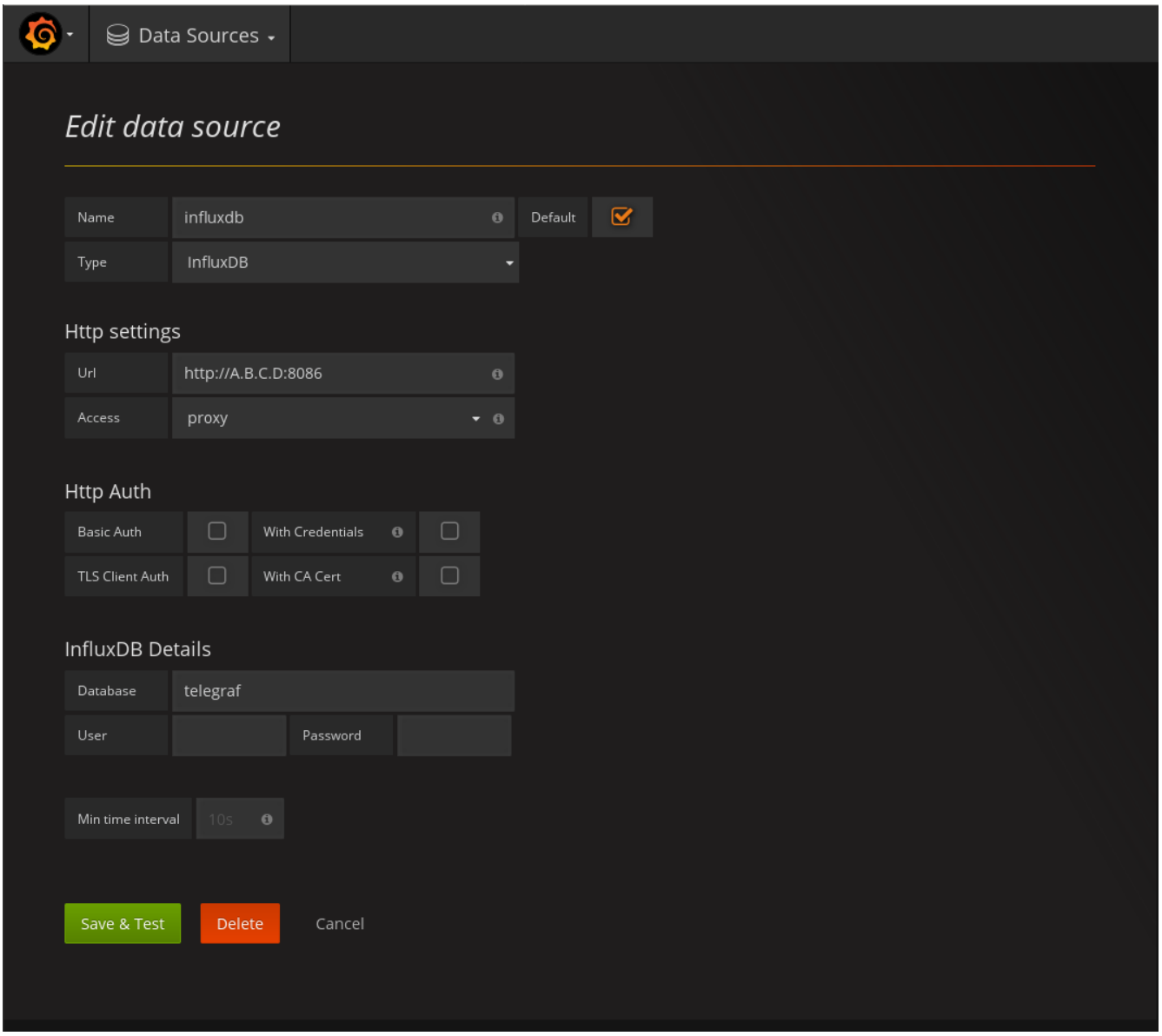
See also:

Using InfluxDB in Grafana (http://docs.grafana.org/features/datasources/influxdb/)

To do so, first login to grafana. The url is `http://A.B.C.D:3000/`, the default user / password are `admin` / `admin`. Then, you should see a screen asking to add a data source. Let's add an InfluxDB data source. On the welcome screen, click on `Add data source`:

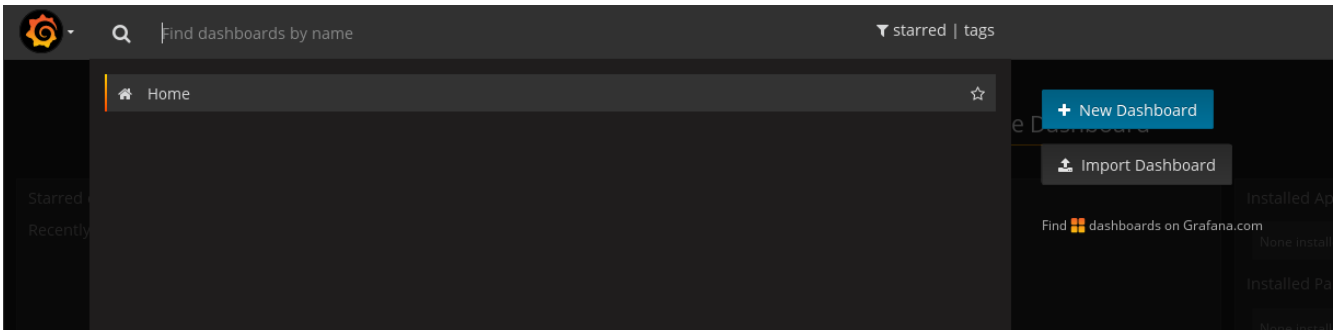


A new screen appears, configure a name for the data source, set InfluxDB as type, configure the url to point to your InfluxDB instance, and the database, so that it looks like this:

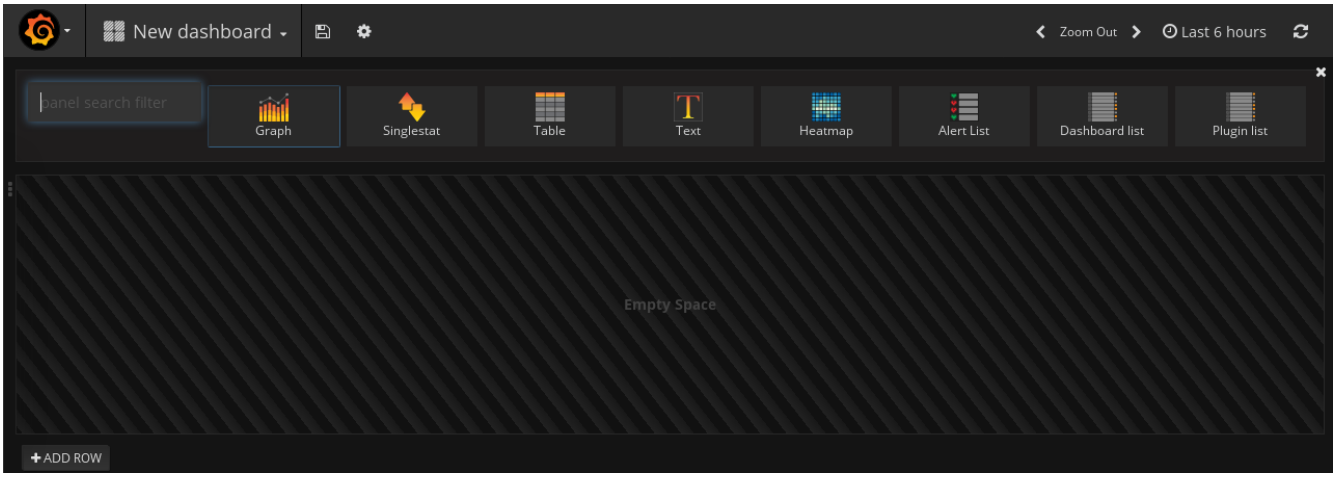


Once the data source is properly configured, this message should be displayed on the screen:

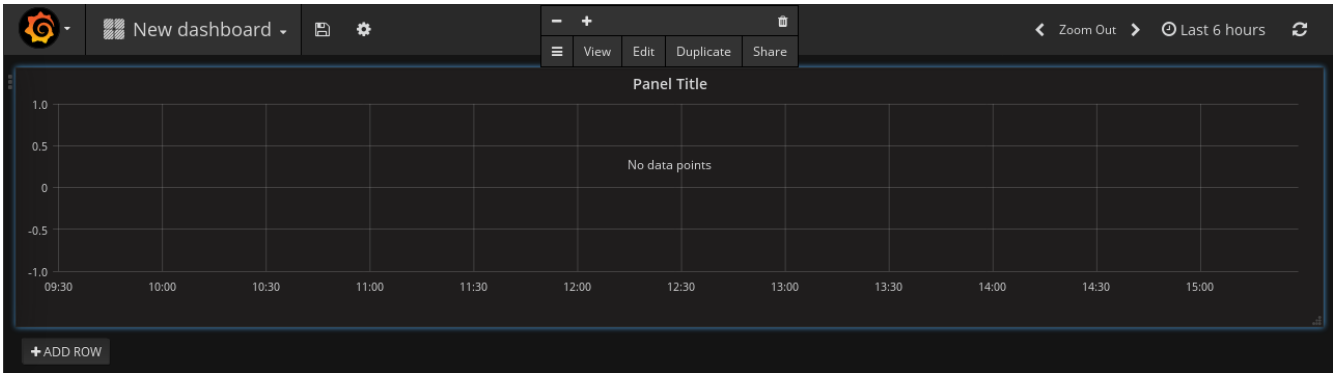
You are now ready to create your first dashboard. Let's add a graph of the system load. Go back to `http://A.B.C.D:3000/` and click on `Home` on the left corner of the page:



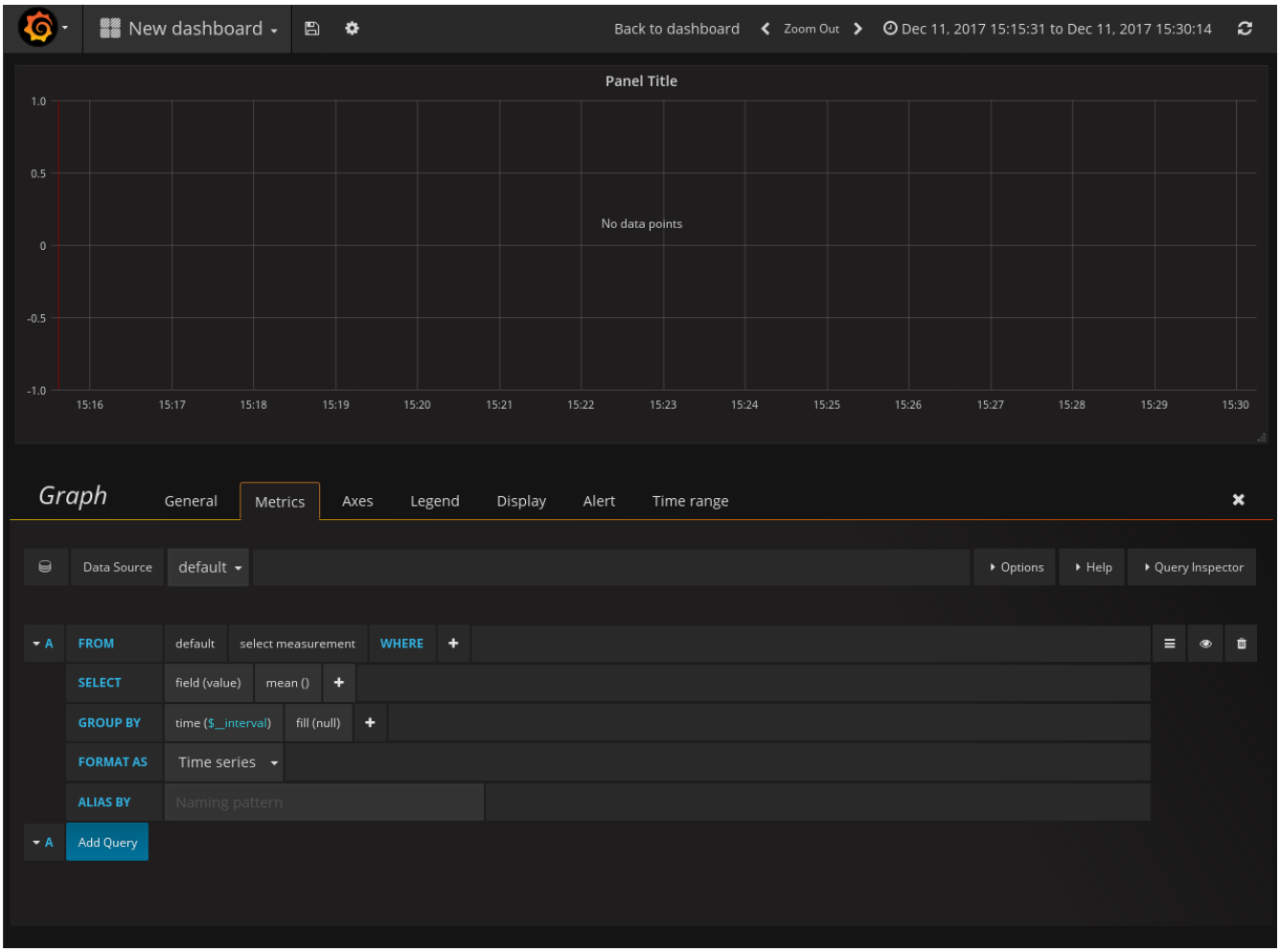
Click on `+ New Dashboard`:



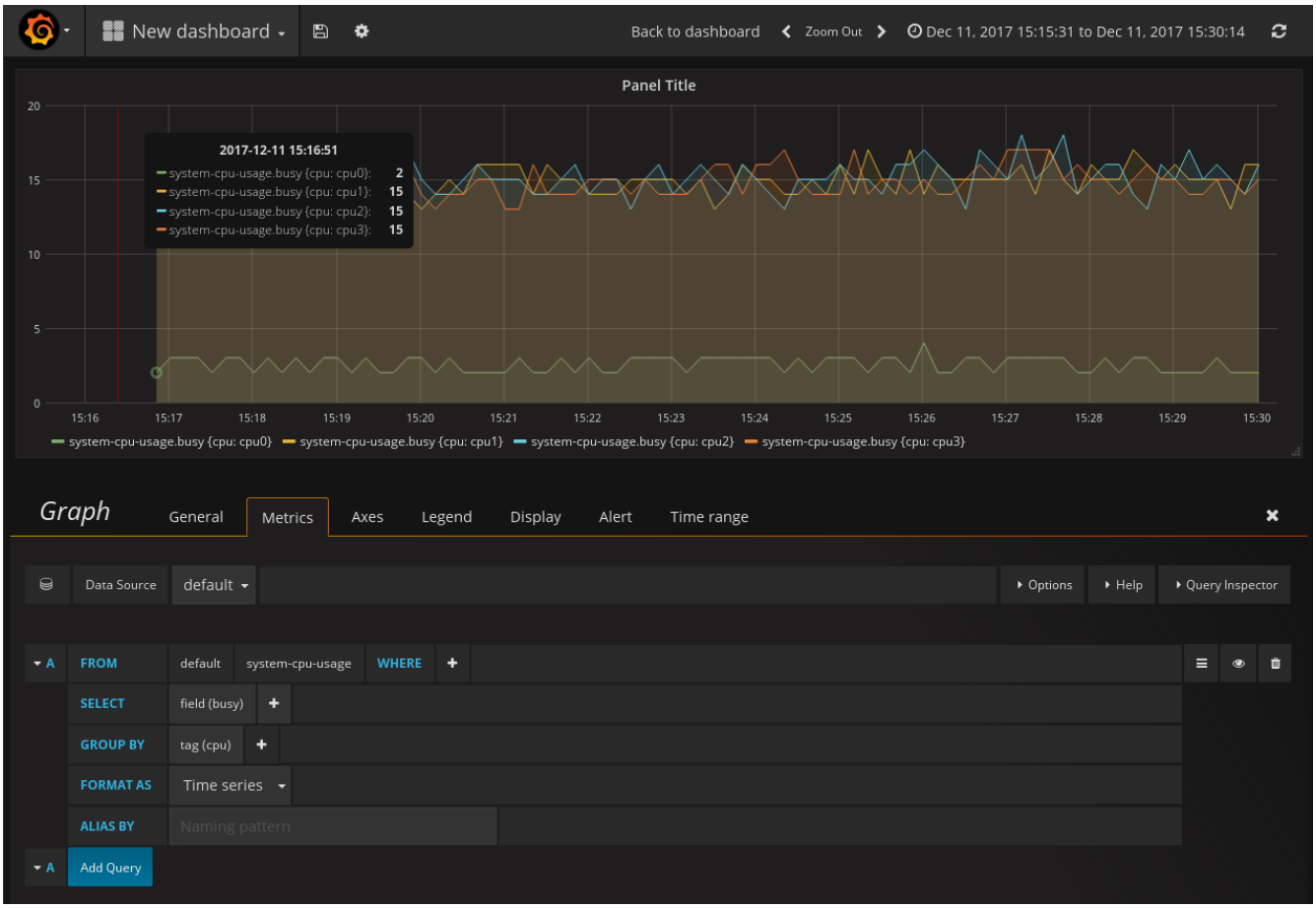
Click on `graph` to add a new one:



Click on `Panel Title`, a menu appears. Click on `Edit`:



Click on `select measurements` on the `From` line, the list of available services appears. Select `system-cpu-usage`. Click on `mean()` on the `Select` line and then `remove`. Click on `value` in the `field` box on the `Select` line and select `busy`. Click on `time` in the `Group By` line, and then `Remove`. Click on + on the `Group By` line and select `tag(cpu)`. The following graph should appear.



See also:

The Grafana documentation (http://docs.grafana.org/).

YANG models

This section shows the content of all the YANG files.

sixwind-router

```
module sixwind-router {
  namespace "urn:6wind:router";
  prefix router;

  organization "6WIND";
  description
    "6WIND router data model";
```

(continues on next page)

(continued from previous page)

```
  contact
    "support@6wind.com";

  revision "2017-12-04" {
    description
      "Initial revision.";
  }

  container monitoring {
     config false;
     description
       "6WIND monitoring data model.";
  }
}
```

fp

```
module fp {
  namespace "urn:6wind:router:monitoring:fp";
  prefix fp;

  include fp-cpu-usage;
  include fp-ctx-switch-stats;
  include fp-exception-queue-stats;
  include fp-filling;
  include fp-filling-cg-nat;
  include fp-cg-nat-stats;
  include fp-ports-stats;
  include fp-exceptions-stats;
  include fp-global-stats;
  include fp-gre-stats;
  include fp-ip-stats;
  include fp-ipsec-stats;
  include fp-l2-stats;
  include fp-vxlan-stats;

  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This module provides support for showing fp information.";
```

(continues on next page)

(continued from previous page)

```
  revision "2017-10-11" {
    description
      "Initial revision.";
  }

  augment /sixwind-router:monitoring {
    description
      "Fast path information.";
    container statistics {
    }
  }
}
```

fp-cpu-usage

```
submodule fp-cpu-usage {

  belongs-to fp {
    prefix fp;
  }
  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This submodule provides support for showing fp load information.";

  revision "2018-02-02" {
    description
      "Initial revision.";
  }

  augment /sixwind-router:monitoring {
    list fp-cpu-usage {
      key cpu;

      description
        "The list of busy percentage per CPU.";

      leaf cpu {
        type string;
```

(continues on next page)

(continued from previous page)

```
        description
          "The CPU number.";
      }

      leaf busy {
        type uint16;
        description
          "The busy percentage.";
      }
    }

    container status {
      leaf text {
        type enumeration {
          enum ok;
          enum loaded;
          enum stopped;
          enum bad;
        }
        description
          "The Fast Path status in text.";
      }
      leaf value {
        type uint8;
        description
          "The Fast Path status in number.
           0 - ok, 1 - loaded, 2 - stopped, 3 - bad.";
      }
    }
  }
}
```

fp-ctx-switch-stats

```
submodule fp-ctx-switch-stats {

  belongs-to fp {
    prefix fp;
  }

  import sixwind-router {
    prefix sixwind-router;
```

(continues on next page)

(continued from previous page)

```
  }

  description
    "This submodule provides support for showing fp context switch stats
     information.";

  revision "2017-10-11" {
    description
      "Initial revision.";
  }

  augment /sixwind-router:monitoring/fp:statistics {
    list ctx-switch {
      key cpu;

      description
        "The list of context switch statistics per fast path core.";

      leaf cpu {
        type string;
        description
          "The CPU number.";
      }
      leaf voluntary {
        type uint64;
        description
          "Voluntary context switches.";
      }
      leaf nonvoluntary {
        type uint64;
        description
          "Nonvoluntary context switches.";
      }
    }
  }
}
```


fp-exception-queue-stats

```
submodule fp-exception-queue-stats {

  belongs-to fp {
    prefix fp;
  }
  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This submodule provides support for showing fp exception queues information.";

  revision "2018-02-02" {
    description
      "Initial revision.";
  }

  augment /sixwind-router:monitoring/fp:statistics {
    list exception-queue {
      key ring;

      description
        "The list of exception queue statistics per ring.";

      leaf ring {
        type string;
        description "The ring ID.";
      }
      leaf dequeue {
        type uint64;
        description
          "Number of packets dequeued by Linux from rx-ring or fast path from
           tx-ring.";
      }
      leaf dequeue-copy-errors {
        type uint64;
        description
          "Number of packets dropped due to copy error on dequeue operation
           (Linux only).";
      }
      leaf enqueue {
        type uint64;
```

(continues on next page)

(continued from previous page)

```
        description
          "Number of packets enqueued by Linux to tx-ring or by fast path to
           rx-ring.";
      }
      leaf enqueue-cp-kept {
        type uint64;
        description
          "Number of recognized control plane packets kept when the ring is more
           than half full.";
      }
      leaf enqueue-dp-drop {
        type uint64;
        description
          "Number of data plane packets dropped when the ring is more than
           half full.";
      }
      leaf enqueue-errors {
        type uint64;
        description
          "Number of packets dropped due to error on enqueue operation.";
      }
    }
  }
}
```

fp-exceptions-stats

```
submodule fp-exceptions-stats {

  belongs-to fp {
    prefix fp;
  }
  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This submodule provides support for showing fp exception information.";

  revision "2018-02-06" {
    description
      "Initial revision.";
```

(continues on next page)

(continued from previous page)

```
}

augment /sixwind-router:monitoring/fp:statistics {

  container exceptions {
    leaf FPTUN_IPV6_OUTPUT_EXCEPT {
      type uint64;
      description
        "Number of FPTUN exception packets for Linux IPv6 output.";
    }
    leaf FPTUN_IPV6_IPSECDONE_OUTPUT_EXCEPT {
      type uint64;
      description
        "Number of FPTUN exception packets for Linux IPv6 output after IPsec
         processing.";
    }
    leaf FPTUN_ETH_INPUT_EXCEPT {
      type uint64;
      description
        "Number of FPTUN exception packets for Linux ethernet input processing.";
    }
    leaf FPTUN_BASIC_EXCEPT {
      type uint64;
      description
        "Number of basic exception packets for Linux reception processing.";
    }
    leaf FPTUN_IPV6_IPSECDONE_INPUT_EXCEPT {
      type uint64;
      description
        "Number of FPTUN exception packets for Linux IPv6 input after IPsec
         processing.";
    }
    leaf FPTUN_IPV4_NATDONE_INPUT_EXCEPT {
      type uint64;
      description
        "Number of FPTUN exception packets for Linux IPv4 input after NAT
         processing.";
    }
    leaf FPTUN_IPV4_IPSECDONE_OUTPUT_EXCEPT {
      type uint64;
      description
        "Number of FPTUN exception packets for Linux IPv4 output after IPsec
         processing.";
    }
```

(continues on next page)

(continued from previous page)

```
      leaf FPTUN_IPV4_OUTPUT_EXCEPT {
        type uint64;
        description
          "Number of FPTUN exception packets for Linux IPv6 output.";
      }
      leaf FPTUN_ETH_NOVNB_INPUT_EXCEPT {
        type uint64;
        description
          "Number of FPTUN exception packets for Linux processing skipping VNB.";
      }
      leaf FPTUN_VNB2VNB_LINUX_TO_FP_EXCEPT {
        type uint64;
        description
          "Number of FPTUN exception packets for Linux VNB processing.";
      }
      leaf FPTUN_IPV4_IPSECDONE_INPUT_EXCEPT {
        type uint64;
        description
          "Number of FPTUN exception packets for Linux IPv4 input after IPsec
           processing.";
      }
      leaf FPTUN_IFACE_INPUT_EXCEPT {
        type uint64;
        description
          "Number of FPTUN exception packets for Linux VNB iface processing.";
      }
      leaf FPTUN_TAP {
        type uint64;
        description
          "Number of FPTUN exception packets for Linux TAP (tcpdump).";
      }
      leaf FPTUN_OUTPUT_EXCEPT {
        type uint64;
        description
          "Number of FPTUN exception packets for Linux interface output
           processing.";
      }
      leaf FPTUN_VNB2VNB_FP_TO_LINUX_EXCEPT {
        type uint64;
        description
          "Number of FPTUN VNB exception packets for Linux VNB input processing.";
      }
      leaf FPTUN_ETH_SP_OUTPUT_REQ {
        type uint64;
```

(continues on next page)

(continued from previous page)

```
        description
          "Number of FPTUN exception packets from Linux for fast path ethernet
           processing.";
      }
      leaf FPTUN_IPSEC_SP_OUTPUT_REQ {
        type uint64;
        description
          "Number of FPTUN exception packets from Linux for fast path IPsec
           processing.";
      }
      leaf FPTUN_IPV4_SP_OUTPUT_REQ {
        type uint64;
        description
          "Number of FPTUN exception packets from Linux for fast path IPv4
           processing.";
      }
      leaf FPTUN_IPV6_SP_OUTPUT_REQ {
        type uint64;
        description
          "Number of FPTUN exception packets from Linux for fast path IPv6
           processing.";
      }
      leaf FPTUN_RFPS_UPDATE {
        type uint64;
        description
          "Number of FPTUN exception packets for fast path statistics processing.";
      }
      leaf FPTUN_TRAFFIC_GEN_MSG {
        type uint64;
        description
          "Number of FPTUN exception packets for fast path traffic generator
           processing.";
      }
      leaf ExcpDroppedFpToLinuxAddMarkFailure {
        type uint64;
        description
          "Number of exception packets to Linux dropped due to a tag addition
           failure.";
      }
      leaf ExcpDroppedFpToLinuxNoIPv4RouteLocal {
        type uint64;
        description
          "Number of exception packets to Linux dropped due to a failure to find
           the IPv4 route.";
```

(continues on next page)

(continued from previous page)

```
      }
      leaf ExcpDroppedFpToLinuxNoIPv6RouteLocal {
        type uint64;
        description
          "Number of exception packets to Linux dropped due to a failure to find
           the IPv6 route.";
      }
      leaf ExcpDroppedFpToLinuxPrependFailure {
        type uint64;
        description
          "Number of exception packets to Linux dropped due to a prepend failure.";
      }
      leaf ExcpDroppedLinuxToFpGenericCommandFailure {
        type uint64;
        description
          "Number of packets from Linux dropped due to a FPTUN internal error.";
      }
      leaf ExcpDroppedLinuxToFpIPv4PullupFailure {
        type uint64;
        description
          "Number of packets from Linux dropped due to a failure when getting the
           IPv4 header of the FPTUN message.";
      }
      leaf ExcpDroppedLinuxToFpIPv6PullupFailure {
        type uint64;
        description
          "Number of packets from Linux dropped due to a failure when getting the
           IPv6 header of the FPTUN message.";
      }
      leaf ExcpDroppedLinuxToFpInvalidPortId {
        type uint64;
        description
          "Number of packets from Linux dropped due to a reception of a FPTUN
           message on an unexpected port.";
      }
      leaf ExcpDroppedLinuxToFpInvalidVersion {
        type uint64;
        description
          "Number of packets from Linux dropped due to an invalid FPTUN version.";
      }
      leaf ExcpDroppedLinuxToFpMsgTooShort {
        type uint64;
        description
          "Number of packets from Linux dropped due to an incomplete FPTUN
```

(continues on next page)

(continued from previous page)

```
           message.";
      }
      leaf ExcpDroppedLinuxToFpNoOutputFunction {
        type uint64;
        description
          "Number of packets from Linux dropped because no TX function has been
           registered.";
      }
      leaf ExcpDroppedLinuxToFpOtherHost {
        type uint64;
        description
          "Number of packets from Linux dropped due to a reception of a FPTUN
           message marked as PACKET_OTHERHOST.";
      }
      leaf ExcpDroppedLinuxToFpOutOperative {
        type uint64;
        description
          "Number of packets from Linux dropped because the destination interface
           is down.";
      }
      leaf ExcpDroppedLinuxToFpUnknownCommand {
        type uint64;
        description
          "Number of packets from Linux dropped due to an invalid FPTUN command.";
      }
      leaf ExcpDroppedLinuxToFpUnknownIfUid {
        type uint64;
        description
          "Number of packets from Linux dropped due to an invalid interface id for
           FPTUN.";
      }
    }
  }
}
```

fp-filling

```
submodule fp-filling {

  belongs-to fp {
    prefix fp;
  }
```

(continues on next page)

(continued from previous page)

```
import sixwind-router {
  prefix sixwind-router;
}

description
  "This submodule provides support for showing fp filling information.";

revision "2018-02-02" {
  description
    "Initial revision.";
}

augment /sixwind-router:monitoring {
  container filling {
    leaf interfaces-current {
      type uint64;
      description
        "Current number of interfaces in the fast path.";
    }
    leaf interfaces-max {
      type uint64;
      description
        "Maximum number of interfaces in the fast path.";
    }
    leaf neighbors-ipv4-current {
      type uint64;
      description
        "Current number of IPv4 neighbors in the fast path.";
    }
    leaf neighbors-ipv4-max {
      type uint64;
      description
        "Maximum number of IPv4 neighbors in the fast path.";
    }
    leaf routes-ipv4-current {
      type uint64;
      description
        "Current number of IPv4 routes in the fast path.";
    }
    leaf routes-ipv4-max {
      type uint64;
      description
        "Maximum number of IPv4 routes in the fast path.";
    }
```

(continues on next page)

(continued from previous page)

```
      leaf vrf-current {
        type uint64;
        description
          "Current number of VRFs (Virtual Routing Functions) in the fast path.";
      }
      leaf vrf-max {
        type uint64;
        description
          "Maximum number of VRFs (Virtual Routing Functions) in the fast path.";
      }
      leaf routing-tree-subtables-current {
        type uint64;
        description
          "Current number of routing tree subtables in the fast path.";
      }
      leaf routing-tree-subtables-max {
        type uint64;
        description
          "Maximum number of routing tree subtables in the fast path.";
      }
      leaf routing-tree-entries-current {
        type uint64;
        description
          "Current number of routing tree entries in the fast path.";
      }
      leaf routing-tree-entries-max {
        type uint64;
        description
          "Maximum number of routing tree entries in the fast path.";
      }
      leaf pbr-rules-current {
        type uint64;
        description
          "Current number of PBR (policy based routing) rules in the fast path.";
      }
      leaf pbr-rules-max {
        type uint64;
        description
          "Maximum number of PBR (policy based routing) rules in the fast path.";
      }
      leaf neighbors-ipv6-current {
        type uint64;
        description
          "Current number of IPv6 neighbors in the fast path.";
```

(continues on next page)

(continued from previous page)

```
      }
      leaf neighbors-ipv6-max {
        type uint64;
        description
          "Maximum number of IPv6 neighbors in the fast path.";
      }
      leaf routes-ipv6-current {
        type uint64;
        description
          "Current number of IPv6 routes in the fast path.";
      }
      leaf routes-ipv6-max {
        type uint64;
        description
          "Maximum number of IPv6 routes in the fast path.";
      }
      leaf ipsec-sa-current {
        type uint64;
        description
          "Current number of IPsec Security Associations in the fast path.";
      }
      leaf ipsec-sa-max {
        type uint64;
        description
          "Maximum number of IPsec Security Associations in the fast path.";
      }
      leaf ipsec-sp-current {
        type uint64;
        description
          "Current number of IPsec Security Policies in the fast path.";
      }
      leaf ipsec-sp-max {
        type uint64;
        description
          "Maximum number of IPsec Security Policies in the fast path.";
      }
      leaf svti-interfaces-current {
        type uint64;
        description
          "Current number of SVTI interfaces in the fast path.";
      }
      leaf svti-interfaces-max {
        type uint64;
        description
```

(continues on next page)

(continued from previous page)

```
        "Maximum number of SVTI interfaces in the fast path.";
    }
    leaf vxlan-port-current {
      type uint64;
      description
        "Current number of VXLAN UDP ports configured in the fast path.";
    }
    leaf vxlan-port-max {
      type uint64;
      description
        "Maximum number of VXLAN UDP ports configured in the fast path.";
    }
    leaf vxlan-interfaces-current {
      type uint64;
      description
        "Current number of VXLAN interfaces in the fast path.";
    }
    leaf vxlan-interfaces-max {
      type uint64;
      description
        "Maximum number of VXLAN interfaces in the fast path.";
    }
    leaf vxlan-fdb-current {
      type uint64;
      description
        "Current number of entries in VXLAN FDB (forwarding database) in the
         fast path.";
    }
    leaf vxlan-fdb-max {
      type uint64;
      description
        "Maximum number of entries in VXLAN FDB (forwarding database) in the
         fast path.";
    }
    leaf gre-interfaces-current {
      type uint64;
      description
        "Current number of GRE interfaces in the fast path.";
    }
    leaf gre-interfaces-max {
      type uint64;
      description
        "Maximum number of GRE interfaces in the fast path.";
    }
```

(continues on next page)

(continued from previous page)

```
      leaf vlan-interfaces-current {
        type uint64;
        description
          "Current number of VLAN interfaces in the fast path.";
      }
      leaf vlan-interfaces-max {
        type uint64;
        description
          "Maximum number of VLAN interfaces in the fast path.";
      }
      leaf macvlan-interfaces-current {
        type uint64;
        description
          "Current number of MACVLAN interfaces in the fast path.";
      }
      leaf macvlan-interfaces-max {
        type uint64;
        description
          "Maximum number of MACVLAN interfaces in the fast path.";
      }
      leaf lag-interfaces-current {
        type uint64;
        description
          "Current number of LAG interfaces in the fast path.";
      }
      leaf lag-interfaces-max {
        type uint64;
        description
          "Maximum number of LAG interfaces in the fast path.";
      }
      leaf lag-slave-interfaces-current {
        type uint64;
        description
          "Current number of LAG slave interfaces in the fast path.";
      }
      leaf lag-slave-interfaces-max {
        type uint64;
        description
          "Maximum number of LAG slave interfaces in the fast path.";
      }
      leaf bridge-ports-current {
        type uint64;
        description
          "Current number of bridge ports in the fast path.";
```

(continues on next page)

(continued from previous page)

```
      }
      leaf bridge-ports-max {
        type uint64;
        description
          "Maximum number of bridge ports in the fast path.";
      }
      leaf bridge-interfaces-current {
        type uint64;
        description
          "Current number of bridges in the fast path.";
      }
      leaf bridge-interfaces-max {
        type uint64;
        description
          "Maximum number of bridges in the fast path.";
      }
      leaf bridge-fdb-current {
        type uint64;
        description
          "Current number of entries in bridge FDB (forwarding database) in the
           fast path.";
      }
      leaf bridge-fdb-max {
        type uint64;
        description
          "Maximum number of entries in bridge FDB (forwarding database) in the
           fast path.";
      }
      leaf filterbridge-rules-current {
        type uint64;
        description
          "Current number of filter bridge rules in the fast path.";
      }
      leaf filterbridge-rules-max {
        type uint64;
        description
          "Maximum number of filter bridge rules in the fast path.";
      }
      leaf filter-ipv4-rules-current {
        type uint64;
        description
          "Current number of IPv4 filter rules in the fast path.";
      }
      leaf filter-ipv4-rules-max {
```

(continues on next page)

(continued from previous page)

```
        type uint64;
        description
          "Maximum number of IPv4 filter rules in the fast path.";
      }
      leaf filter-ipv6-rules-current {
        type uint64;
        description
          "Current number of IPv6 filter rules in the fast path.";
      }
      leaf filter-ipv6-rules-max {
        type uint64;
        description
          "Maximum number of IPv6 filter rules in the fast path.";
      }
      leaf conntracks-ipv4-current {
        type uint64;
        description
          "Current number of IPv4 conntracks in the fast path.";
      }
      leaf conntracks-ipv4-max {
        type uint64;
        description
          "Maximum number of IPv4 conntracks in the fast path.";
      }
      leaf conntracks-ipv6-current {
        type uint64;
        description
          "Current number of IPv6 conntracks in the fast path.";
      }
      leaf conntracks-ipv6-max {
        type uint64;
        description
          "Maximum number of IPv6 conntracks in the fast path.";
      }
      leaf ipsets-current {
        type uint64;
        description
          "Current number of ipsets in the fast path.";
      }
      leaf ipsets-max {
        type uint64;
        description
          "Maximum number of ipsets in the fast path.";
      }
```

(continues on next page)

(continued from previous page)

```
      leaf fp-vswitch-ports-current {
        type uint64;
        description
          "Current number of fp-vswitch ports in the fast path.";
      }
      leaf fp-vswitch-ports-max {
        type uint64;
        description
          "Maximum number of fp-vswitch ports in the fast path.";
      }
      leaf fp-vswitch-flows-current {
        type uint64;
        description
          "Current number of fp-vswitch flows in the fast path.";
      }
      leaf fp-vswitch-flows-max {
        type uint64;
        description
          "Maximum number of fp-vswitch flows in the fast path.";
      }
      leaf fp-vswitch-masks-current {
        type uint64;
        description
          "Current number of fp-vswitch masks in the fast path.";
      }
      leaf fp-vswitch-masks-max {
        type uint64;
        description
          "Maximum number of fp-vswitch masks in the fast path.";
      }
      leaf qos-sched-current {
        type uint64;
        description
          "Current number of QoS schedulers in the fast path.";
      }
      leaf qos-sched-max {
        type uint64;
        description
          "Maximum number of QoS schedulers in the fast path.";
      }
      leaf qos-class-current {
        type uint64;
        description
          "Current number of QoS classes in the fast path.";
```

(continues on next page)

(continued from previous page)

```
      }
      leaf qos-class-max {
        type uint64;
        description
          "Maximum number of QoS classes in the fast path.";
      }
      leaf qos-filter-current {
        type uint64;
        description
          "Current number of QoS filters in the fast path.";
      }
      leaf qos-filter-max {
        type uint64;
        description
          "Maximum number of QoS filters in the fast path.";
      }
      leaf qos-meter-current {
        type uint64;
        description
          "Current number of QoS meters in the fast path.";
      }
      leaf qos-meter-max {
        type uint64;
        description
          "Maximum number of QoS meters in the fast path.";
      }
    }
  }

}
```

fp-filling-cg-nat

```
submodule fp-filling-cg-nat {

  belongs-to fp {
    prefix fp;
  }
  import sixwind-router {
    prefix sixwind-router;
  }
```

(continues on next page)

(continued from previous page)

```
  description
    "This module provides support for showing fp cg-nat filling information.";

  revision "2019-07-03" {
    description
      "Initial revision.";
  }

  augment /sixwind-router:monitoring/fp:filling {
    leaf cgnat-conntracks-current {
      type uint64;
      description
        "Current number of conntracks for the cg-nat module.";
    }
    leaf cgnat-conntracks-max {
      type uint64;
      description
        "Maximum number of conntracks for the cg-nat module.";
    }
    leaf cgnat-nats-current {
      type uint64;
      description
        "Current number of nat entries for the cg-nat module.";
    }
    leaf cgnat-nats-max {
      type uint64;
      description
        "Maximum number of nat entries for the cg-nat module.";
    }
    leaf cgnat-cpes-current {
      type uint64;
      description
        "Current number of CPEs for the cg-nat module.";
    }
    leaf cgnat-cpes-max {
      type uint64;
      description
        "Maximum number of CPEs for the cg-nat module.";
    }
    leaf cgnat-blocks-current {
      type uint64;
      description
        "Current number of blocks for the cg-nat module.";
    }
```

(continues on next page)

(continued from previous page)

```
    leaf cgnat-blocks-max {
      type uint64;
      description
        "Maximum number of blocks for the cg-nat module.";
    }
  }
}
```

fp-global-stats

```
submodule fp-global-stats {

  belongs-to fp {
    prefix fp;
  }
  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This submodule provides support for showing fp global information.";

  revision "2018-02-02" {
    description
      "Initial revision.";
  }

  augment /sixwind-router:monitoring/fp:statistics {

    container global {
      leaf fp_dropped {
        type uint64;
        description
          "Number of packets dropped by fast path.";
      }
      leaf fp_dropped_arp {
        type uint64;
        description
          "Number of packets dropped by fast path in ARP.";
      }
      leaf fp_dropped_bonding {
        type uint64;
```

(continues on next page)

(continued from previous page)

```
        description
          "Number of packets dropped by fast path in bonding.";
      }
      leaf fp_dropped_bridge {
        type uint64;
        description
          "Number of packets dropped by fast path in bridge.";
      }
      leaf fp_dropped_ebpf {
        type uint64;
        description
          "Number of packets dropped by fast path in enhanced BPF (Berkeley
           Packet Filtering).";
      }
      leaf fp_dropped_ebtables {
        type uint64;
        description
          "Number of packets dropped by fast path in layer 2 filtering.";
      }
      leaf fp_dropped_ether {
        type uint64;
        description
          "Number of packets dropped by fast path at the generic ethernet layer.";
      }
      leaf fp_dropped_excp {
        type uint64;
        description
          "Number of packets dropped by fast path in exception path.";
      }
      leaf fp_dropped_gre {
        type uint64;
        description
          "Number of packets dropped by fast path in GRE.";
      }
      leaf fp_dropped_ib_roce {
        type uint64;
        description
          "Number of packets dropped by fast path in RDMA over Converged
           Ethernet.";
      }
      leaf fp_dropped_ip {
        type uint64;
        description
          "Number of packets dropped by fast path in generic IPv4.";
```

(continues on next page)

(continued from previous page)

```
      }
      leaf fp_dropped_ipsec {
        type uint64;
        description
          "Number of packets dropped by fast path in IPv4 IPsec.";
      }
      leaf fp_dropped_ipsec6 {
        type uint64;
        description
          "Number of packets dropped by fast path in IPv6 IPsec.";
      }
      leaf fp_dropped_ipv6 {
        type uint64;
        description
          "Number of packets dropped by fast path in generic IPv6.";
      }
      leaf fp_dropped_macvlan {
        type uint64;
        description
          "Number of packets dropped by fast path in MACVLAN.";
      }
      leaf fp_dropped_mcast {
        type uint64;
        description
          "Number of packets dropped by fast path in IPv4 multicast.";
      }
      leaf fp_dropped_mcast6 {
        type uint64;
        description
          "Number of packets dropped by fast path in IPv6 multicast.";
      }
      leaf fp_dropped_mpls {
        type uint64;
        description
          "Number of packets dropped by fast path in MultiProtocol Label
           Switching.";
      }
      leaf fp_dropped_netfilter {
        type uint64;
        description
          "Number of packets dropped by fast path in IPv4 filtering.";
      }
      leaf fp_dropped_netfilter6 {
        type uint64;
```

(continues on next page)

(continued from previous page)

```
        description
          "Number of packets dropped by fast path in IPv6 filtering.";
      }
      leaf fp_dropped_npf {
        type uint64;
        description
          "Number of packets dropped by fast path in Network Address Translation
           and Carrier-grade NAT.";
      }
      leaf fp_dropped_ovs {
        type uint64;
        description
          "Number of packets dropped by fast path in Open vSwitch.";
      }
      leaf fp_dropped_plugins {
        type uint64;
        description
          "Number of packets dropped by fast path in plugin.";
      }
      leaf fp_dropped_qos {
        type uint64;
        description
          "Number of packets dropped by fast path in QoS.";
      }
      leaf fp_dropped_reasm {
        type uint64;
        description
          "Number of packets dropped by fast path in IPv4 reassembly.";
      }
      leaf fp_dropped_reasm6 {
        type uint64;
        description
          "Number of packets dropped by fast path in IPv6 reassembly.";
      }
      leaf fp_dropped_system {
        type uint64;
        description
          "Number of packets dropped by fast path in internal processing.";
      }
      leaf fp_dropped_tc {
        type uint64;
        description
          "Number of packets dropped by fast path in generic traffic
           conditioning.";
```

(continues on next page)

(continued from previous page)

```
      }
      leaf fp_dropped_tc_erl {
        type uint64;
        description
          "Number of packets dropped by fast path in traffic conditioning by
           exception rate limitation.";
      }
      leaf fp_dropped_tgen {
        type uint64;
        description
          "Number of packets dropped by fast path in traffic generator.";
      }
      leaf fp_dropped_tunnel {
        type uint64;
        description
          "Number of packets dropped by fast path in IPinIP tunnel.";
      }
      leaf fp_dropped_vlan {
        type uint64;
        description
          "Number of packets dropped by fast path in VLAN.";
      }
      leaf fp_dropped_vnb {
        type uint64;
        description
          "Number of packets dropped by fast path in VNB.";
      }
      leaf fp_dropped_vxlan {
        type uint64;
        description
          "Number of packets dropped by fast path in VXLAN.";
      }
      leaf fp_exception_bonding {
        type uint64;
        description
          "Number of packets send in exception in bonding.";
      }
      leaf fp_exception_bridge {
        type uint64;
        description
          "Number of packets send in exception in bridge.";
      }
      leaf fp_exception_ebtables {
        type uint64;
```

(continues on next page)

(continued from previous page)

```
        description
          "Number of packets send in exception in layer 2 filtering.";
      }
      leaf fp_exception_ether {
        type uint64;
        description
          "Number of packets send in exception in generic layer 2.";
      }
      leaf fp_exception_gre {
        type uint64;
        description
          "Number of packets send in exception in GRE.";
      }
      leaf fp_exception_ifnet {
        type uint64;
        description
          "Number of packets send in exception by a virtual interface.";
      }
      leaf fp_exception_ip {
        type uint64;
        description
          "Number of packets send in exception in generic IPv4.";
      }
      leaf fp_exception_ipsec {
        type uint64;
        description
          "Number of packets send in exception in IPv4 IPsec.";
      }
      leaf fp_exception_ipsec6 {
        type uint64;
        description
          "Number of packets send in exception in IPv6 IPsec.";
      }
      leaf fp_exception_ipv6 {
        type uint64;
        description
          "Number of packets send in exception in generic IPv6.";
      }
      leaf fp_exception_macvlan {
        type uint64;
        description
          "Number of packets send in exception in MACVLAN.";
      }
      leaf fp_exception_mcast {
```

(continues on next page)

(continued from previous page)

```
        type uint64;
        description
          "Number of packets send in exception by IPv4 multicast.";
      }
      leaf fp_exception_mcast6 {
        type uint64;
        description
          "Number of packets send in exception by IPv6 multicast.";
      }
      leaf fp_exception_mpls {
        type uint64;
        description
          "Number of packets send in exception in MPLS.";
      }
      leaf fp_exception_netfilter {
        type uint64;
        description
          "Number of packets send in exception in IPv4 filtering.";
      }
      leaf fp_exception_netfilter6 {
        type uint64;
        description
          "Number of packets send in exception in IPv6 filtering.";
      }
      leaf fp_exception_netfpc {
        type uint64;
        description
          "Number of packets send in exception in netfpc. This case can occur
           with some cli commands.";
      }
      leaf fp_exception_npf {
        type uint64;
        description
          "Number of packets send in exception in NPF.";
      }
      leaf fp_exception_ovs {
        type uint64;
        description
          "Number of packets send in exception in Open vSwitch.";
      }
      leaf fp_exception_packet_steer {
        type uint64;
        description
          "Number of packets send in exception in packet steering";
```

(continues on next page)

(continued from previous page)

```
      }
      leaf fp_exception_reass {
        type uint64;
        description
          "Number of packets send in exception in IPv4 reassembly.";
      }
      leaf fp_exception_syslog {
        type uint64;
        description
          "Number of packets send in exception for logging (for system
           without syslog).";
      }
      leaf fp_exception_tap {
        type uint64;
        description
          "Number of packets send in exception in eBPF (Enhanced Berkeley
           Packet Filtering), typically when there is a tcpdump or sflow.";
      }
      leaf fp_exception_tunnel {
        type uint64;
        description
          "Number of packets send in exception in IPinIP tunnel.";
      }
      leaf fp_exception_vnb {
        type uint64;
        description
          "Number of packets send in exception in VNB.";
      }
      leaf fp_exception_vxlan {
        type uint64;
        description
          "Number of packets send in exception in VXLAN.";
      }
    }
  }
}
```

fp-gre-stats

```
submodule fp-gre-stats {

  belongs-to fp {
    prefix fp;
  }
  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This submodule provides support for showing fp gre information.";

  revision "2018-02-06" {
    description
      "Initial revision.";
  }

  augment /sixwind-router:monitoring/fp:statistics {

    container gre {
      leaf GREDroppedInIPv4Operative {
        type uint64;
        description
          "Number of input packets dropped in GRE because the incoming interface
           is down.";
      }
      leaf GREDroppedInIPv6Operative {
        type uint64;
        description
          "Number of input packets dropped in GRE6 because the incoming interface
           is down.";
      }
      leaf GREDroppedInitGreIPv4HeaderFailure {
        type uint64;
        description
          "Number of output packets dropped in GRE due to a failure to add the
           GRE header.";
      }
      leaf GREDroppedInitGreIPv6HeaderFailure {
        type uint64;
        description
          "Number of output packets dropped in GRE6 due to a failure to add the
```

(continues on next page)

(continued from previous page)

```
          GRE header.";
      }
      leaf GREDroppedInitIPv4HeaderFailure {
        type uint64;
        description
          "Number of output packets dropped in GRE due to a failure to add the
           IPv4 header.";
      }
      leaf GREDroppedInitIPv6HeaderFailure {
        type uint64;
        description
          "Number of output packets dropped in GRE due to a failure to add the
           IPv4 header.";
      }
      leaf GREDroppedMissingChecksum {
        type uint64;
        description
          "Number of input packets dropped in GRE due to a missing checksum.";
      }
      leaf GREDroppedOutOperative {
        type uint64;
        description
          "Number of output packets dropped in GRE because the outgoing interface
           is down.";
      }
      leaf GREDroppedParseIPv4HeaderFailure {
        type uint64;
        description
          "Number of input packets dropped in GRE due to failure to parse
           IPv4 header.";
      }
      leaf GREDroppedParseIPv6HeaderFailure {
        type uint64;
        description
          "Number of input packets dropped in GRE due to failure to parse
           IPv6 header.";
      }
      leaf GREDroppedPullupIPv4HeaderFailure {
        type uint64;
        description
          "Number of input packets dropped in IPv4 GRE due to pullup failure on
           gre header.";
      }
      leaf GREDroppedPullupIPv6HeaderFailure {
```

(continues on next page)

(continued from previous page)

```
        type uint64;
        description
          "Number of input packets dropped in IPv6 GRE due to pullup failure on
           gre header.";
      }
      leaf GREDroppedUnexpectedChecksum {
        type uint64;
        description
          "Number of input packets dropped in GRE due to an unexpected checksum.";
      }
      leaf GREDroppedWrongChecksum {
        type uint64;
        description
          "Number of input packets dropped in GRE due to an incorrect checksum.";
      }
      leaf GREExceptionIPv4Route {
        type uint64;
        description
          "Number of output packets sent to exception by GRE due to specific route
           (for IPv4 packet).";
      }
      leaf GREExceptionIPv4SourceSelectFailed {
        type uint64;
        description
          "Number of output packets sent to exception by GRE due to no src address
           can be set (for IPv4 packet).";
      }
      leaf GREExceptionIPv6Route {
        type uint64;
        description
          "Number of output packets sent to exception by GRE due to specific route
           (for IPv6 packet).";
      }
      leaf GREExceptionInputUnsupportedProtocol {
        type uint64;
        description
          "Number of input packets sent to exception by GRE due to unsupported
           GRE protocol.";
      }
      leaf GREExceptionOutputUnsupportedProtocol {
        type uint64;
        description
          "Number of output packets sent to exception by GRE due to unsupported
           GRE protocol.";
```

(continues on next page)

(continued from previous page)

```
      }
      leaf GREExceptionUnknownIface {
        type uint64;
        description
          "Number of output packets sent to exception by GRE due to an invalid
           GRE interface id.";
      }
      leaf GREExceptionUnsupportedEtherType {
        type uint64;
        description
          "Number of output packets sent to exception by GRE due to unsupported
           ethernet type.";
      }
      leaf GREExceptionUnsupportedFamily {
        type uint64;
        description
          "Number of output packets sent to exception by GRE due to unsupported
           IP family (case of IPv6 with no support of IPv6 by fastpath).";
      }
      leaf GREInvalidHeader {
        type uint64;
        description
          "Number of input packets not managed by GRE due to routing flags set
           (see rfc 1701) or version number different to 0. The packet can be
           dropped or sent to exception later in other fast path processing part.";
      }
      leaf GRETAPDroppedInOperative {
        type uint64;
        description
          "Number of input packets dropped in GRETAP because the incoming
           interface is down.";
      }
      leaf GRETAPDroppedOutOperative {
        type uint64;
        description
          "Number of output packets dropped in GRETAP because the outgoing
           interface is down.";
      }
      leaf GRETAPExceptionUnknownIface {
        type uint64;
        description
          "Number of output packets sent to exception by GRETAP due to an invalid
           GRE interface id.";
      }
```

(continues on next page)

(continued from previous page)

```
    }
  }
}
```

fp-ip-stats

```
submodule fp-ip-stats {

  belongs-to fp {
    prefix fp;
  }
  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This submodule provides support for showing fp ip information.";

  revision "2018-02-06" {
    description
      "Initial revision.";
  }

  augment /sixwind-router:monitoring/fp:statistics {

    container ip {
      leaf IpForwDatagrams {
        type uint64;
        description
          "Number of IP packets forwarded.";
      }
      leaf IpInReceives {
        type uint64;
        description
          "Number of IP packets received.";
      }
      leaf IpInDelivers {
        type uint64;
        description
          "Number of IP packets delivered to user-protocols.";
      }
      leaf IpInHdrErrors {
```

(continues on next page)

(continued from previous page)

```
        type uint64;
        description
          "Number of IP packets discarded due to errors in header.";
      }
      leaf IpDroppedForwarding {
        type uint64;
        description
          "Number of IP packets discarded due to forwarding being disabled.";
      }
      leaf IpInAddrErrors {
        type uint64;
        description
          "Number of IP packets discarded due to invalid IP address.";
      }
      leaf IpInTruncatedPkts {
        type uint64;
        description
          "Number of IP packets discarded due to a truncate IP header.";
      }
      leaf IpCsumErrors {
        type uint64;
        description
          "Number of IP packets discarded due to an invalid checksum.";
      }
      leaf IpDroppedNoMemory {
        type uint64;
        description
          "Number of IP packets discarded due to memory allocation errors.";
      }
      leaf IpDroppedNoArp {
        type uint64;
        description
          "Number of IP packets discarded due to missing ARP resolution.";
      }
      leaf IpDroppedIPsec {
        type uint64;
        description
          "Number of IP packets discarded by IPsec processing.";
      }
      leaf IpDroppedBlackhole {
        type uint64;
        description
          "Number of IP packets discarded due to matching blackhole route.";
      }
```

(continues on next page)

(continued from previous page)

```
      leaf IpDroppedRouteException {
        type uint64;
        description
          "Number of IP packets sent to exception due to specific route";
      }
      leaf IpReasmOKs {
        type uint64;
        description
          "Number of IP packets successfully reassembled.";
      }
      leaf IpDroppedInvalidInterface {
        type uint64;
        description
          "Number of IP packets discarded due to invalid outgoing interface.";
      }
      leaf IPDroppedOutOperative {
        type uint64;
        description
          "Number of IP packets discarded because the outgoing interface is down.";
      }
      leaf IpDroppedNetfilter {
        type uint64;
        description
          "Number of IP packets discarded by filtering processing.";
      }
      leaf IpReasmTimeout {
        type uint64;
        description
          "Number of IP packets discarded due to timeout in reassembly
           processing.";
      }
      leaf IpReasmReqds {
        type uint64;
        description
          "Number of IP fragments packets submitted to reassembly processing.";
      }
      leaf IpReasmFails {
        type uint64;
        description
          "Number of IP packets discarded due to failures during reassembly
           processing.";
      }
      leaf IpReasmExceptions {
        type uint64;
```

(continues on next page)

(continued from previous page)

```
        description
          "Number of IP fragment packets sent in exception path.";
      }
      leaf IpReasmErrorHeaderEncap {
        type uint64;
        description
          "Number of IP packets discarded during reassembly due to header
           encapsulation error.";
      }
      leaf IpReasmErrorIPOptionUnsupported {
        type uint64;
        description
          "Number of IP packets discarded during reassembly due to unsupported
           IP option.";
      }
      leaf IpReasmErrorLastAlreadyReceived {
        type uint64;
        description
          "Number of IP packets discarded during reassembly due to receive twice
           the last fragment.";
      }
      leaf IpReasmErrorOffsetTooLarge {
        type uint64;
        description
          "Number of IP packets discarded during reassembly with offset due to an
           offset too big.";
      }
      leaf IpReasmErrorOverlapNext {
        type uint64;
        description
          "Number of IP packets discarded during reassembly due to receive
           overlapping fragment with next one.";
      }
      leaf IpReasmErrorOverlapPrevious {
        type uint64;
        description
          "Number of IP packets discarded during reassembly due to receive
           overlapping fragment with previous one.";
      }
      leaf IpReasmErrorPacketTooShort {
        type uint64;
        description
          "Number of IP packets discarded during reassembly due to reception of
           a too short fragment.";
```

(continues on next page)

(continued from previous page)

```
      }
      leaf IpReasmErrorQueueAlloc {
        type uint64;
        description
          "Number of IP packets discarded during reassembly due to reassembly
           queue allocation failure.";
      }
      leaf IpReasmErrorQueueFull {
        type uint64;
        description
          "Number of IP packets discarded during reassembly due to reassembly
           queue full (too many fragments have been received).";
      }
      leaf IpReasmErrorSizeExceed {
        type uint64;
        description
          "Number of IP packets discarded during reassembly due to total received
           bytes greater than the maximal authorized value (65535).";
      }
      leaf IpReasmErrorSizeOverflow {
        type uint64;
        description
          "Number of IP packets discarded during reassembly due to total received
           bytes greater than the expected value.";
      }
      leaf IpReasmErrorTooManySegments {
        type uint64;
        description
          "Number of IP packets discarded during reassembly due to too many
           segments in IP packets.";
      }
      leaf IpFragCreates {
        type uint64;
        description
          "Number of IP fragment packets created on fragmentation processing.";
      }
      leaf IpFragOKs {
        type uint64;
        description
          "Number of IP fragment packets sent successfully.";
      }
      leaf IpFragFails {
        type uint64;
        description
```

(continues on next page)

(continued from previous page)

```
        "Number of IP packets discarded due to failures during fragmentation
         processing.";
    }
  }

  container ip6 {
    leaf IpForwDatagrams {
      type uint64;
      description
        "Number of IPv6 packets forwarded.";
    }
    leaf IpInReceives {
      type uint64;
      description
        "Number of IPv6 packets received.";
    }
    leaf IpInDelivers {
      type uint64;
      description
        "Number of IPv6 packets delivered to user-protocols.";
    }
    leaf IpInHdrErrors {
      type uint64;
      description
        "Number of IPv6 packets discarded due to errors in header.";
    }
    leaf IpDroppedForwarding {
      type uint64;
      description
        "Number of IPv6 packets discarded due to forwarding being disabled.";
    }
    leaf IpInAddrErrors {
      type uint64;
      description
        "Number of IPv6 packets discarded due to invalid IPv6 address.";
    }
    leaf IpInTruncatedPkts {
      type uint64;
      description
        "Number of IPv6 packets discarded due to a truncate IP header.";
    }
    leaf IpDroppedNoMemory {
      type uint64;
      description
```

(continues on next page)

(continued from previous page)

```
        "Number of IPv6 packets discarded due to memory allocation errors.";
    }
    leaf IpDroppedNoArp {
      type uint64;
      description
        "Number of IPv6 packets discarded due to missing ARP resolution.";
    }
    leaf IpDroppedIPsec {
      type uint64;
      description
        "Number of IPv6 packets discarded by IPsec processing.";
    }
    leaf IpDroppedBlackhole {
      type uint64;
      description
        "Number of IPv6 packets discarded due to matching blackhole route.";
    }
    leaf IpDroppedRouteException {
      type uint64;
      description
        "Number of IPv6 packets sent to exception due to specific route";
    }
    leaf IpReasmOKs {
      type uint64;
      description
        "Number of IPv6 packets successfully reassembled.";
    }
    leaf IpDroppedInvalidInterface {
      type uint64;
      description
        "Number of IPv6 packets discarded due to invalid outgoing interface.";
    }
    leaf IPDroppedOutOperative {
      type uint64;
      description
        "Number of IP packets discarded because the outgoing interface is down.";
    }
    leaf IpDroppedNetfilter {
      type uint64;
      description
        "Number of IPv6 packets discarded by filtering processing.";
    }
    leaf IpReasmTimeout {
      type uint64;
```

(continues on next page)

(continued from previous page)

```
        description
          "Number of IPv6 packets discarded due to timeout in reassembly
           processing.";
      }
      leaf IpReasmReqds {
        type uint64;
        description
          "Number of IPv6 fragments packets submitted to reassembly processing.";
      }
      leaf IpReasmFails {
        type uint64;
        description
          "Number of IPv6 packets discarded due to failures during reassembly
           processing.";
      }
      leaf IpReasmExceptions {
        type uint64;
        description
          "Number of IPv6 fragment packets sent in exception path.";
      }
      leaf IpReasmErrorFragmentHeader {
        type uint64;
        description
          "Number of IPv6 packets discarded during reassembly due to header
           reading error.";
      }
      leaf IpReasmErrorHeaderEncap {
        type uint64;
        description
          "Number of IPv6 packets discarded during reassembly due to header
           encapsulation error.";
      }
      leaf IpReasmErrorIPOptionTooLarge {
        type uint64;
        description
          "Number of IPv6 packets discarded during reassembly due to IPv6 option
           too large.";
      }
      leaf IpReasmErrorLastAlreadyReceived {
        type uint64;
        description
          "Number of IPv6 packets discarded during reassembly due to receive
           twice the last fragment.";
      }
```

(continues on next page)

(continued from previous page)

```
      leaf IpReasmErrorOffsetTooLarge {
        type uint64;
        description
          "Number of IPv6 packets discarded during reassembly with offset due to
           an offset too big.";
      }
      leaf IpReasmErrorOverlapNext {
        type uint64;
        description
          "Number of IPv6 packets discarded during reassembly due to receive
           overlapping fragment with next one.";
      }
      leaf IpReasmErrorOverlapPrevious {
        type uint64;
        description
          "Number of IPv6 packets discarded during reassembly due to receive
           overlapping fragment with previous one.";
      }
      leaf IpReasmErrorPacketTooShort {
        type uint64;
        description
          "Number of IPv6 packets discarded during reassembly due to reception
           of a too short fragment.";
      }
      leaf IpReasmErrorQueueAlloc {
        type uint64;
        description
          "Number of IPv6 packets discarded during reassembly due to reassembly
           queue allocation failure.";
      }
      leaf IpReasmErrorQueueFull {
        type uint64;
        description
          "Number of IPv6 packets discarded during reassembly due to reassembly
           queue full (too many fragments have been received).";
      }
      leaf IpReasmErrorSizeExceed {
        type uint64;
        description
          "Number of IPv6 packets discarded during reassembly due to total
           received bytes greater than the maximal authorized value (65535).";
      }
      leaf IpReasmErrorSizeOverflow {
        type uint64;
```

(continues on next page)

(continued from previous page)

```
        description
          "Number of IPv6 packets discarded during reassembly due to total
           received bytes greater than the expected value.";
      }
      leaf IpReasmErrorTooManySegments {
        type uint64;
        description
          "Number of IPv6 packets discarded during reassembly due to too many
           segments in IP packets.";
      }
      leaf IpFragCreates {
        type uint64;
        description
          "Number of IPv6 fragment packets created on fragmentation processing.";
      }
      leaf IpFragOKs {
        type uint64;
        description
          "Number of IPv6 fragment packets sent successfully.";
      }
      leaf IpFragFails {
        type uint64;
        description
          "Number of IPv6 packets discarded due to failures during fragmentation
           processing.";
      }
    }
  }
}
```

fp-ipsec-stats

```
submodule fp-ipsec-stats {

  belongs-to fp {
    prefix fp;
  }
  import sixwind-router {
    prefix sixwind-router;
  }

  description
```

(continues on next page)

(continued from previous page)

```
  "This submodule provides support for showing fp ipsec information.";

revision "2018-02-06" {
  description
    "Initial revision.";
}

augment /sixwind-router:monitoring/fp:statistics {

  container ipsec {
    leaf IpsecDroppedInError  {
      type uint64;
      description
        "Number of input packets dropped in IPv4 IPsec due to a generic error.";
    }
    leaf IpsecDroppedInHdrError  {
      type uint64;
      description
        "Number of input packets dropped in IPv4 IPsec due to an header error.";
    }
    leaf IpsecDroppedInNoPols  {
      type uint64;
      description
        "Number of input packets dropped in IPv4 IPsec because no policy is
         found for states (i.e. Inbound SAs are correct but no SP is found).";
    }
    leaf IpsecDroppedInNoStates  {
      type uint64;
      description
        "Number of input packets dropped in IPv4 IPsec because no state is found
         (i.e. Either inbound SPI, address, or IPsec protocol at SA is wrong).";
    }
    leaf IpsecDroppedInPolBlock  {
      type uint64;
      description
        "Number of input packets dropped in IPv4 IPsec due to a policy discard.";
    }
    leaf IpsecDroppedInPolError  {
      type uint64;
      description
        "Number of input packets dropped in IPv4 IPsec due to a policy error.";
    }
    leaf IpsecDroppedInStateModeError  {
      type uint64;
```

(continues on next page)

(continued from previous page)

```
        description
          "Number of input packets dropped in IPv4 IPsec due to a transformation
           mode specific error.";
      }
      leaf IpsecDroppedInStateSeqError  {
        type uint64;
        description
          "Number of input packets dropped in IPv4 IPsec due to a sequence error
           (i.e. Sequence number is out of window).";
      }
      leaf IpsecDroppedOutError  {
        type uint64;
        description
          "Number of output packets dropped in IPv4 IPsec due to a generic error.";
      }
      leaf IpsecDroppedOutNoStates  {
        type uint64;
        description
          "Number of output packets dropped in IPv4 IPsec because no state is
           found (i.e. Either outbound SPI, address, or IPsec protocol at
           SA is wrong).";
      }
      leaf IpsecDroppedOutPolDead  {
        type uint64;
        description
          "Number of output packets dropped in IPv4 IPsec because policy is dead.";
      }
      leaf IpsecDroppedOutPolError  {
        type uint64;
        description
          "Number of output packets dropped in IPv4 IPsec due to a policy error.";
      }
      leaf IpsecDroppedOutStateModeError  {
        type uint64;
        description
          "Number of output packets dropped in IPv4 IPsec due to a transformation
           mode specific error.";
      }
      leaf IpsecDroppedOutStateSeqError  {
        type uint64;
        description
          "Number of output packets dropped in IPv4 IPsec due to a sequence error
           (i.e. Sequence number is out of window).";
      }
```

(continues on next page)

(continued from previous page)

```
      leaf IpsecExceptionUnsupportedMode {
        type uint64;
        description
          "Number of packets sent to exception due to unsupported SA mode.";
      }
    }

    container ipsec6 {
      leaf Ipsec6DroppedInError {
        type uint64;
        description
          "Number of input packets dropped in IPv6 IPsec due to a generic error.";
      }
      leaf Ipsec6DroppedInHdrError {
        type uint64;
        description
          "Number of input packets dropped in IPv6 IPsec due to an header error.";
      }
      leaf Ipsec6DroppedInNoPols {
        type uint64;
        description
          "Number of input packets dropped in IPv6 IPsec because no policy is
           found for states (i.e. Inbound SAs are correct but no SP is found).";
      }
      leaf Ipsec6DroppedInNoStates {
        type uint64;
        description
          "Number of input packets dropped in IPv6 IPsec because no state is found
           (i.e. Either inbound SPI, address, or IPsec protocol at SA is wrong).";
      }
      leaf Ipsec6DroppedInPolBlock {
        type uint64;
        description
          "Number of input packets dropped in IPv6 IPsec due to a policy discard.";
      }
      leaf Ipsec6DroppedInPolError {
        type uint64;
        description
          "Number of input packets dropped in IPv6 IPsec due to a policy error.";
      }
      leaf Ipsec6DroppedInStateModeError {
        type uint64;
        description
          "Number of input packets dropped in IPv6 IPsec due to a transformation
```

(continues on next page)

(continued from previous page)

```
          mode specific error.";
      }
      leaf Ipsec6DroppedInStateSeqError {
        type uint64;
        description
          "Number of input packets dropped in IPv6 IPsec due to a sequence error
           (i.e. Sequence number is out of window).";
      }
      leaf Ipsec6DroppedOutError {
        type uint64;
        description
          "Number of output packets dropped in IPv6 IPsec due to a generic error.";
      }
      leaf Ipsec6DroppedOutNoStates {
        type uint64;
        description
          "Number of output packets dropped in IPv6 IPsec because no state is
           found (i.e. Either outbound SPI, address, or IPsec protocol at SA is
           wrong).";
      }
      leaf Ipsec6DroppedOutPolDead {
        type uint64;
        description
          "Number of output packets dropped in IPv6 IPsec because policy is dead.";
      }
      leaf Ipsec6DroppedOutPolError {
        type uint64;
        description
          "Number of output packets dropped in IPv6 IPsec due to a policy error.";
      }
      leaf Ipsec6DroppedOutStateModeError {
        type uint64;
        description
          "Number of output packets dropped in IPv6 IPsec due to a transformation
           mode specific error.";
      }
      leaf Ipsec6DroppedOutStateSeqError {
        type uint64;
        description
          "Number of output packets dropped in IPv6 IPsec due to a sequence error
           (i.e. Sequence number is out of window).";
      }
      leaf Ipsec6ExceptionUnsupportedMode {
        type uint64;
```

(continues on next page)

(continued from previous page)

```
        description
          "Number of packets sent to exception due to unsupported SA mode.";
      }
    }
  }
}
```

fp-l2-stats

```
submodule fp-l2-stats {

  belongs-to fp {
    prefix fp;
  }
  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This submodule provides support for showing fp l2 information.";

  revision "2018-02-06" {
    description
      "Initial revision.";
  }

  augment /sixwind-router:monitoring/fp:statistics {

    container bridge {
      leaf BridgeDroppedFwdInvalid {
        type uint64;
        description
          "Number of output packets dropped in bridge due to forbidden forwarding
           (forwarding disable or originating port).";
      }
      leaf BridgeDroppedInputLookupError {
        type uint64;
        description
          "Number of input packets dropped in bridge due to a lookup error.";
      }
      leaf BridgeDroppedInvalidOutPort {
        type uint64;
```

(continues on next page)

(continued from previous page)

```
        description
          "Number of output packets dropped in bridge because output port index
           is invalid.";
      }
      leaf BridgeDroppedInvalidSrc {
        type uint64;
        description
          "Number of input packets dropped in bridge due to invalid mac source.";
      }
      leaf BridgeDroppedInvalidState {
        type uint64;
        description
          "Number of input packets dropped in bridge due to invalid state (not
           learning or forwarding) of the bridge.";
      }
      leaf BridgeDroppedLearning {
        type uint64;
        description
          "Number of output packets dropped in bridge while it is in learning
           state.";
      }
      leaf BridgeDroppedMtuExceeded {
        type uint64;
        description
          "Number of output packets dropped in bridge due to MTU greater than
           the authorized one.";
      }
      leaf BridgeDroppedNoOutputPort {
        type uint64;
        description
          "Number of output packets dropped in bridge due to no valid output.";
      }
      leaf BridgeDroppedOutOperative {
        type uint64;
        description
          "Number of output packets dropped in bridge because the outgoing
           interface is down.";
      }
      leaf BridgeDroppedOutputLookupError {
        type uint64;
        description
          "Number of output packets dropped in bridge due to a lookup error.";
      }
      leaf BridgeDroppedOutputUnknown {
```

(continues on next page)

(continued from previous page)

```
      type uint64;
      description
        "Number of output packets dropped in bridge due to an unknown output.";
    }
    leaf BridgeDroppedPauseFrame {
      type uint64;
      description
        "Number of input packets dropped in bridge because it is a pause frame.";
    }
    leaf BridgeDroppedUnknownIface {
      type uint64;
      description
        "Number of input packets dropped in bridge due to an invalid interface.";
    }
    leaf L2ForwFrames {
      type uint64;
      description
        "Number of packets forwarded at layer 2 (bridging processing).";
    }
  }

  container ebtables {
    leaf L2FilterDroppedHeaderTooShort {
      type uint64;
      description
        "Number of packets dropped in L2 filtering due to a too short ethernet
         frame.";
    }
    leaf L2FilterDroppedIpInvalid {
      type uint64;
      description
        "Number of packets dropped in L2 filtering due to an invalid
         IPv4 header.";
    }
    leaf L2FilterDroppedIpv6Invalid {
      type uint64;
      description
        "Number of packets dropped in L2 filtering due to an invalid
         IPv6 header.";
    }
    leaf L2FilterDroppedPrependFailure {
      type uint64;
      description
        "Number of packets dropped in L2 filtering due to a failure to restore
```

(continues on next page)

(continued from previous page)

```
          ethernet frame.";
      }
      leaf L2FilterDroppedVerdict {
        type uint64;
        description
          "Number of packets dropped in L2 filtering due to L2 drop filter rule.";
      }
    }

    container vlan {
      leaf VlanDroppedInOperative {
        type uint64;
        description
          "Number of input packets dropped in VLAN because the incoming interface
           is down.";
      }
      leaf VlanDroppedInputUnknownIf {
        type uint64;
        description
          "Number of input packets dropped in VLAN due to unknown interface.";
      }
      leaf VlanDroppedInvalidTag {
        type uint64;
        description
          "Number of input packets dropped in VLAN due to an invalid tag.";
      }
      leaf VlanDroppedOutOperative {
        type uint64;
        description
          "Number of output packets dropped in VLAN because the outgoing interface
           is down.";
      }
      leaf VlanDroppedOutputUnknownIf {
        type uint64;
        description
          "Number of output packets dropped in VLAN due to unknown interface.";
      }
      leaf VlanDroppedPrependFailure {
        type uint64;
        description
          "Number of output packets dropped in VLAN due to a failure to add
           VLAN tag.";
      }
      leaf VlanUnknownTag {
```

(continues on next page)

(continued from previous page)

```
        type uint64;
        description
          "Number of packets with unknown VLAN tag.";
      }
    }
  }
}
```

fp-cg-nat-stats

```
submodule fp-cg-nat-stats {

  belongs-to fp {
    prefix fp;
  }
  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This module provides support for showing fp cg-nat information.";

  revision 2019-06-12 {
    description
      "Rename npf to cg-nat.";
  }
  revision "2018-02-02" {
    description
      "Initial revision.";
  }

  augment /sixwind-router:monitoring/fp:statistics {
    container cg-nat {
      leaf default-passed-packets {
        type uint64;
        description
          "Number of packets passed because it matches the default rule.";
      }
      leaf ruleset-passed-packets {
        type uint64;
        description
          "Number of packets passed because it matches a rule.";
```

(continues on next page)

(continued from previous page)

```
      }
      leaf state-passed-packets {
        type uint64;
        description
          "Number of packets passed because it matches a conntrack.";
      }
      leaf default-blocked-packets {
        type uint64;
        description
          "Number of packets dropped because it matches the default rule.";
      }
      leaf ruleset-blocked-packets {
        type uint64;
        description
          "Number of packets dropped because it matches a rule.";
      }
      leaf state-allocations {
        type uint64;
        description
          "Number of conntrack allocations.";
      }
      leaf state-allocation-failures {
        type uint64;
        description
          "Number of conntrack allocation failures. It happens when there are no
           conntracks available in the pool.";
      }
      leaf state-reverse {
        type uint64;
        description
          "Number of reversed conntracks.";
      }
      leaf state-destructions {
        type uint64;
        description
          "Number of conntrack destructions.";
      }
      leaf nat-allocations {
        type uint64;
        description
          "Number of nat allocations";
      }
      leaf nat-allocation-failures {
        type uint64;
```

(continues on next page)

(continued from previous page)

```
        description
          "Number of nat allocation failures. It happens when there are no
           nat entry available in the pool.";
      }
      leaf nat-destructions {
        type uint64;
        description
          "Number of nat destructions.";
      }
      leaf nat-port-allocation-failures {
        type uint64;
        description
          "Number of nat port allocation failures.";
      }
      leaf cpe-allocations {
        type uint64;
        description
          "Number of CPE allocations.";
      }
      leaf cpe-destructions {
        type uint64;
        description
          "Number of CPE destructions.";
      }
      leaf cpe-allocation-failures {
        type uint64;
        description
          "Number of CPE allocation failures. It happens when there are no
           CPE available in the pool.";
      }
      leaf block-allocations {
        type uint64;
        description
          "Number of block allocations.";
      }
      leaf block-allocation-failures {
        type uint64;
        description
          "Number of block allocation failures. It happens when there are no
           blocks available in the pool.";
      }
      leaf block-destructions {
        type uint64;
        description
```

(continues on next page)

(continued from previous page)

```
          "Number of block destructions.";
      }
      leaf no-public-ip-errors {
        type uint64;
        description
          "Number of CPE allocation failures because all the public ips are used.";
      }
      leaf full-public-ip-errors {
        type uint64;
        description
          "Number of block allocation failures because there are no port blocks
           available in the public ip of the CPE.";
      }
      leaf invalid-total-packet-states {
        type uint64;
        description
           "Number of total dropped packets because they have an invalid state.";
     }
     leaf invalid-first-tcp-packet-states {
        type uint64;
        description
           "Number of TCP conntrack we failed to create because packet is not a
           SYN one.";
     }
     leaf invalid-transition-tcp-packet-states {
        type uint64;
        description
           "Number of TCP packets dropped because there are no valid transitions
            between the current TCP state and the packet (i.e. TCP state machine
            is updated in function of the packet's TCP flags).";
     }
     leaf invalid-rst-tcp-packet-states {
        type uint64;
        description
           "Number of out-of-order TCP rst packets dropped (See RFC 5961).";
     }
     leaf invalid-tcp-case1-packet-states {
        type uint64;
        description
           "Number of TCP packets dropped because it is out of the TCP window
           (upper boundary).";
     }
     leaf invalid-tcp-case2-packet-states {
        type uint64;
```

(continues on next page)

(continued from previous page)

```
        description
           "Number of TCP packets dropped because it is out of the TCP window
           (lower boundary).";
     }
     leaf invalid-tcp-case3-packet-states {
        type uint64;
        description
           "Number of ACK TCP packets dropped because it acknowledges a packet
           not sent.";
     }
     leaf cpe-creation-races {
        type uint64;
        description
          "Number of CPE creation race. It happens when a cpu try to create a
           new CPE. But, this CPE has already been created by an other cpu in the
           meantime. In this case, new CPE is dropped, current one is used.";
      }
      leaf cpe-association-races {
        type uint64;
        description
          "Number of CPE association race. This race happens when it is not
           possible take a reference on a CPE. It means that the CPE has been
           released by other cpu or the maximun of references (i.e. maximun of
           ports that can be nated) is reached.";
      }
      leaf nat-association-races {
        type uint64;
        description
          "Number of nat association race. It happens when we try to associate a
           nat object to a connection object. But, this one has been already
           associated by on other cpu in the meantime.";
      }
      leaf duplicate-state-races {
        type uint64;
        description
          "Number of duplicate conntracks. It happens when we try to create a
           conntrack. But, this one has been already created by on other cpu in
           the meantime.";
      }
      leaf hairpinning-packets {
        type uint64;
        description
          "Number of hairpinning packets. Number of packets forwarded between
          two hosts behind the same NAT device.";
```

(continues on next page)

(continued from previous page)

```
      }
      leaf loop-hairpinning-packets {
        type uint64;
        description
          "Number of hairpinning packets dropped because a routing loop has been
          detected.";
      }
      leaf self-hairpinning-packets {
        type uint64;
        description
          "Number of hairpinning packets dropped because an host tries to send
          packets on its own NATed address.";
      }
      leaf nat64-invalid-udp-null-checksum {
        type uint64;
        description
          "Number of IPv4 udp null checksum packets dropped. In IPv6, null
          checksum are not allowed for UDP. When nating from IPv4 to IPv6, if
          UDP checksum is NULL, it can not be mangled. Thus, in this case, it's
          dropped. It's possible to compute a full checksum by changing an
          an option for NAT64";
      }
    }
  }
}
```

fp-ports-stats

```
submodule fp-ports-stats {

  belongs-to fp {
    prefix fp;
  }
  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This submodule provides support for showing fp port information.";

  revision "2018-02-02" {
    description
```

(continues on next page)

(continued from previous page)

```
    "Initial revision.";
}

augment /sixwind-router:monitoring/fp:statistics {
  list control-plane-protection {
    key "name vrf";

    description
      "The control plane protection module statistics per network interface.";

    leaf name {
      type string;
      description
        "The interface name.";
    }

    leaf vrf {
      type string;
      description
        "The interface vrf.";
    }

    leaf rx_cp_kept {
      type uint64;
      description
        "When the Rx ring filling reaches a threshold, packets are inspected
         by the Control Plane Protection mechanism. This statistic is incremented
         for packets recognized as CP packets, which are kept and processed by
         the stack.";
    }

    leaf rx_cp_overrun {
      type uint64;
      description
        "When CPU consumption used by Control Plane Protection exceeds a
         threshold, this system is disabled. This statistic is incremented for
         each Rx packet not analyzed by Control Plane Protection due to CPU
         overload.";
    }

    leaf rx_cp_passthrough {
      type uint64;
      description
        "When Control Plane Protection is enabled, this statistic is
```

(continues on next page)

(continued from previous page)

```
          incremented for each packet received when machine is not
          overloaded. These packets are processed normally.";
      }

      leaf rx_dp_drop {
        type uint64;
        description
          "When the Rx ring filling reaches a threshold, packets are inspected
           by the Control Plane Protection mechanism. This statistic is
           incremented for packets recognized as DP packets, which are dropped.";
      }

      leaf tx_cp_kept {
        type uint64;
        description
          "When the Tx ring filling reaches a threshold, packets are inspected
           by the Control Plane Protection mechanism. This statistic is
           incremented for packets recognized as CP packets, which are sent on
           the wire.";
      }

      leaf tx_cp_overrun {
        type uint64;
        description
          "When CPU consumption used by Control Plane Protection exceeds a
           threshold, this system is disabled. This statistic is incremented for
           each Tx packet not analyzed by Control Plane Protection due to CPU
           overload.";
      }

      leaf tx_cp_passthrough {
        type uint64;
        description
          "When Control Plane Protection is enabled, this statistic is
           incremented for each packet transmitted on a link that is not
           overloaded. These packets are sent normally.";
      }

      leaf tx_dp_drop {
        type uint64;
        description
          "When the Tx ring filling reaches a threshold, packets are inspected
           by the Control Plane Protection mechanism. This statistic is incremented
           for packets recognized as DP packets, which are dropped.";
```

(continues on next page)

(continued from previous page)

```
    }
  }

  container dpvi {
   leaf kernel_tx_handoff {
      type uint64;
      description
        "Hand off on another CPU.";
    }

    leaf kernel_tx_enqueue_fail {
      type uint64;
      description
        "Data ring entry is NULL.";
    }

    leaf kernel_tx_fail {
      type uint64;
      description
        "Cannot send dpvi message.";
    }

    leaf kernel_rx_invalid_msg {
      type uint64;
      description
        "Invalid dpvi message.";
    }

    leaf kernel_rx_fp_timeout {
      type uint64;
      description
        "No Answer from FP.";
    }

    leaf kernel_rx_fp_error {
      type uint64;
      description
        "FP returned an error.";
    }

    leaf rx_data {
      type uint64;
      description
        "Data dpvi pkt (delegated tx) received.";
```

(continues on next page)

(continued from previous page)

```
      }

      leaf rx_ctrl {
        type uint64;
        description
          "Control dpvi pkt (delegated tx) received.";
      }

      leaf rx_other {
        type uint64;
        description
          "Non-dpvi pkt (soft input) received.";
      }

      leaf tx {
        type uint64;
        description
          "Send to linux (exceptions).";
      }

      leaf mem_err {
        type uint64;
        description
          "Memory error (ex: not enough mbuf).";
      }

      leaf invalid_pkt {
        type uint64;
        description
          "Malformed dpvi packet.";
      }

      leaf unsupported_cmd {
        type uint64;
        description
          "Dpvi command not implemented.";
      }
    }

    list ports {
      key "name vrf";

      description
        "The list of fpn statistics per network interface.";
```

(continues on next page)

(continued from previous page)

```
    leaf name {
      type string;
      description
        "The interface name.";
    }

    leaf vrf {
      type string;
      description
        "The interface vrf.";
    }

    list stat {
      key name;

      description
        "The key/value list of fpn statistics for one interface.";

      leaf name {
        type string;
        description
          "The statistic name.";
      }
      leaf value {
        type uint64;
        description
          "The statistic value.";
      }
    }
  }

  list gro {
    key "name vrf";

    description
      "The GRO module statistics per network interface.";

    leaf name {
      type string;
      description
        "The interface name.";
    }
```

(continues on next page)

(continued from previous page)

```
      leaf vrf {
        type string;
        description
          "The interface vrf.";
      }

      leaf ctx_curr {
        type uint64;
        description
          "The current number of GRO contexts in processing.";
      }

      leaf ctx_flush {
        type uint64;
        description
          "The number of GRO contexts flushed before timeout.";
      }

      leaf ctx_timeout {
        type uint64;
        description
          "The number of GRO contexts flushed by timeout.";
      }

      leaf done {
        type uint64;
        description
          "The number of packets merged by GRO module.";
      }

      leaf in {
        type uint64;
        description
          "The number of packets entering GRO module.";
      }

      leaf out {
        type uint64;
        description
          "The number of packets exiting GRO module.";
      }

      leaf per_reass {
        type uint64;
```

(continues on next page)

(continued from previous page)

```
        description
          "The mean of packets per GRO reassembly.";
      }
    }
  }
}
```

fp-vxlan-stats

```
submodule fp-vxlan-stats {

  belongs-to fp {
    prefix fp;
  }
  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This submodule provides support for showing fp vxlan information.";

  revision "2018-02-06" {
    description
      "Initial revision.";
  }

  augment /sixwind-router:monitoring/fp:statistics {

    container vxlan {
      leaf VxlanDroppedHeaderTooShort {
        type uint64;
        description
          "Number of input packets dropped in VXLAN due to a VXLAN header too
           short.";
      }
      leaf VxlanDroppedIPv4NoDst {
        type uint64;
        description
          "Number of output packets dropped in IPv4 VXLAN due to a null
           destination address.";
      }
      leaf VxlanDroppedIPv6NoDst {
```

(continues on next page)

(continued from previous page)

```
        type uint64;
        description
          "Number of output packets dropped in IPv6 VXLAN due to a null
           destination address.";
      }
      leaf VxlanDroppedInOperative {
        type uint64;
        description
          "Number of input packets dropped in VXLAN because the incoming
           interface is down.";
      }
      leaf VxlanDroppedInvalidIPv4Csum {
        type uint64;
        description
          "Number of input packets dropped in IPv4 VXLAN due to an invalid
           checksum.";
      }
      leaf VxlanDroppedInvalidIPv4Header {
        type uint64;
        description
          "Number of input packets dropped in IPv4 VXLAN due to a failure to get
           the VXLAN header.";
      }
      leaf VxlanDroppedInvalidIPv6Csum {
        type uint64;
        description
          "Number of input packets dropped in IPv4 VXLAN due to an invalid
           checksum.";
      }
      leaf VxlanDroppedInvalidIPv6Header {
        type uint64;
        description
          "Number of input packets dropped in IPv6 VXLAN due to a failure to get
           the VXLAN header.";
      }
      leaf VxlanDroppedInvalidIpFamily {
        type uint64;
        description
          "Number of output packets dropped in VXLAN due to a failure to get the
           VXLAN header.";
      }
      leaf VxlanDroppedOvsNoDst {
        type uint64;
        description
```

(continues on next page)

(continued from previous page)

```
          "Number of output packets dropped in OVS VXLAN due to a null
           destination address.";
      }
      leaf VxlanDroppedPrependIPv4Failure {
        type uint64;
        description
          "Number of output packets dropped in IPv4 VXLAN due to add IP header.";
      }
      leaf VxlanDroppedPrependIPv6Failure {
        type uint64;
        description
          "Number of output packets dropped in IPv6 VXLAN due to add IP header.";
      }
      leaf VxlanDroppedPrependOvsFailure {
        type uint64;
        description
          "Number of output packets dropped in OVS VXLAN due to add IP header.";
      }
      leaf VxlanDroppedUnknownIface {
        type uint64;
        description
          "Number of input packets dropped in VXLAN due to an invalid interface.";
      }
      leaf VxlanDroppedUnknownVNI {
        type uint64;
        description
          "Number of input packets dropped in VXLAN due to an invalid VNI.";
      }
      leaf VxlanExceptionIFlagNotSet {
        type uint64;
        description
          "Number of input packets sent to exception by VXLAN due a I flags not
           set (see rfc 7348).";
      }
      leaf VxlanExceptionIPv4MtuExceeded {
        type uint64;
        description
          "Number of output packets sent to exception by IPv4 VXLAN due a MTU
           exceeded the authorized value.";
      }
      leaf VxlanExceptionIPv4NoMcastSrc {
        type uint64;
        description
          "Number of output packets sent to exception by IPv4 VXLAN due to no
```

(continues on next page)

(continued from previous page)

```
          valid src address found for a multicast packet.";
      }
      leaf VxlanExceptionIPv4Route {
        type uint64;
        description
          "Number of output packets sent to exception by IPv4 VXLAN due to
           specific route.";
      }
      leaf VxlanExceptionIPv6MtuExceeded {
        type uint64;
        description
          "Number of output packets sent to exception by IPv6 VXLAN due a MTU
           exceeded the authorized value.";
      }
      leaf VxlanExceptionIPv6NoMcastSrc {
        type uint64;
        description
          "Number of output packets sent to exception by IPv6 VXLAN due to no
           valid src address found for a multicast packet.";
      }
      leaf VxlanExceptionIPv6Route {
        type uint64;
        description
          "Number of output packets sent to exception by IPv6 VXLAN due to
           specific route.";
      }
      leaf VxlanExceptionNoInputFdb {
        type uint64;
        description
          "Number of input packets sent to exception by VXLAN due to no valid
           fdb found.";
      }
      leaf VxlanExceptionNoOutputFdb {
        type uint64;
        description
          "Number of output packets sent to exception by VXLAN due to no valid
           fdb found.";
      }
      leaf VxlanExceptionNoRemote {
        type uint64;
        description
          "Number of output packets sent to exception by VXLAN due to no remote
           found.";
      }
```

(continues on next page)

(continued from previous page)

```
      leaf VxlanExceptionOvsMtuExceeded {
        type uint64;
        description
          "Number of output packets sent to exception by Ovs VXLAN due a MTU
           exceeded the authorized value.";
      }
      leaf VxlanExceptionOvsRoute {
        type uint64;
        description
          "Number of output packets sent to exception by Ovs VXLAN due to
           specific route.";
      }
      leaf VxlanExceptionTooManyFlags {
        type uint64;
        description
          "Number of input packets sent to exception by VXLAN due to a presence
           of an unsupported flag (neither I and G ones, see rfc 7348).";
      }
      leaf VxlanFdbForwDuplicateError {
        type uint64;
        description
          "Number of failure to duplicate a packet for fdb forwarding.";
      }
    }
  }
}
```

network

```
module network {
  namespace "urn:6wind:router:monitoring:network";
  prefix network;

  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This module provides support for showing network information.";

  revision "2017-11-14" {
    description "Initial revision.";
```

(continues on next page)

(continued from previous page)

```
}

augment /sixwind-router:monitoring {
  container interfaces {

    list hardware-information {
      key "name vrf";

      description
        "The list of network interface hardware information.";

      leaf name {
        type string;
        description
          "The interface name.";
      }

      leaf vrf {
        type string;
        description
          "The interface vrf.";
      }

      leaf driver {
        type string;
        description
          "The interface driver used.";
      }

      leaf description {
        type string;
        description
          "The user-defined interface alias.";
      }

      leaf model {
        type string;
        description
          "The interface model.";
      }

      leaf pci-address {
        type string;
        description
```

(continues on next page)

(continued from previous page)

```
            "The interface pci address.";
        }

        leaf speed {
          type uint64;
          description
            "The interface capacity in bytes.";
        }
      }

      list traffic-stats {
        key "name vrf";

        description
          "The list of traffic statistics per network interface.";

        leaf name {
          type string;
          description
            "The interface name.";
        }
        leaf vrf {
          type string;
          description
            "The interface vrf.";
        }
        leaf bytes-sent {
          type uint64;
          description
            "Number of bytes sent.";
        }
        leaf bytes-recv {
          type uint64;
          description
            "Number of bytes received.";
        }
        leaf pkts-sent {
          type uint64;
          description
            "Number of packets sent.";
        }
        leaf pkts-recv {
          type uint64;
          description
```

(continues on next page)

(continued from previous page)

```
            "Number of packets received.";
        }
      }

      list eth-stats {
        key "name vrf";

        description
          "The list of eth statistics per network interface.";

        leaf name {
          type string;
          description
            "The interface name.";
        }
        leaf vrf {
          type string;
          description
            "The interface vrf.";
        }

        list stat {
          key name;

          description
            "The key/value list of eth statistics for one interface.";

          leaf name {
            type string;
            description
              "The statistic name.";
          }
          leaf value {
            type uint64;
            description
              "The statistic value.";
          }
        }
      }
    }
  }
}
```

product

```
module product {
  namespace "urn:6wind:router:monitoring:product";
  prefix product;

  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This module provides support for showing product information.";

  revision "2017-11-14" {
    description
      "Initial revision.";
  }

  augment /sixwind-router:monitoring {

    leaf version {
      type string;
      description
        "The product version.";
    }

    container license {
      description
        "The license detailed info.";

      leaf enabled {
        type boolean;
        description
          "True if the license daemon is enabled on the system.";
      }

      leaf valid {
        type boolean;
        description
          "True if the license is valid.";
      }

      leaf short-license-type {
        type enumeration {
```

(continues on next page)

(continued from previous page)

```
        enum evaluation {
          description
            "The license is an evaluation.";
        }
        enum perpetual {
          description
            "The license is perpetual.";
        }
        enum subscription {
          description
            "The license is a subscription.";
        }
      }
      description
        "A shorter version of the license type.";
    }

    leaf support-end-date {
      type string;
      description
        "The support end date.";
    }

    leaf remaining-days {
      type union {
        type string;
        type enumeration {
          enum unset {
            description
              "The end date is not set.";
          }
          enum expired {
            description
              "The date has expired.";
          }
        }
      }
      description
        "The number of days remaining.";
    }

    leaf throughput-allowed {
      description
        "The allowed throughput.";
```

(continues on next page)

(continued from previous page)

```
        type decimal64 {
          fraction-digits 2;
        }
        units "Gb";
      }

      leaf throughput-used {
        description
          "The throughput currently in use.";
        type decimal64 {
          fraction-digits 2;
        }
        units "Gb";
      }

      leaf cgnat-conntracks-allowed {
        type uint32;
        description
          "The number of CG-NAT conntracks allowed.";
      }

      leaf cgnat-conntracks-used {
        type uint32;
        description
          "The number of CG-NAT conntracks currently in use.";
      }

      leaf ipsec-tunnels-allowed {
        type uint32;
        description
          "The number of IPsec tunnels allowed.";
      }

      leaf ipsec-tunnels-used {
        type uint32;
        description
          "The number of IPsec tunnels currently in use.";
      }
    }
  }
}
```

system

```
module system {
  namespace "urn:6wind:router:monitoring:system";
  prefix system;

  import sixwind-router {
    prefix sixwind-router;
  }

  description
    "This module provides support for showing system information.";

  revision "2017-11-14" {
    description
      "Initial revision.";
  }

  augment /sixwind-router:monitoring {

    list cpu-usage {
      key cpu;

      description
        "The list of busy percentage per CPU.";

      leaf cpu {
        type string;
        description
          "The CPU number.";
      }
      leaf busy {
        type uint16;
        description
          "The busy percentage.";
      }
    }

    list disk-usage {
      key disk;

      description
        "The disk usage information per device.";
```

(continues on next page)

(continued from previous page)

```
    leaf disk {
      type string;
      description
        "The disk name.";
    }

    leaf total {
      type uint64;
      description
        "The disk total size.";
    }

    leaf free {
      type uint64;
      description
        "The disk free size.";
    }
  }

  container memory {
    description
      "The total and available memory.";

    leaf available {
      type uint64;
      description
        "The memory that can be given instantly to processes without the system
         going into swap.";
    }

    leaf total {
      type uint64;
      description
        "The total physical memory.";
    }
  }

  list numa-stats {
    key node;

    description
      "The list of NUMA statistics per node.";

    leaf node {
```

(continues on next page)

(continued from previous page)

```
        type string;
        description
          "The node ID.";
      }
      leaf numa-hit {
        type uint64;
        description
          "Memory successfully allocated on this node as intended.";
      }
      leaf numa-miss {
        type uint64;
        description
          "Memory allocated on this node despite the process preferring some
           different node.";
      }
      leaf numa-foreign {
        type uint64;
        description
          "Memory intended for this node but actually allocated on some different
           node.";
      }
      leaf interleave-hit {
        type uint64;
        description
          "Interleaved memory successfully allocated on this node as intended.";
      }
      leaf local-node {
        type uint64;
        description
          "Memory allocated on this node while a process was running on it.";
      }
      leaf other-node {
        type uint64;
        description
          "Memory allocated on this node while a process was running on some
           other node.";
      }
    }

    list processes {
      key name;

      description
        "The list of monitored processes on the system.";
```

(continues on next page)

(continued from previous page)

```
    leaf name {
      type string;
      description
        "The name of the process.";
    }

    leaf fds {
      type uint32;
      description
        "The number of file descriptors opened by this process.";
    }

    leaf memory {
      type uint64;
      description
        "The memory used by this process.";
    }

    leaf busy {
      type uint16;
      description
        "The busy percentage for this process.";
    }
  }

  leaf uptime {
    type string;
    description
      "The system uptime.";
  }

  leaf user-count {
    type uint16;
    description
      "The number of logged users.";
  }

  list users {
    key terminal;

    description
      "The list of logged users on the system.";
```

(continues on next page)

(continued from previous page)

```
      leaf terminal {
        type string;
        description
          "The terminal name.";
      }

      leaf user {
        type string;
        description
          "The user.";
      }

      leaf source {
        type string;
        description
          "The host from where the user logged in.";
      }

      leaf started {
        type string;
        description
          "The date at which the connection was started.";
      }
    }

    list soft-interrupts-stats {
      key cpu;

      description
        "The list of soft interrupts statistics per CPU.";

      leaf cpu {
        type string;
        description
          "The CPU number.";
      }
      leaf hi {
        type uint64;
        description
          "Number of high priority tasklets.";
      }
      leaf timer {
        type uint64;
        description
```

(continues on next page)

(continued from previous page)

```
          "Number of timer interrupts.";
      }
      leaf net-tx {
        type uint64;
        description
          "Number of interrupts for transmitting packets to network cards.";
      }
      leaf net-rx {
        type uint64;
        description
          "Number of interrupts for receiving packets from network cards.";
      }
      leaf block {
        type uint64;
        description
          "Number of block-device I/O interrupts.";
      }
      leaf tasklet {
        type uint64;
        description
          "Number of regular tasklets interrupts.";
      }
      leaf sched {
        type uint64;
        description
          "Number of scheduling interrupts.";
      }
      leaf rcu {
        type uint64;
        description
          "Number of read-copy-update interrupts.";
      }
    }
  }
}
```

3. Troubleshooting

This guide references common configuration issues one may encounter when using Virtual Accelerator, and indications on how to address them. These indications suppose you are logged as root and have access to the Linux shell.

3.1 Relevant Information for Bug Reporting

In case you cannot investigate and resolve the issue by yourself using this document, make sure you open a ticket on your 6WIND Customer Zone with the relevant troubleshooting information.

This information can be generated and exported using the `troubleshooting-report.sh` script.

Note: If you are in an OpenStack environment, call the script providing as arguments:

- `--controller` inside a controller node
- `--compute` inside a compute node
- `--network` inside a network node

Also make sure credentials for access to nova are exported in the environment (i.e.: `OS_USERNAME`, `OS_PASSWORD`, `OS_PROJECT_DOMAIN_ID`, etc.).

If possible, reproduce your issue with debug info enabled. Enable debug for the shortest time possible, as it produces a flabbergasting amount of log.

To enable debug, look at the *OpenStack logs* section.

We recommend installing the `sos` package before calling the following script, to leverage the distribution bug reporting mechanism (`sosreport`). Once installed, `troubleshooting-report.sh` will detect it and run it.

`troubleshooting-report.sh` is provided here for information:

```
#!/bin/sh
#
# Copyright 2016 6WIND S.A.

cleanup()
{
        [ -d "$TMPDIR" ] && rm -rf $TMPDIR
}
```

(continues on next page)

(continued from previous page)

```
usage_option()
{
        printf "\t%s" "$1"
        [ -n "$3" ] && printf " %s" "$3"
        printf "\t\t%s\n" "$2"
}

usage()
{
        printf "%s\n\n" "$0 [-h|--help] [-c|--compute] [-C|--controller] [-N|--
↪network] [-f|--file <file path>] [-o|--no-core-file] [-O|--clean-core-files] [-e|--
↪extra <file path>]"
        usage_option -h "display this help"
        usage_option -c "Use this for OpenStack compute nodes"
        usage_option -C "Use this for OpenStack controller nodes"
        usage_option -N "Use this for OpenStack network nodes"
        usage_option -f "Output file path"
        usage_option -o "Do not archive core files"
        usage_option -O "Cleanup core files after generating the report"
        usage_option -e "Add an external file in the report archive"
}

parse_args()
{
        # Turn long options into short ones
        for arg in "$@"; do
                shift
                case "$arg" in
                "--help")        set -- "$@" "-h" ;;
                "--controller")  set -- "$@" "-C" ;;
                "--compute")     set -- "$@" "-c" ;;
                "--network")     set -- "$@" "-N" ;;
                "--file")        set -- "$@" "-f" ;;
                "--no-core-file")set -- "$@" "-o" ;;
                "--clean-core-files")set -- "$@" "-O" ;;
                "--extra")       set -- "$@" "-e" ;;
                *)               set -- "$@" "$arg"
                esac
        done

        OPTIND=1
        while getopts hcCNf:oOe: name
        do
```

(continues on next page)

(continued from previous page)

```
                case "$name" in
                c) COMPUTE="yes" && OPENSTACK="yes" && \
                        SUFFIX="${SUFFIX}_compute" ;;
                C) CONTROLLER="yes" && OPENSTACK="yes" && \
                        SUFFIX="${SUFFIX}_controller" ;;
                N) NETWORK="yes" && OPENSTACK="yes" && \
                        SUFFIX="${SUFFIX}_network" ;;
                f) ARCHIVE=${OPTARG} ;;
                o) NO_CORE_FILE="yes" ;;
                O) CLEAN_CORE_FILES="yes" ;;
                e) EXTRAFILES="${OPTARG} ${EXTRAFILES}" ;;
                h) usage && exit 0 ;;
                *) usage && exit 1 ;;
                esac
        done
}

# Check whether the given command is executable, and if the timeout utility is
# available, run the command using a $TIMEOUT_S seconds timeout.
exec_cmd()
{
        local netns_cmd="$(echo $@ | sed -n 's/^ip netns exec [^ ]\+ \([^ ]\+\).*$/\1/p
↪')"
        # $cmd contains the command to check for without arguments,
        # stripped from any leading `ip netns exec XXX`.
        cmd=${netns_cmd:-$1}

        [ -x "$(command -v $cmd)" ] && $TIMEOUT $@
}

get_linux_info()
{
        local ns= name=$1
        if [ -n "$name" ]; then
                ns="ip netns exec $name"
        else
                name=main
        fi
        # information on all known links
        for iface in $($ns ip link show | sed -n 's/^[^:]\+: \([^:]\+\):.*$/\1/p'); do
                $ns fp-cli dpdk-port-stats $iface > $BUGDIR/$name-fp-cli_dpdk-port-
↪stats_$iface.txt 2>&1
                $ns fp-cli dpdk-port-offload $iface > $BUGDIR/$name-fp-cli_dpdk-port-
↪offload_$iface.txt 2>&1
```

(continues on next page)

(continued from previous page)

```
                $ns fp-cli dpdk-port-advertise $iface > $BUGDIR/$name-fp-cli_dpdk-port-
↪advertise_$iface.txt 2>&1
        done
        # interfaces, addresses, routes, neighbours and IPsec
        $ns ip -detail -statistics link > $BUGDIR/$name-ip_link.txt
        $ns ip -detail address > $BUGDIR/$name-ip_address.txt
        $ns ip -detail route > $BUGDIR/$name-ip_route.txt
        $ns ip -6 -detail route > $BUGDIR/$name-ip_route6.txt
        $ns ip neigh > $BUGDIR/$name-ip_neigh.txt
        $ns ip -6 neigh > $BUGDIR/$name-ip_neigh6.txt
        $ns ip -statistics xfrm policy > $BUGDIR/$name-ip_xfrm_policy.txt
        $ns ip -statistics xfrm state > $BUGDIR/$name-ip_xfrm_state.txt
        # get fpn0 stats
        if [ "$name" = "vrf0" ]; then
                fpexec ethtool -S fpn0 > $BUGDIR/$name-ethtool_S_fpn0.txt 2>&1
        fi
        # active network connections
        $ns netstat -anp > $BUGDIR/$name-nestat_anp.txt
        # Netfilter
        exec_cmd $ns iptables-save > $BUGDIR/$name-iptables_save.txt
        exec_cmd $ns ip6tables-save > $BUGDIR/$name-ip6tables_save.txt
        exec_cmd $ns ebtables-save > $BUGDIR/$name-ebtables_save.txt
        exec_cmd $ns ipset save > $BUGDIR/$name-ipset_save.txt
        exec_cmd $ns ipset list > $BUGDIR/$name-ipset_list.txt
        # Check bridge info
        exec_cmd $ns brctl show > $BUGDIR/$name-brctl_show.txt
        exec_cmd $ns ovs-vsctl show > $BUGDIR/$name-ovs_vsctl_show.txt 2>&1
}

get_system_info()
{
        # system topology
        exec_cmd lstopo-no-graphics --output-format xml > $BUGDIR/lstopo.xml
        # processors hierarchy
        cp /proc/cpuinfo $BUGDIR/cpuinfo.txt
        exec_cmd lscpu > $BUGDIR/lscpu.txt
        # interrupts
        cp /proc/interrupts $BUGDIR/interrupts.txt
        # memory record
        cp /proc/meminfo $BUGDIR/meminfo.txt
        exec_cmd vmstat -ws > $BUGDIR/vmstat_ws.txt
        exec_cmd numastat -zm > $BUGDIR/numastat_zm.txt
        exec_cmd numastat -zs > $BUGDIR/numastat_zs.txt
        # PCI peripherals
```

(continues on next page)

(continued from previous page)

```
        exec_cmd lspci -vvv > $BUGDIR/lspci.txt
        # DMI/SMBIOS
        exec_cmd dmidecode > $BUGDIR/dmidecode.txt

        # kernel version, logs, cmdline and loaded modules
        uname -a > $BUGDIR/uname.txt
        dmesg > $BUGDIR/dmesg.txt
        cp /proc/cmdline $BUGDIR/cmdline.txt
        lsmod > $BUGDIR/lsmod.txt

        # distribution
        exec_cmd lsb_release -a > $BUGDIR/lsb_release.txt 2>&1

        # services list
        exec_cmd service --status-all > $BUGDIR/service_status_all.txt 2>&1
        exec_cmd systemctl list-units > $BUGDIR/systemctl_list_units.txt 2>&1

        # logs
        exec_cmd journalctl --all --no-pager --this-boot > $BUGDIR/journal.txt
        [ -f "/var/log/syslog" ] && cp /var/log/syslog $BUGDIR/syslog.txt
        [ -d "/var/log/libvirt" ] && cp -r /var/log/libvirt $BUGDIR/libvirt
        [ -f "/var/log/netlimits.log" ] && cp /var/log/netlimits.log \
                $BUGDIR/6WIND-vRouter-license-usage-report.txt
        [ -f "/var/log/netlimits.log.sig" ] && cp /var/log/netlimits.log.sig \
                $BUGDIR/6WIND-vRouter-license-usage-report.txt.sig

        # processes list
        ps auxww > $BUGDIR/ps_auxww.txt
        # cpuset
        [ -d "/dev/cpuset" ] && cp -r /dev/cpuset $BUGDIR/cpuset 2>/dev/null
        # /dev
        ls -al /dev > $BUGDIR/ls_al_dev.txt
        # IRQ affinity
        find /proc/irq -maxdepth 1 -mindepth 1 -print -type d -exec \
                cat '{}/smp_affinity' \; > $BUGDIR/proc_irq_smp_affinity.txt

        # mounted partitions
        exec_cmd mount > $BUGDIR/mount.txt

        # VNB
        exec_cmd ngctl list > $BUGDIR/ngctl_list.txt 2>&1

        exec_cmd vrfctl list > $BUGDIR/vrf_list.txt 2>&1
}
```

(continues on next page)

(continued from previous page)

```
get_coredump()
{
        if [ "$NO_CORE_FILE" != 'yes' ] && ls /var/lib/systemd/coredump/core* >/dev/
↪null 2>&1
        then
                mkdir $BUGDIR/coredump
                cp /var/lib/systemd/coredump/core* $BUGDIR/coredump/
                coredumpctl info > $BUGDIR/coredump/coredumpctl_info.txt
                if [ "$CLEAN_CORE_FILES" = 'yes' ]; then
                        rm -f /var/lib/systemd/coredump/core*
                fi
        fi
}

get_fp_info()
{
        if [ -x "$(command -v fast-path.sh)" ]; then
                # Record your fp configuration:
                exec_cmd fp-conf-tool -DSFv > $BUGDIR/fp_config.txt \
                        2>$BUGDIR/fp_config.log

                # Record what 6windgate version is used
                if [ -x "$(command -v dpkg)" ]; then
                        exec_cmd dpkg -s 6windgate-fp > $BUGDIR/fp_version.txt 2>/dev/
↪null
                else
                        exec_cmd yum info 6windgate-fp > $BUGDIR/fp_version.txt 2>/dev/
↪null
                fi

                # in buildroot, /etc/issue contains 6WG version
                [ -f "/etc/issue" ] && cp /etc/issue $BUGDIR/etc_issue.txt

                # Copy fp logs
                [ -f "/var/log/fast-path.log" ] && \
                        cp /var/log/fast-path.log $BUGDIR/fast-path.log
                [ -f "/var/log/messages" ] && \
                        cp /var/log/messages $BUGDIR/messages.log

                exec_cmd linux-fp-sync.sh status > $BUGDIR/linux_fp_sync_status.txt
                fast-path.sh status > $BUGDIR/fast_path_status.txt
                if grep -q "fp-.*not running" "$BUGDIR/fast_path_status.txt"; then
                        FP_RUNNING="false"
```

(continues on next page)

(continued from previous page)

```
                else
                        FP_RUNNING="true"
                fi
        fi

        # Copy fp info
        if [ "$FP_RUNNING" = "true" ]; then
                # compilation options used for your FP:
                $TIMEOUT fp-cli conf compiled > $BUGDIR/fp_compil_options.txt 2>&1
                $TIMEOUT fp-cli stats percore > $BUGDIR/fp_cli_dump_stats.txt
                $TIMEOUT fp-cli fp-state > $BUGDIR/fp_cli_fp_state.txt
                $TIMEOUT fp-cli filling > $BUGDIR/fp_cli_filling.txt
                if [ -n "$(fp-cli conf compiled | grep CONFIG_MCORE_ARCH_DPDK=y)" ];␣
↪then
                        $TIMEOUT fp-cli dpdk-debug-pool | grep common_pool_count >
↪$BUGDIR/fp_cli_dpdk_pool.txt
                fi

                $TIMEOUT fp-cli iface > $BUGDIR/fp_cli_iface.txt
                $TIMEOUT fp-cli neigh4 > $BUGDIR/fp_cli_neigh4.txt
                $TIMEOUT fp-cli neigh6 > $BUGDIR/fp_cli_neigh6.txt
                $TIMEOUT fp-cli route4 > $BUGDIR/fp_cli_route4.txt
                $TIMEOUT fp-cli route6 > $BUGDIR/fp_cli_route6.txt
                $TIMEOUT fp-cli crypto-lib > $BUGDIR/fp_cli_crypto_lib.txt
                for cryptolib in $(cat $BUGDIR/fp_cli_crypto_lib.txt | grep -v
↪"Available crypto"); do
                        $TIMEOUT fp-cli stats-crypto $cryptolib > $BUGDIR/fp_cli_stats_
↪crypto_$cryptolib.txt
                done
                $TIMEOUT fp-cli crypto-offload-stats > $BUGDIR/fp_cli_crypto_offload_
↪stats.txt
                # bridge
                $TIMEOUT fp-cli bridge > $BUGDIR/fp_cli_bridge.txt
                $TIMEOUT fp-cli fp-vswitch-flows > $BUGDIR/fp_cli_fpvs_flows.txt
                $TIMEOUT fp-cli fp-vswitch-port > $BUGDIR/fp_cli_fpvs_ports.txt
                $TIMEOUT fp-cli fp-vswitch-stats > $BUGDIR/fp_cli_fpvs_stats.txt
                $TIMEOUT fp-cli fp-vswitch-stats > $BUGDIR/fp_cli_fpvs_stats.txt
                # Netfilter
                $TIMEOUT fp-cli nf4-table filter all > $BUGDIR/fp_cli_nf4_table.txt
                $TIMEOUT fp-cli nf4-table mangle all >> $BUGDIR/fp_cli_nf4_table.txt
                $TIMEOUT fp-cli nf4-table nat all >> $BUGDIR/fp_cli_nf4_table.txt
                $TIMEOUT fp-cli nf4-rules filter > $BUGDIR/fp_cli_nf4_rules.txt
                $TIMEOUT fp-cli nf4-rules mangle >> $BUGDIR/fp_cli_nf4_rules.txt
                $TIMEOUT fp-cli nf4-rules nat >> $BUGDIR/fp_cli_nf4_rules.txt
```

(continues on next page)

(continued from previous page)

```
                $TIMEOUT fp-cli nf6-table filter all > $BUGDIR/fp_cli_nf6_table.txt
                $TIMEOUT fp-cli nf6-table mangle all >> $BUGDIR/fp_cli_nf6_table.txt
                $TIMEOUT fp-cli nf6-rules filter > $BUGDIR/fp_cli_nf6_rules.txt
                $TIMEOUT fp-cli nf6-rules mangle >> $BUGDIR/fp_cli_nf6_rules.txt
                $TIMEOUT fp-cli filter-bridge broute all > $BUGDIR/fp_cli_ebtables.txt
                $TIMEOUT fp-cli filter-bridge filter all >> $BUGDIR/fp_cli_ebtables.txt
                $TIMEOUT fp-cli vrf-exec all nf-ipset > $BUGDIR/fp_cli_nf_ipset.txt
                # IPsec
                $TIMEOUT fp-cli ipsec4-spd all > $BUGDIR/fp_cli_ipsec4_spd_all.txt
                $TIMEOUT fp-cli ipsec4-sad all > $BUGDIR/fp_cli_ipsec4_sad_all.txt
                $TIMEOUT fp-cli ipsec6-spd all > $BUGDIR/fp_cli_ipsec6_spd_all.txt
                $TIMEOUT fp-cli ipsec6-sad all > $BUGDIR/fp_cli_ipsec6_sad_all.txt
                # License
                if [ -n "$(fp-cli conf compiled | grep CONFIG_MCORE_LICENSE=y)" ];␣
↪then
                    $TIMEOUT fp-cli license > $BUGDIR/fp_cli_license.txt
                fi

                # FP-VNB
                exec_cmd fpngctl list > $BUGDIR/fpngctl_list.txt
                # shmem-ports
                exec_cmd fp-shmem-ports -d -e all > $BUGDIR/fp_shmem_ports_d.txt
                exec_cmd fp-shmem-ports -s -e all > $BUGDIR/fp_shmem_ports_s.txt

                if [ -n "$(fp-cli conf compiled | grep CONFIG_MCORE_DPVI=y)" ]; then
                        exec_cmd fp-shmem-dpvi > $BUGDIR/fp_shmem_dpvi.txt
                fi

                exec_cmd fp-cpu-usage > $BUGDIR/fp_cpu_usage.txt

                # dpvi
                [ -f "/proc/sys/dpvi_shmem/list_shm" ] && \
                        cp /proc/sys/dpvi_shmem/list_shm $BUGDIR/list_shm.txt
                [ -f "/proc/sys/dpvi/list_interfaces" ] && \
                        cp /proc/sys/dpvi/list_interfaces $BUGDIR/dpvi_list_interfaces.
↪txt
                [ -f "/proc/sys/dpvi/running_fastpath" ] && \
                        cp /proc/sys/dpvi/running_fastpath $BUGDIR/dpvi_running_
↪fastpath.txt

                exec_cmd fp-shmem-dpvi > $BUGDIR/fp_shmem_dpvi.txt

                # npf
                if [ -x "$(command -v fp-npfctl)" ]; then
```

(continues on next page)

(continued from previous page)

```
                        exec_cmd fp-npfctl vrf-exec all show > $BUGDIR/fp_npf_conf.txt␣
↪2>&1
                        exec_cmd fp-npfctl stats > $BUGDIR/fp_npf_stats.txt  2>&1
                        exec_cmd fp-npfctl show-params > $BUGDIR/fp_npf_params.txt  2>&
↪1
                        exec_cmd fp-npfctl pool-usage > $BUGDIR/fp_npf_pool_usage.txt␣
↪2>&1
                        exec_cmd fp-npfctl htable-stats > $BUGDIR/fp_npf_htable_stats.
↪txt 2>&1
                        exec_cmd fp-npfctl vrf-exec all cgnat all conntrack-stats >
↪$BUGDIR/fp_cgnat_conn_stats.txt 2>&1
                        exec_cmd fp-npfctl vrf-exec all cgnat all block-stats >
↪$BUGDIR/fp_cgnat_block_stats.txt 2>&1
                        exec_cmd fp-npfctl vrf-exec all cgnat all block-stats >
↪$BUGDIR/fp_cgnat_port_stats.txt 2>&1
                        exec_cmd fp-npfctl vrf-exec all cgnat all ip-stats > $BUGDIR/
↪fp_cgnat_ip_stats.txt 2>&1
                fi
        fi
}

get_openstack_info()
{
        # Try setting Openstack if the user forgot telling it in arguments.
        # Only working when Openstack processes are still running.
        if [ -n "$(ps auxww | grep nova-compute | grep -v 'grep')" ]; then
                COMPUTE=yes
                OPENSTACK=yes
        fi
        if [ -n "$(ps auxww | grep nova-api | grep -v 'grep')" ]; then
                CONTROLLER=yes
                OPENSTACK=yes
        fi
        if [ -n "$(ps auxww | grep neutron-server | grep -v 'grep')" ]; then
                NETWORK=yes
                OPENSTACK=yes
        fi

        [ "$OPENSTACK" != 'yes' ] && return

        [ -d "/etc/nova" ] && cp -r /etc/nova $BUGDIR/etc_nova
        [ -d "/etc/neutron" ] && cp -r /etc/neutron $BUGDIR/etc_neutron
        [ -d "/var/log/nova" ] && cp -r /var/log/nova $BUGDIR/log_nova
        [ -d "/var/log/neutron" ] && cp -r /var/log/neutron $BUGDIR/log_neutron
```

(continues on next page)

(continued from previous page)

```
        if [ "$CONTROLLER" = 'yes' ]; then
                # Check whether OpenStack credentials are correctly set
                failure=$($TIMEOUT nova list 2>&1)
                if [ "$?" -ne "0" ]; then
                        printf "\`nova list\` failed with this error: ${failure}\n"
                        printf "Make sure your credentials are properly exported, \
                                and re-run the script.\n"
                        return
                fi

                $TIMEOUT nova --version 2> $BUGDIR/nova_version.txt
                $TIMEOUT nova list > $BUGDIR/nova_list.txt
                $TIMEOUT nova host-list > $BUGDIR/nova_host_list.txt
                $TIMEOUT nova hypervisor-list > $BUGDIR/nova_hypervisor_list.txt
                $TIMEOUT nova flavor-list > $BUGDIR/nova_flavor_list.txt
                $TIMEOUT nova image-list > $BUGDIR/nova_image_list.txt
                $TIMEOUT nova network-list > $BUGDIR/nova_network_list.txt
                $TIMEOUT nova service-list > $BUGDIR/nova_service_list.txt
                for id in $($TIMEOUT nova list --minimal | tail -n +4 | awk '{ print
↪$2 }'); do
                        [ -n '${id}' ] &&
                        printf "##### ${id} #####\n" >> $BUGDIR/nova_show.txt &&
                        $TIMEOUT nova show "${id}" >> $BUGDIR/nova_show.txt
                done
        fi

        if [ "$NETWORK" = 'yes' ]; then
                $TIMEOUT neutron agent-list > $BUGDIR/neutron_agent_list.txt
                $TIMEOUT neutron router-list > $BUGDIR/neutron_router_list.txt
                $TIMEOUT neutron net-list > $BUGDIR/neutron_net_list.txt
                $TIMEOUT neutron subnet-list > $BUGDIR/neutron_net_list.txt
        fi
}

get_management_info()
{
        if [ -d /etc/sysrepo ]; then
                cp -a /etc/sysrepo $BUGDIR/etc_sysrepo
                exec_cmd sysrepocfg -X$BUGDIR/vrouter_startup.json \
                        -m vrouter -d startup -f json
                exec_cmd sysrepocfg -X$BUGDIR/vrouter_running.json \
                        -m vrouter -d running -f json
        fi
```

(continues on next page)

(continued from previous page)

```
}

get_license_info()
{
        if command -v vrl-status > /dev/null; then
                vrl-status > $BUGDIR/vrl_status.txt
        fi
}

trap cleanup EXIT INT QUIT

TMPDIR=$(mktemp -d)
SUFFIX=$(hostname)
TIMEOUT_S=10
BUGDIR=$TMPDIR/bug_info/$SUFFIX
EXTRAFILES=""
mkdir -p $BUGDIR
[ -x "$(command -v timeout)" ] && TIMEOUT="$(command -v timeout) ${TIMEOUT_S}s"

parse_args $@

ARCHIVE=${ARCHIVE:-"/tmp/troubleshooting-report_${SUFFIX}.tar.gz"}

printf 'Gathering information. This may take some time...\n'

# octeon-specific info
[ -f "/proc/octeon_ethernet_stats" ] && \
        cp /proc/octeon_ethernet_stats > $BUGDIR/octeon_ethernet_stats.txt
[ -f "/proc/octeon_info" ] && cp /proc/octeon_info > $BUGDIR/octeon_info.txt

if [ -n "$(ip netns)" ]; then
        for vrf in $(ip netns | cut -d " " -f 1); do
                get_linux_info $vrf
        done
fi
get_linux_info

# va-specefic files [SF13646] [SF13686]
[ -f "/tmp/conf.old" ] && cp /tmp/conf.old $BUGDIR/conf.old
[ -f "/tmp/conf.json" ] && cp /tmp/conf.json $BUGDIR/conf.json

get_system_info
get_coredump
get_fp_info
```

(continues on next page)

(continued from previous page)

```
get_openstack_info
get_management_info
get_license_info

##### External SOSREPORT results
if [ -x "$(command -v sosreport)" ]; then
        SOSDIR="$BUGDIR/sosreport"
        mkdir -p $SOSDIR
        sosreport -a --batch --build --tmp-dir=$SOSDIR > $BUGDIR/sosreport.txt
fi

# Add extra files
for file in ${EXTRAFILES}; do
        cp ${file} $BUGDIR/`basename ${file}`
done

tar -czf ${ARCHIVE} -C ${TMPDIR} .
printf "Saved into ${ARCHIVE}\n"
```

Send the `troubleshooting-report*.tar.gz` files along with any other information you deem relevant.

Note: The script makes use of the following commands if available:

`brctl` `dmidecode` `ebtables-save` `ip6tables-save` `iptables-save` `lsb_release` `lscpu` `lspci` `lstopo` `mount` `numastat` `ovs-vsctl` `sosreport` `timeout` `vmstat`

For better results, make sure these tools are available on your system.

3.2 Typical issues

3.2.1 Startup Issues

Virtual Accelerator cannot start

Symptoms

- `systemctl status virtual-accelerator` shows issues

Hints

- On Intel and Arm, check whether the configuration file is correct by looking at `fast-path.sh config` output for relevancy, and by checking config file syntactic correctness with `fast-path.sh config -c`. Follow the advice regarding deprecated options as it may become problematic in later versions. Take into account the WARNINGs in the output.

Refer to the *Fast Path Baseline* for further details on the wizard (`fast-path.sh config`).

- If you tried running the fast path and it crashed or failed along the way, some "runtime-only" files may be left unremoved. Make sure to call `fast-path.sh stop` before trying to start the fast path again.

  Refer to the *Fast Path Baseline* documentation for further details on `fast-path.sh` commands.

- Look for error messages either on the console or in the logs. See *rsyslog* and *journalctl* sections for details regarding what can be found in the logs.
- Executable paths may change between two Virtual Accelerator versions. Some shells (bash for example) keep a cache of the executable paths. After upgrading Virtual Accelerator, if some commands are not found, you may need to start a new shell.

Hugepages fragmentation

Symptoms

- One of the following messages appears on he console or in the logs:

```
No more huge pages left for fastpath initialization

EAL: Not enough memory available! Requested: <X>MB, available: <Y smaller␣
↪than X>MB
PANIC in rte_eal_init(): Cannot init memory

EAL: rte_eal_common_log_init(): cannot create log_history mempool
PANIC in rte_eal_init():
Cannot init logs

Not enough physically contiguous memory to allocate the mbuf pool on this␣
↪socket (0): max_seg_size=178257920, total_mem=459276288, nb_seg=35
Increase the number of huge pages, use larger huge pages, or reboot the␣
↪machine
PANIC in fpn_socket_mbufpool_create():
Cannot create mbuf pool for socket 0
```

Hints

- There is a problem with the available memory.
- Add more memory.
- Check the output from `/proc/meminfo`, especially the `MemFree` and `HugePage_Free` fields. See *meminfo* section for details.

  `MemFree` gives an indication of how much memory you may use for the fast path shared memory.

  `HugePage_Free` indicates how many huge pages are available for use by the fast path.

Beware, if hugepages are fragmented, you need to allocate more or simply reboot, as the DPDK requires contiguous physical memory.

- Check the output from `fast-path.sh config --dump --full`, optionally with the `--long` option. `FP_MEMORY` might need adjustment to reserve more hugepages, which reduces the risk of fragmentation. Refer to the *Fast Path Baseline* for further details on the wizard (`fast-path.sh config`).
- If you tried running the fast path and it crashed or failed along the way, some "runtime-only" files may be left unremoved. Make sure to call `fast-path.sh stop` before trying to start the fast path again. Refer to the *Fast Path Baseline* documentation for further details on `fast-path.sh` commands.

Not enough memory

Symptoms

- The following message appears on he console or in the logs (and subsequent commands fail with similar messages):

```
/usr/bin/fast-path.sh: 435: /usr/bin/fast-path.sh: Cannot fork
/usr/bin/fast-path.sh: 668: /usr/bin/fast-path.sh: Cannot fork
```

- The following message appears on he console or in the logs:

```
...
EAL:   PCI memory mapped at 0x7ffae4a40000
PMD: eth_em_dev_init(): port_id 2 vendorID=0x8086 deviceID=0x100e
Using fpn_port 0x7ffae654c000 size=150576 (0M)
Killed
//usr/bin/fast-path.sh: error starting //usr/bin//fp-rte. Check logs for␣
↪details.
```

At this point, the machine may have hung. Check the logs after reboot, especially if they contain something similar to:

```
...
fp-rte[5113]: Using fp_ebtables_vr_shared=0x7ffae63c2000 size=4352 (0M)
fp-rte[5113]: Using fp-tc-shared=0x7ffad976f000 size=524608 (0M)
kernel: [ 1022.485264] fp-rte invoked oom-killer: gfp_mask=0x2d2, order=0,␣
↪oom_score_adj=0
kernel: [ 1022.485271] fp-rte cpuset=/ mems_allowed=0
```

Note: Look for error messages either on the console or in the logs. See *rsyslog* and *journalctl* sections for details regarding what can be found in the logs.

Hints

- There is a problem with the available memory, the fast path process has been killed because available memory was getting too small. Typically, after hugepages allocation, the fast path tried to allocate memory and there was not enough free.
- Add more memory.
- Check the output from `/proc/meminfo`, especially the `MemFree` field. See *meminfo* section for details.

  `MemFree` estimates how much memory is free before starting the fast path.
- Check the output from `fast-path.sh config --dump --full`, optionally with the `--long` option. `FP_MEMORY` and `VM_MEMORY` might need adjustment to reserve less hugepages. Refer to the *Fast Path Baseline* for further details on the wizard (`fast-path.sh config`).

1G hugepages problems

Symptoms

- The following message appears on he console or in the logs:

```
sh: echo: I/O error
WARNING: Can not allocate 1 hugepages for fast path
         0 pages of size 1024 MB were allocated
```

Hints

- It seems you enabled the support of 1G hugepages in the kernel boot command line (`hugepagesz=1G default_hugepagesz=1G`). The fast path starting script failed to allocate the required amount of hugepages.
- Ensure that enough hugepages are reserved at boot time. Check the output from `fast-path.sh config --dump --full`, optionally with the `--long` option, to get the value of `FP_MEMORY` representing the per-socket amount of memory needed by the fast path, and reserve the hugepages accordingly.

Invalid core-port mapping

Symptoms

- The following message appears on he console or in the logs:

```
WARNING - skipping invalid cpus [X, Y, Z, ...] in CORE_PORT_MAPPING
```

Hints

- On Intel and Arm, check whether the configuration file is correct by looking at `fast-path.sh config` output for relevancy, and by checking config file syntactic correctness with `fast-path.sh config -c`. Follow the advice regarding deprecated options as it may become problematic in later versions. Take into account the WARNINGs in the output. Refer to the *Fast Path Baseline* for further details on the wizard (`fast-path.sh config`).

- Check the output from `fast-path.sh config --dump --full`, optionally with the `--long` option. Ensure consistency between the FP_MASK and CORE_PORT_MAPPING options. The cores used for polling may not be among the ones defined as the fast path core mask. As soon as the core/port mapping is manual, the fast path core mask should be manual too.

3.2.2 Networking Issues

Ports synchronization problems

Symptoms

- No ports are displayed when calling `fp-cli iface`.

Hints

- Check the fast path and its daemons status, using `fast-path.sh status`. Refer to the *Fast Path Baseline* documentation for further details on `fast-path.sh` commands. You should have at least `fpmd` running and `fp-rte` (Intel and Arm only).
- Check which interfaces are known to the `cmgrd`, using `daemonctl cmgrd show interfaces`. Refer to the *Linux - Fast Path Synchronization* documentation for further details on the cache manager.
- If you are dealing with physical NIC: Check that your NIC is detected by Linux, using `lspci`. See *lspci* section for details.
- Check the output from `fast-path.sh config --display` and make sure your NIC is among the selected ethernet cards.

  Refer to the *Fast Path Baseline* for further details on the wizard (`fast-path.sh config`).
- Check the output from `fast-path.sh config --dump --full`, optionally with the `--long` option. Refer to the *Fast Path Baseline* for further details on the wizard (`fast-path.sh config`).
- Check Linux-FP sync associated daemons status, using `linux-fp-sync.sh status`. Refer to the *Linux - Fast Path Synchronization* documentation for further details on `linux-fp-sync.sh` commands. You should have at least `cmgrd` running.

No packets are forwarded

Symptoms

- No packets are forwarded.
- `fp-cli stats non-zero` shows no (or low) IpForwDatagrams stats.
- `fp-cli dpdk-port-stats <port>` shows no (or low) rx/tx packets stats.
- `ip -s link show <interface>` shows no (or low) rx/tx packets stats.
- `kill -USR1 $(pidof fp-rte)` (Intel and Arm only) shows no (or low) rx/tx packets stats.

Hints

- Check whether configurations between Linux and the fast path are consistent:
  - Check Linux-FP sync associated daemons status, using `linux-fp-sync.sh status`. Refer to the *Linux - Fast Path Synchronization* documentation for further details on `linux-fp-sync.sh` commands.
  - Check IP addresses and routes configured in the kernel, using `ip address show` and `ip route show`. Check whether the interfaces and bridges are up and running using `ip link show` and `brctl show <bridge_name>`.
- Check IP addresses and routes known to the fast path, using `fp-cli route4 type all`.

  Refer to the *Fast Path Baseline* documentation for further details on `fp-cli` commands.
- If you are using bridges, check whether your bridges have correct states, using `fp-cli bridge`.

  Refer to the *Fast Path Baseline* documentation for further details on `fp-cli` commands.
- Check that `fp_dropped` fast path statistics are not too high using `fp-cli stats percore non-zero`. A high `fp_dropped` stat suggests that packets are somehow not acceptable for the fast path. The ideal case is when forwarding stats are evenly spread throughout cores, that is when each core more or less forwards as many packets as the others. See *Fast Path statistics* section for an example of stats.

  Refer to the *Fast Path Baseline* documentation for further details on `fp-cli` commands.
- Check that `exception stats` fast path statistics are not too high. Basic exceptions indicate how many packets could not be processed by the fast path, and have thus been injected in the linux stack for slow path processing. If the value is high, it is a good indicator that IP addresses/routes/tunnels in the fast path are badly configured. See *Fast Path statistics* section for an example of stats.

  Refer to the *Fast Path Baseline* documentation for further details on `fp-cli` commands.
- Check whether it works correctly when the fast path is turned off. See *Turn Fast Path off* section for details.

Netfilter synchronization problems

Symptoms

- Packets are not filtered according to your iptables rules.

Hints

- Check whether filtering rules between Linux and fast path are consistent:
  - Check filtering rules in the kernel, using `ip[6]tables -S`. Refer to the `ip[6]tables` manpage for details on this command.
  - Check filtering rules known to the fast path, using `fp-cli nf[4|6]-table <filter|mangle|nat> [all|nonzero]`. Check also whether the filtering module is enabled, using `fp-cli nf[4|6]`. Some targets and rules are not supported in the fast path: check that you are using only documented supported options.

    Refer to the *Fast Path Baseline* documentation for details on the filtering module.

Connectivity problems

Symptoms

- I can no longer connect (via the network) to my machine.
- The VM was configured to redirect connections to the guest (using something like `-netdev user,id=user.0,hostfwd=tcp::35047-:22`).

Hints

- When starting Virtual Accelerator, NIC kernel drivers have been unloaded and thus all IP configuration lost.

Network configuration lost after restart

Symptoms

- My network configuration no longer works after reboot or Virtual Accelerator restart. For example:
  1. My OVS bridge complains `No such device` for a port. The port name has been changed from what was initially configured at boot. Now the bridge shows a wrong device:

```
# ovs-vsctl show
d4bf3dfd-1f25-4316-a01b-0bedb17470ab
Bridge "br0"
    Port "br0"
        Interface "br0"
            type: internal
    Port "eth1"
        Interface "eth1"
    Port "tap0"
        Interface "tap0"
            error: "could not open network device tap0 (No such device)"
ovs_version: "2.4.0-4312f7"
```

Note: Last known OVS configuration is automatically applied at startup.

  2. My linux bridge is empty after stopping (or restarting) the fast path:

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
```

Hints

- The fast path may replace, change, delete and create netdevices. Any tool (`brctl`, iproute2, etc.) that use "old" references to netdevices must have its configuration refreshed when the fast path is stopped.

  - For OVS, it may be necessary to manually delete and re-add the guilty port from the bridge. e.g.:

```
# ovs-vsctl del-port br0 tap0
# ovs-vsctl add-port br0 tap0
# ovs-vsctl show
d4bf3dfd-1f25-4316-a01b-0bedb17470ab
Bridge "br0"
    Port "br0"
        Interface "br0"
            type: internal
    Port "tap0"
        Interface "tap0"
    Port "eth1"
        Interface "eth1"
ovs_version: "2.4.0-4312f7"
```

  - For linux bridge, recreate the bridge and re-add the ports if need be. e.g.:

```
# brctl addbr br0
# brctl addif br0 eth1
# brctl addif br0 tap0
```

- To ensure your network configuration will get restored at reboot, consider using `/etc/network/interfaces` with appropriate options.

  See also:

  - Refer to Debian's documentation on network configuration (https://wiki.debian.org/NetworkConfiguration) for persistent configuration.
  - Refer to Open vSwitch's documentation (https://github.com/openvswitch/ovs/blob/master/debian/openvswitch-switch.README.Debian) for details on persistent OVS ports and bridge configuration.

Hotplug ports lost after reboot

Symptoms

- My hotplug port (initially created with `fp-vdev` and a VM instanciation) disappeared after reboot or fast path restart.
- My bridge configuration that used it is now referencing an unexisting netdevice.

Hints

- Hotplug ports are not persistent. Any tool (`ovs-vsctl`, `brctl`, iproute2, etc.) that use "old" references to netdevices must have its configuration refreshed when the fast path is stopped.
- Remove any references to your hotplug ports from your existing Open vSwitch / linux bridge configuration after a reboot or a fast path restart.

DKMS takes too long

Symptoms

- Modules recompilation/removal with DKMS takes too long.

Hints

- Edit the DKMS configuration in `/etc/dkms/framework.conf`, to prevent it from running some long operations:

```
# mkdir -p /etc/dkms
# echo 'no_initrd="y"' >> /etc/dkms/framework.conf
# echo 'no_depmod="y"' >> /etc/dkms/framework.conf
```

- Disable weak-modules:

```
# chmod a-x /sbin/weak-modules
```

Open vSwitch synchronization problems

Symptoms

- My ports are detected by `ovs-vsctl` but not in the fast path:

```
# ovs-vsctl show
51d477cd-0592-4180-91b3-c5704869ae25
    Bridge "br0"
        Port "ntfp1"
            Interface "ntfp1"
        Port "ntfp2"
            Interface "ntfp2"
        Port "br0"
            Interface "br0"
                type: internal
    ovs_version: "2.4.0-4312f7"

# fpcmd fp-vswitch-ports
<Empty output>
```

- My packets are consequently bridged through Linux (slow path), and appear as exceptions in the stats:

```
# fpcmd stats non-zero
[...]
==== exception stats:
  LocalBasicExceptions:78
  LocalExceptionClass:
```

(continues on next page)

(continued from previous page)

```
  FPTUN_EXC_SP_FUNC:78
  LocalExceptionType:
  FPTUN_BASIC_EXCEPT:78
```

Hints

- Each time the fast path is restarted, you must also restart Open vSwitch.

  When in doubt, clean your bridge and re-configure it. Refer to the *Fast Path OVS Acceleration* documentation for further details on Open vSwitch related configuration.

Open vSwitch and GRE / VXLAN issues

Symptoms

- Packets are not bridged through my OVS GRE port.
- Packets are not bridged through my OVS VXLAN port.
- Statistics show an increasing number of `output_failed_unknown_type`:

```
# fpcmd fp-vswitch-stats non-zero
  flow_not_found:5
  output_failed_unknown_type:145
  set_tunnel_id:145
```

Hints

- Check whether your fast path is compiled with support for GRE or VXLAN respectively, using:

```
# fp-cli conf compiled | grep GRE=y
CONFIG_MCORE_GRE=y
...
# fp-cli conf compiled | grep VXLAN=y
CONFIG_MCORE_VXLAN=y
...
```

  If the command output is empty, then your fast path does not support it.

VRRP is unable to work on VMware virtual machines

Symptoms

- VRRP reports master state on all members but no member receives packets intended for the VRRP virtual IP

Cause

The VMware VSwitch drops frames to MAC addresses that are unknown from the network card properties of the

- VRRP gives the ability to define a virtual IP that can move between machines. By design, the virtual IP is associated to a virtual MAC address - different from the real network card's MAC address. Using a virtual MAC address instead of a real MAC address makes the switchover quicker as no update of ARP tables is needed. However, a Virtual MAC address is not supported by the VMware VSwitch. Unlike physical switches, the VSwitch has no MAC learning mechanism capability. The network section of the VM properties defines virtual network cards and one MAC address per card. The VSwitch only knows those addresses to determine on what port to send a frame. Frames to any unknown addresses, including virtual VRRP MAC addresses, are dropped.
- VRRP uses multicast packets to send VRRP protocol messages between its members. Multicast packets use typical multicast MAC addresses that are also not known by the VSwitch.

Hints

One of the following solutions should be applied:

- Warning: this solution may impact the performance on VMware hypervisor. Enable promiscuous mode on all VSwitches associated with the VLAN you want VRRP to run on. Basically, the VM will receive all traffic within the VSwitch VLAN. Refer to https://kb.vmware.com/s/article/1002934 for more information.
- Set up the VRRP instance to disable the usage of a virtual MAC address "vmac" and to use manual unicast peers to exchange VRRP protocol data unit instead of using multicast. This solution is only applicable to VMware and should not be applied on any other context without an explicit request from support. In this mode, the virtual IP address is associated to the real NIC MAC address of the active member. Upon member switchover, a gratuitous ARP is sent to advertise other machines to update their ARP table with the new MAC. You must ensure and test that gratuitous ARP are treated correctly by all machines. If not, some machines would lose connectivity until the ARP cache timeout expires.

Conflict between i40e FW-LLDP and software LLDP agents

Symptoms

- LLDPDU are not received while the source correctly sends them and the link between both machine works.

Cause

LLDPDU may be consumed by the LLDP engine integrated in the network card firmware:

- Some Intel network adapter (like Ethernet 700 series) has built-in hardware LLDP engine, which is enabled by default. The LLDP Engine is responsible for receiving and consuming LLDP frames, and also replies to the LLDP frames that it receives. The LLDP engine does not forward LLDP frames to the network stack of the Operating System. The i40e driver enable this feature by default.

Hints

The firmware LLDP must be disabled in i40e ports:

- For fast path ports:

```
# fp-cli dpdk-i40e-debug-lldp-cmd on|off
```

- For Linux ports: To disable the FW-LLDP:

```
# ethtool --set-priv-flags <ifname> disable-fw-lldp off
```

To check the FW-LLDP setting:

```
# ethtool --show-priv-flags <ethX>
```

See also:

The `FW-LLDP (Firmware Link Layer Discovery Protocol)` chapter of the i40e Linux Base Driver for Intel controller (https://downloadmirror.intel.com/24411/eng/README.txt).

3.2.3 Performance Tuning

Slow packet processing

Symptoms

- Packet processing performance is not as high as expected.

Hints

- Follow the advice provided in `fast-path.sh config -i` when using the advanced configuration.
- If running in a VM, check that the `qemu` instance handling your VM is pinned on specific cores. See *CPU Pinning for VMs* section for details.

- Check the output from `fast-path.sh config --dump`, optionally with the `--long` option, looking for the number of enabled cores (`FP_MASK`). You might need to add more, especially if `fp-cpu-usage` shows that enabled cores are all at 100%. See `fp-cpu-usage` section for details. Refer to the *Fast Path Baseline* for further details on the wizard (`fast-path.sh config`).
- Check the output from `fast-path.sh config --dump`, optionally with the `--long` option, making sure it is consistent with the available resources in the system. Typically, use memory linked with the appropriate socket considering your system.
- Check that your fast path configuration (memory, sockets, hugepages, ...) fits your current system. Look at *lstopo* output to see if your current configuration is consistent.
- Check whether offload is supported and enabled on your port, using `fp-cli dpdk-port-offload`. See *fp-cli dpdk-port-stats* section for details.
- Check that `exception stats` fast path statistics are not too high. Basic exceptions indicate how many packets could not be processed by the fast path, and have thus been injected in the Linux stack for slow path processing. If the value is high, it is a good indicator that IP addresses/routes/tunnels in the fast path are badly configured. See *Fast Path statistics* section for an example of stats. Refer to the *Fast Path Baseline* documentation for further details on `fp-cli` commands.
- Check the dynamic core/port mapping (which core poll which port) using `fp-cli dpdk-core-port-mapping` (DPDK only), and ensure that all cores will get work from NICs.
- Check what functions the fast path is spending most of its time in, using `perf top`. See *perf* section for details.

Netfilter interferences

Symptoms

- Packet processing performance is not as high as expected.
- Netfilter or Ebtables are enabled:

```
# fp-cli nf4
IPv4 netfilter is on
# fp-cli filter-bridge
filter bridge is on
```

- Libvirt created automatically netfilter rules along with its `virbr0` interface. For instance:

```
root@dut-vm:~# iptables-save
```

```
# Generated by iptables-save v1.4.21 on Mon Apr  4 14:28:56 2016
*mangle
:PREROUTING ACCEPT [8:974]
:INPUT ACCEPT [8:974]
```

(continues on next page)

(continued from previous page)

```
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [6:432]
:POSTROUTING ACCEPT [6:432]
-A POSTROUTING -o virbr0 -p udp -m udp --dport 68 -j CHECKSUM --checksum-fill
COMMIT
# Completed on Mon Apr  4 14:28:56 2016
# Generated by iptables-save v1.4.21 on Mon Apr  4 14:28:56 2016
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [6:432]
:POSTROUTING ACCEPT [6:432]
-A POSTROUTING -s 192.168.122.0/24 -d 224.0.0.0/24 -j RETURN
-A POSTROUTING -s 192.168.122.0/24 -d 255.255.255.255/32 -j RETURN
-A POSTROUTING -s 192.168.122.0/24 ! -d 192.168.122.0/24 -p tcp -j MASQUERADE␣
↪--to-ports 1024-65535
-A POSTROUTING -s 192.168.122.0/24 ! -d 192.168.122.0/24 -p udp -j MASQUERADE␣
↪--to-ports 1024-65535
-A POSTROUTING -s 192.168.122.0/24 ! -d 192.168.122.0/24 -j MASQUERADE
COMMIT
# Completed on Mon Apr  4 14:28:56 2016
# Generated by iptables-save v1.4.21 on Mon Apr  4 14:28:56 2016
*filter
:INPUT ACCEPT [8:974]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [6:432]
-A INPUT -i virbr0 -p udp -m udp --dport 53 -j ACCEPT
-A INPUT -i virbr0 -p tcp -m tcp --dport 53 -j ACCEPT
-A INPUT -i virbr0 -p udp -m udp --dport 67 -j ACCEPT
-A INPUT -i virbr0 -p tcp -m tcp --dport 67 -j ACCEPT
-A FORWARD -d 192.168.122.0/24 -o virbr0 -m conntrack --ctstate RELATED,
↪ESTABLISHED -j ACCEPT
-A FORWARD -s 192.168.122.0/24 -i virbr0 -j ACCEPT
-A FORWARD -i virbr0 -o virbr0 -j ACCEPT
-A FORWARD -o virbr0 -j REJECT --reject-with icmp-port-unreachable
-A FORWARD -i virbr0 -j REJECT --reject-with icmp-port-unreachable
-A OUTPUT -o virbr0 -p udp -m udp --dport 68 -j ACCEPT
COMMIT
# Completed on Mon Apr  4 14:28:56 2016
```

Hints

- Netfilter rules impact performance. If your rules only affect traffic going to Linux and not dataplane traffic, you may want to disable Netfilter synchronization using the following commands:

```
# fp-cli nf4-set off
IPv4 netfilter is off
# fp-cli filter-bridge-set off
filter bridge is off
```

Performance drop with Mellanox ConnectX-3 devices

Symptoms:

- Packet processing is slower than expected

Hints:

- On Dell and SuperMicro servers, PCI read buffer may be misconfigured for ConnectX-3/ConnectX-3-Pro NICs. Check the output of `setpci -s <NIC_PCI_address> 68.w`. For instance:

```
# lspci | grep Mellanox
04:00.0 Ethernet controller: Mellanox Technologies MT27520 Family [ConnectX-3␣
↪Pro]
# setpci -s 04:00.0 68.w
202e
```

> **Warning:** Beware with the following command, it is known to cause spontaneous reboot on some systems.

If the value is below 0x5020 (here that's the case), set it to 0x5020:

```
# setpci -s 04:00.0 68.w=5020
```

Performance optimization for the NFV use case

Symptoms:

- Expecting more performance in the NFV use case

Hints:

- The *nfv* profile can used by setting it in the `/etc/fp-vdev.ini` file or by passing *–profile=nfv* to *fp-vdev* in order to increase the performance of the packet processing, especially when:
  - the VM runs a fast path or a DPDK application.
  - the VM does not terminate the traffic and does not require the offload.
  - the number of cores in the host is higher than the number of cores in the guest.

This option will lower the lock contention on Tx rings, at the price of degrading the packets spreading among guest cores. It also disables the *mergeable buffer* Virtio feature.

It is possible to check if the *nfv* profile is set by checking the used configuration to create the virtual interface with the fp-shmem-ports command.

```
# fp-shmem-ports -d
[snip]
port 2: tap0-vrf0 [snip] driver net_6wind_vhost (args sockname=/tmp/tap0.sock,
↪sockmode=client,profile=endpoint) [snip]
[snip]
port 3: tap1-vrf0 [snip] driver net_6wind_vhost (args sockname=/tmp/tap1.sock,
↪sockmode=client,profile=nfv) [snip]
```

On the above example the tap0 created by fp-vdev does not have the nfv profile whereas the tap1 has it.

See also:

The 6WINDGate *Fast Path Managing virtual devices* documentation for more information about the `fp-vdev` command.

3.2.4 Virtual Environment

This section gathers issues that happen with VMs started on top of Virtual Accelerator.

libvirt and hugepages

Symptoms

- Starting a VM (with hugepages support) with libvirt fails, yielding:

```
error: internal error: Unable to find any usable hugetlbfs mount for 2048 KiB
```

Your XML libvirt domain contains something like this:

```
<memory unit='KiB'>2097152</memory>
<memoryBacking>
  <hugepages>
    <page size="2048" unit="KiB" nodeset="0"/>
  </hugepages>
</memoryBacking>
```

- Starting a VM with libvirt fails, yielding:

```
error: operation failed: domain is no longer running
```

Or:

```
error: unsupported configuration: Shared memory mapping is supported only␣
↪with hugepages
```

- The logs in `/var/log/libvirt/libvirtd.log` show:

```
Cannot set up guest memory 'pc.ram': Cannot allocate memory
error : qemuDomainObjExitMonitor:1603 : operation failed: domain is no longer␣
↪running
```

Or:

```
error : qemuBuildNumaArgStr:6852 : unsupported configuration: Shared memory␣
↪mapping is supported only with hugepages
```

Hints

- The hugepages are allocated by Virtual Accelerator at startup and libvirt cannot detect them dynamically. libvirt must be restarted to take the hugepages into account. Use `service libvirt-bin restart` after having started the fast path. If you manually handle your hugepages, make sure there are enough (and also restart `libvirt-bin` afterwards).
- Check the output from `/proc/meminfo`, especially the `MemFree` and `HugePage_Free` fields. See *meminfo* section for details.

  `HugePage_Free` indicates how many huge pages are available for your VMs.
- Check the output from `fast-path.sh config --dump --full`, optionally with the `--long` option. `VM_MEMORY` might need adjustment to reserve more hugepages for VMs. Refer to the *Fast Path Baseline* for further details on the wizard (`fast-path.sh config`).
- Your XML libvirt domain needs an appropriate <`hugepage`> tag to leverage hugepages reserved by the fast path.

Cannot read disk image

Symptoms

- Starting a VM with libvirt fails, yielding:

```
error: internal error: process exited while connecting to monitor:
qemu-system-x86_64: -drive file=/root/ubuntu-14.04-template.qcow2,if=none,
↪id=drive-virtio-disk0,format=qcow2,cache=none: could not open disk image
```

Hints

- Your image is not readable (or in an inaccessible folder) by user `libvirt`, which runs qemu. Change your image/folder's rights to allow access to libvirt. You may also move your image directly into `/var/lib/libvirt/images`.

Performance degradation and serial link

Symptoms

- My VM forwards packets slower when I use intensely the serial link to it. For instance, when refreshing `top` regularly in `virsh console`.

Hints

- The serial link triggers interruptions (IRQ (Interrupt Requests)) when there is something happening on it. You need to make sure dataplane cores are not among those that handle IRQ. Set appropriate affinities (with cores not used by the fast path) for IRQ in `/proc/irq/*/smp_affinity`. E.g.: if the fast path runs on core 1 (on a machine with 4 cores), use a `0xd` mask:

```
for aff in /proc/irq/*/smp_affinity; do echo d > $aff; done
```

Performance degradation and offloads

Symptoms

- VM packet processing is slower when packets are forwarded in the VM than when packets are terminated there.

Hints

- Consider disabling TSO/GRO offloading in case your VMs mostly forward traffic, to increase performance. You can dynamically try that by calling `fp-shmem-ports -e all -K gro off` and see if that improves performance. You can also configure that permanently in your fast path configuration file using the wizard (`Advanced configuration` section regarding offloads) or directly editing FP_OFFLOAD in `fast-path.env`.

  Refer to *Configuration wizard* for further details.

3.3 Fast Path Information

3.3.1 Fast Path statistics

Use `fp-cli stats [percore] [non-zero]` to get the statistics recorded by the fast path:

```
# fp-cli stats non-zero
==== interface stats:
lo-vr0 port:254
mgmt0-vr0 port:254
enp3s0f1-vr0 port:254
ens785f1-vr0 port:254
ens787f1-vr0 port:254
```

(continues on next page)

(continued from previous page)

```
ens804f1-vr0 port:254
ens806f1-vr0 port:254
fpn0-vr0 port:254
ntfp4-vr0 port:0
ntfp1-vr0 port:1
ntfp2-vr0 port:2
ntfp3-vr0 port:3
==== global stats:
==== exception stats:
  LocalBasicExceptions:7
  LocalExceptionClass:
  LocalExceptionType:
==== IPv4 stats:
  IpForwDatagrams:509870627
  IpInReceives:509870627
==== arp stats:
==== IPv6 stats:
==== TCP stats:
==== UDP stats:
==== UDP6 stats:
==== IPsec stats:
==== IPsec IPv6 stats:
==== L2 stats:
==== fp-vswitch stats:
```

Note: Refer to *Fast Path Baseline* documentation for details on this command.

3.3.2 fp-cpu-usage

Use this command to get the fast path usage per core, and the number of cycles to process one packet:

```
# fp-cpu-usage
Fast path CPU usage:
cpu: %busy     cycles   cycles/packet
  2:   70%  227166479             990
  3:   69%  222733174             991
...
average cycles/packets received from NIC: 991 (5389132282/5436242)
```

It is a good indicator regarding how busy the fast path cores are, processing packets.

Note: Refer to FPN-SDK documentation for details on this command.

3.3.3 Turn Fast Path off

Use the following command to turn most of the fast path off:

```
# fp-cli fp-state-set off
FP is stopped (was started)
```

By doing this, no processing will be done by the fast path. As soon as the fast path receives a packet on a port, without any processing, it will inject it in the linux stack.

If the test works with the fast path thus `disabled`, it usually means the fast path drops packets.

3.4 System Information

3.4.1 CPU Pinning for VMs

For each virtual CPU, QEMU uses one pthread for actually running the VM and pthreads for management. For best performance, you need to make sure cores used to run fast path dataplane are only used for that.

To get the threads associated with each virtual CPU, use `info cpus` in QEMU monitor console:

```
QEMU 2.3.0 monitor - type 'help' for more information
(qemu) info cpus
* CPU #0: pc=0xffffffff8104f596 (halted) thread_id=26773
  CPU #1: pc=0x00007faee19be9f9 thread_id=26774
  CPU #2: pc=0xffffffff8104f596 (halted) thread_id=26775
  CPU #3: pc=0x0000000000530233 thread_id=26776
```

To get all threads associated with your running VM (including management threads) and what CPU they are currently pinned on, call:

```
# taskset -ap <qemu_pid>
pid 26770's current affinity mask: f
pid 26771's current affinity mask: f
pid 26773's current affinity mask: f
pid 26774's current affinity mask: f
pid 26775's current affinity mask: f
pid 26776's current affinity mask: f
pid 27053's current affinity mask: f
```

By pinning our VM on a specific set of cores, we ensure less overload.

You may either run qemu with a specific set of cores when starting, using:

```
# taskset -c 0-1 <qemu command>
```

You may also pin a VM after it has been started, using the PID (Process Identifier) of its threads. For instance, to change the physical CPU on which to pin the virtual CPU #0, use:

```
# taskset -cp 0-1 26773
pid 26773's current affinity list: 0-3
pid 26773's new affinity list: 0,1
```

Note: Refer to the taskset manpage for specific options.

When using libvirt, you may use <cputune> with vcpupin to pin virtual CPUs to physical ones. e.g.:

```
<vcpu cpuset='7-8,27-28'>4</vcpu>
<cputune>
  <vcpupin vcpu="0" cpuset="7"/>
  <vcpupin vcpu="1" cpuset="8"/>
  <vcpupin vcpu="2" cpuset="27"/>
  <vcpupin vcpu="3" cpuset="28"/>
</cputune>
```

Note: Refer to the libvirt Domain XML format (http://libvirt.org/formatdomain.html#elementsCPUTuning) documentation for further details.

We can look at htop results (after filtering results for this qemu instance) to confirm what threads are actually used:

```
  PID USER       VIRT   RES   SHR S CPU% MEM%   TIME+  NLWP Command
26770 mazon     7032M 4067M  7092 S 200. 25.5  2h19:55    7 |- qemu-system-x86_64 -
↪daemonize --enable-kvm -m 6G -cpu host -smp sockets=1,cores=4,threads=1 ...
27053 mazon     7032M 4067M  7092 S  0.0 25.5  0:01.13    7 |  |- qemu-system-x86_64 -
↪daemonize --enable-kvm -m 6G -cpu host -smp sockets=1,cores=4,threads=1 ...
26776 mazon     7032M 4067M  7092 R 99.1 25.5  1h04:38    7 |  |- qemu-system-x86_64 -
↪daemonize --enable-kvm -m 6G -cpu host -smp sockets=1,cores=4,threads=1 ...
26775 mazon     7032M 4067M  7092 S  0.9 25.5  2:48.21    7 |  |- qemu-system-x86_64 -
↪daemonize --enable-kvm -m 6G -cpu host -smp sockets=1,cores=4,threads=1 ...
26774 mazon     7032M 4067M  7092 R 99.1 25.5  1h09:42    7 |  |- qemu-system-x86_64 -
↪daemonize --enable-kvm -m 6G -cpu host -smp sockets=1,cores=4,threads=1 ...
26773 mazon     7032M 4067M  7092 S  0.0 25.5  2:23.03    7 |  |- qemu-system-x86_64 -
↪daemonize --enable-kvm -m 6G -cpu host -smp sockets=1,cores=4,threads=1 ...
26771 mazon     7032M 4067M  7092 S  0.0 25.5  0:00.00    7 |  |- qemu-system-x86_64 -
↪daemonize --enable-kvm -m 6G -cpu host -smp sockets=1,cores=4,threads=1 ...
```

You may even change CPU affinity by typing `a` when on a specific PID line in `htop`.

Similarly, you can get threads PID by looking in `/proc/<pid>/task/`, e.g.:

```
# ls /proc/26773/task
26770/  26771/  26773/  26774/  26775/  26776/  27053/
```

3.4.2 fp-cli dpdk-port-stats

The `fp-cli dpdk-port-stats` command is used to display and set options related to network drivers (for those that support it).

To display the statistics for a given port, use `fp-cli dpdk-port-stats <port>`:

```
# fp-cli dpdk-port-stats <port>

    rx_good_packets: 261064663
    tx_good_packets: 256512062
    rx_good_bytes: 15663879780
    tx_good_bytes: 15390725600
    rx_missed_errors: 0
    rx_errors: 36554346
    tx_errors: 0
    rx_mbuf_allocation_errors: 0
    rx_q0_packets: 32632951
    rx_q0_bytes: 1957977060
    rx_q0_errors: 0
    ...
    tx_q0_packets: 128251039
    tx_q0_bytes: 7695062334
    ...
    rx_total_packets: 297619011
    rx_total_bytes: 17857140760
    tx_total_packets: 256512062
    tx_size_64_packets: 256512046
    ...
```

The most important stats to look at are the `{r,t}x_good_{packets,bytes}` and errors.

They indicate globally how well the port is handling packets.

There is also per queue statistics that might be interesting in case of multiqueue. It's better to have packets transmitted on as many different queues as possible, but it depends on various factors, such as the IP addresses and UDP / TCP ports.

The drop statistics provide useful information about why packets are dropped. For instance, the `rx_missed_errors` counter represents the number of packets dropped because the CPU was not fast enough to dequeue them. A non-

zero value for `rx_mbuf_allocation_errors` shows that there is not enough `mbuf` structure configured in the fast path.

Note: Statistics field names may vary considering the driver.

`fp-cli dpdk-port-offload` can be used to check whether offload is enabled, using the following:

```
# fp-cli dpdk-port-offload <port>

    TX vlan insert off [fixed]
    TX IPv4 checksum off [fixed]
    TX TCP checksum off [fixed]
    TX UDP checksum off [fixed]
    TX SCTP checksum off [fixed]
    TX TSO off [fixed]
    TX UFO off [fixed]
    RX vlan strip off
    RX vlan filter off
    RX IPv4 checksum on
    RX TCP checksum on
    RX UDP checksum on
    RX MPLS IP off
    RX LRO off
    RX GRO off
```

If you want to enable TSO (which should provide you with better performance for TCP, as the hardware will handle the segmentation), use:

```
# fp-cli dpdk-port-offload-set eth1 tso on
```

Note: You can get various error messages when trying to change hardware parameters. For instance, `Cannot change tcp-segmentation-offload` may appear if the driver does not support to dynamically change TSO offload.

3.4.3 `lspci`

The `lspci` command is useful to display information about PCI (Peripheral Component Interconnect) buses. In most cases, we look for “Ethernet” devices.

Use `lspci` to get basic information on all connected devices:

```
# lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
00:02.0 VGA compatible controller: Device 1234:1111 (rev 02)
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller␣
↪(rev 03)
```

To display the driver handling devices, use:

```
# lspci -k
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
        Subsystem: Red Hat, Inc Qemu virtual machine
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
        Subsystem: Red Hat, Inc Qemu virtual machine
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]
        Subsystem: Red Hat, Inc Qemu virtual machine
        Kernel driver in use: ata_piix
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
        Subsystem: Red Hat, Inc Qemu virtual machine
00:02.0 VGA compatible controller: Device 1234:1111 (rev 02)
        Subsystem: Red Hat, Inc Device 1100
00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller␣
↪(rev 03)
        Subsystem: Red Hat, Inc QEMU Virtual Machine
        Kernel driver in use: igb_uio
```

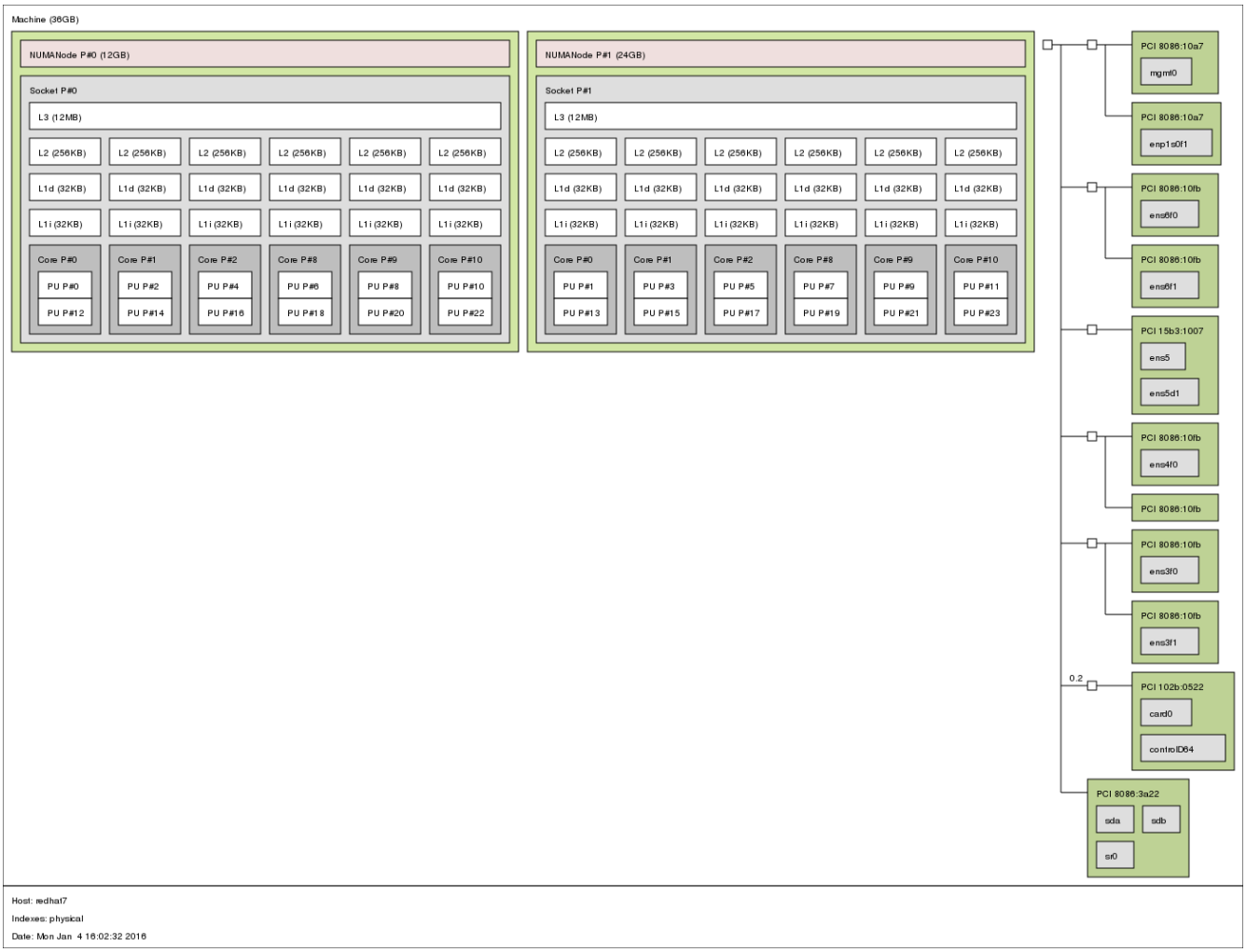
Note: Refer to the `lspci` manpage for specific options.

3.4.4 `lstopo`

`lstopo` provides a global view of the system's topology. It details what machines contain what nodes, containing sockets, containing cores, containing processor units.

The following command presents a graphical representation of a big machine's topology:

```
# lstopo --of png > lstopo_output.png
```



You can use the following command to get a textual representation:

```
# lstopo --of txt
```

3.4.5 meminfo

The file `/proc/meminfo` presents a memory status summary. You can also look at memory by node through `/sys/devices/system/node/node<node_id>/meminfo`.

On a VM with 1GB of RAM (Random-Access Memory) running redhat-7, we have this:

```
# cat /proc/meminfo
MemTotal:        1016548 kB
MemFree:          107716 kB
MemAvailable:     735736 kB
Buffers:           83244 kB
```

(continues on next page)

(continued from previous page)

```
Cached:           626528 kB
SwapCached:            0 kB
Active:           400416 kB
Inactive:         352892 kB
Active(anon):      49808 kB
Inactive(anon):    13304 kB
Active(file):     350608 kB
Inactive(file):   339588 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:         43652 kB
Mapped:             9500 kB
Shmem:             19572 kB
Slab:             130972 kB
SReclaimable:     100896 kB
SUnreclaim:        30076 kB
KernelStack:        1888 kB
PageTables:         2692 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:      507248 kB
Committed_AS:     214004 kB
VmallocTotal:   34359738367 kB
VmallocUsed:        4412 kB
VmallocChunk:   34359730912 kB
HardwareCorrupted:     0 kB
AnonHugePages:      4096 kB
HugePages_Total:       1
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:       79744 kB
DirectMap2M:      968704 kB
```

Note: The kernel documentation provides details regarding meminfo here (http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation/filesystems/proc.txt?id=HEAD#n819).

For details regarding the `HugePages` fields, look there (http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Document

The most interesting fields in our case are:

`MemTotal` should be the same as the "total" memory displayed on top lines when running `top`

`MemFree` should be the same as the "free" memory displayed on top lines when running `top`

`MemAvailable` estimate of how much memory is available for starting new applications, without swapping

`HugePages_Total` size of the pool of huge pages

`HugePages_Free` number of huge pages in the pool that are not yet allocated

`HugePages_Rsvd` number of huge pages for which a commitment to allocate from the pool has been made, but no allocation has yet been made

3.4.6 `numastat`

This tool shows per-NUMA-node memory statistics for processes and the operating system.

Without argument, it displays per-node NUMA hit and miss system statistics from the kernel memory allocator. A high value in `other_node` means that there are cross-NUMA memory transfers, which impacts performance. This information is dynamic and can be monitored with the `watch` command.

```
# numastat
                           node0           node1
numa_hit                589246433       556912817
numa_miss                     0               0
numa_foreign                  0               0
interleave_hit            11616           17088
local_node              589229023       556900289
other_node                17410           12528
```

When a PID or a pattern is passed, it shows per-node memory allocation information for the specified process (including all its pthreads). The hugepages correspond to the DPDK memory, and the private area mainly corresponds to the shared memories.

```
# numastat fp-rte
Per-node process memory usage (in MBs) for PID 2176 (fp-rte:8)
                           Node 0          Node 1           Total
                  --------------- --------------- ---------------
Huge                       842.00          842.00         1684.00
Heap                         0.41            0.00            0.41
Stack                        0.03            0.00            0.03
Private                   1004.35           24.27         1028.62
----------------  --------------- --------------- ---------------
Total                     1846.79          866.27         2713.06
```

Note: Refer to the `numa` manpage for details.

3.4.7 Scheduler statistics

The fast path dataplane threads are designed to run on dedicated cores. An interruption of a fast path thread will prevent the Rx queues to be polled. If this interruption lasts too long, it can induce packet drops.

This condition is enforced by the `cpuset.sh` script which is started by the fast path service, which prevents all other tasks to run on cores reserved for the fast path. It is enabled by default, see `/etc/cpuset.env` for its configuration.

Some tasks may still run on the control plane cores, especially the kernel threads, which may not be an issue if the duration of the kernel task is not too high. The number of context switches can be monitored in `/proc/<pid>/task/<tid>/status`:

```
# grep 'Name\|ctxt\|Cpus' /proc/$(pidof fp-rte)/task/*/status
/proc/2176/task/2176/status:Name:       fp-rte:8
/proc/2176/task/2176/status:Cpus_allowed:       000100
/proc/2176/task/2176/status:Cpus_allowed_list:  8
/proc/2176/task/2176/status:voluntary_ctxt_switches:    65
/proc/2176/task/2176/status:nonvoluntary_ctxt_switches: 19883672
/proc/2176/task/2177/status:Name:       eal-intr-thread
/proc/2176/task/2177/status:Cpus_allowed:       fffcff
/proc/2176/task/2177/status:Cpus_allowed_list:  0-7,10-23
/proc/2176/task/2177/status:voluntary_ctxt_switches:    10
/proc/2176/task/2177/status:nonvoluntary_ctxt_switches: 0
/proc/2176/task/2178/status:Name:       fp-rte:9
/proc/2176/task/2178/status:Cpus_allowed:       000200
/proc/2176/task/2178/status:Cpus_allowed_list:  9
/proc/2176/task/2178/status:voluntary_ctxt_switches:    237
/proc/2176/task/2178/status:nonvoluntary_ctxt_switches: 19740497
/proc/2176/task/2179/status:Name:       dpdk-ctrl-port
/proc/2176/task/2179/status:Cpus_allowed:       fffcff
/proc/2176/task/2179/status:Cpus_allowed_list:  0-7,10-23
/proc/2176/task/2179/status:voluntary_ctxt_switches:    3514703
/proc/2176/task/2179/status:nonvoluntary_ctxt_switches: 208025
/proc/2176/task/2182/status:Name:       npf-mgmt
/proc/2176/task/2182/status:Cpus_allowed:       fffcff
/proc/2176/task/2182/status:Cpus_allowed_list:  0-7,10-23
/proc/2176/task/2182/status:voluntary_ctxt_switches:    9
/proc/2176/task/2182/status:nonvoluntary_ctxt_switches: 0
```

The number of context switches for the dataplane fast path threads should be as small as possible. The `perf` monitoring tool can give even more information about the cause and duration of these interruptions.

Example that monitors the CPU 8 (one of the fast path cores):

```
# perf sched record -C 8 -- sleep 1
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.283 MB perf.data (954 samples) ]
# perf script --header
[...]
        fp-rte:8  2176 [008] 608356.309895: sched:sched_stat_runtime: comm=fp-rte:8␣
↪pid=2176 runtime=3999603 [ns] vruntime=5078834868400
        fp-rte:8  2176 [008] 608356.309898: sched:sched_stat_sleep: comm=kworker/8:2␣
↪pid=1373 delay=12000294 [ns]
        fp-rte:8  2176 [008] 608356.309899: sched:sched_wakeup: comm=kworker/8:2␣
↪pid=1373 prio=120 target_cpu=008
        fp-rte:8  2176 [008] 608356.309900: sched:sched_stat_sleep: comm=rcu_sched␣
↪pid=8 delay=3949943 [ns]
        fp-rte:8  2176 [008] 608356.309900: sched:sched_wakeup: comm=rcu_sched pid=8␣
↪prio=120 target_cpu=004
        fp-rte:8  2176 [008] 608356.309902: sched:sched_stat_runtime: comm=fp-rte:8␣
↪pid=2176 runtime=4680 [ns] vruntime=507883486844717
        fp-rte:8  2176 [008] 608356.309902: sched:sched_stat_wait: comm=kworker/8:2␣
↪pid=1373 delay=0 [ns]
        fp-rte:8  2176 [008] 608356.309903: sched:sched_switch: prev_comm=fp-rte:8␣
↪prev_pid=2176 prev_prio=120 prev_state=R ==> next_com
    kworker/8:2  1373 [008] 608356.309904: sched:sched_stat_runtime: comm=kworker/8:2␣
↪pid=1373 runtime=6359 [ns] vruntime=1029668683487
    kworker/8:2  1373 [008] 608356.309905: sched:sched_stat_wait: comm=fp-rte:8␣
↪pid=2176 delay=6359 [ns]
    kworker/8:2  1373 [008] 608356.309905: sched:sched_switch: prev_comm=kworker/8:2␣
↪prev_pid=1373 prev_prio=120 prev_state=S ==> next_
        fp-rte:8  2176 [008] 608356.313893: sched:sched_stat_runtime: comm=fp-rte:8␣
↪pid=2176 runtime=3988661 [ns] vruntime=5078834908333
```

If this causes packet loss, it can be avoided by increasing the size of the Rx queues (example: `FPNSDK_OPTIONS="--nb-rxd=4096"` option in the fast path configuration file).

When running in a latency-sensitive environment, increasing the Rx queue size is not suitable, and kernel thread interruptions can be an issue. This can be tempered by:

- Disabling the watchdog:

  ```
  # echo 0 > /proc/sys/kernel/watchdog
  ```

- Booting the kernel with specific arguments to isolate fast path cores.

  Example with FP_MASK=8-9:

  ```
  isolcpus=8-9 nohz_full=8-9 rcu_nocbs=8-9
  ```

  `isolcpus` isolates the CPUs from the general scheduler. `nohz_full` will stop the scheduler tick whenever

possible (CONFIG_NO_HZ_FULL=y must be set). `rcu_nocbs` prevents rcu callbacks to be scheduled on these CPUs (CONFIG_RCU_NOCB_CPU=y must be set).

Note: Refer to the Linux kernel documentation for details.

3.5 Log Management

3.5.1 `rsyslog`

The `rsyslogd` daemon writes syslog messages in various places (considering its configuration in `/etc/rsyslog.conf`). Messages for the "daemon" facility are usually stored in `/var/log/daemon.log` (this is the case for buildroot images). However all messages are usually stored in `/var/log/syslog` (all facilities included), too.

The fast path sends syslog messages that you can look at later to figure out what happened during startup (and runtime). Here is an extract from `/var/log/syslog`:

```
fp-rte[3660]: EAL: Master lcore 1 is ready (tid=c4fdc300;cpuset=[1])
fp-rte[3660]: EAL: PCI device 0000:00:04.0 on NUMA socket -1
fp-rte[3660]: EAL:   probe driver: 8086:100e rte_em_pmd
fp-rte[3660]: EAL:   PCI memory mapped at 0x7f22c3c00000
fp-rte[3660]: PMD: eth_em_dev_init(): port_id 0 vendorID=0x8086 deviceID=0x100e
...
fp-rte[3660]: libfpn_shmem: write procfs: File exists
fp-rte[3660]: FPN: fp_mask=0x2 l_mask=0x2 dpvi_mask=0x1 stats_mask=0xd online=0xf
...
fp-rte[3660]: Create a mbuf pool on socket 0, nb_mbufs=16384
fp-rte[3660]: Bus  Device        ID         Port#  RXQ  RXD/Q  TXQ  TXD/Q  Excl␣
↪Interface       Driver name
fp-rte[3660]: PCI  0000:00:04.0  8086:100e  0      1    128    1    512    1     N/A␣
↪          rte_em_pmd
fp-rte[3660]: PCI  0000:00:05.0  8086:100e  1      1    128    1    512    1     N/A␣
↪          rte_em_pmd
fp-rte[3660]: PCI  0000:00:06.0  8086:100e  2      1    128    1    512    1     N/A␣
↪          rte_em_pmd
fp-rte[3660]: Initializing port 0... ntxq=1 nrxq=1 [de:ed:01:f0:95:88] txq0=c1 rxq0=c1␣
↪PMD: eth_em_tx_queue_setup(): sw_ring=0x7f22acdccc00 hw_ring=0x7f22acdced00 dma_
↪addr=0xaffced00
fp-rte[3660]: PMD: eth_em_rx_queue_setup(): sw_ring=0x7f22acdbc6c0 hw_
↪ring=0x7f22acdbcbc0 dma_addr=0xaffbcbc0
fp-rte[3660]: PMD: eth_em_rx_init(): forcing scatter mode
fp-rte[3660]: PMD: eth_em_start(): <<
fp-rte[3660]: done
fp-rte[3660]: Initializing port 1... ntxq=1 nrxq=1 [de:ed:02:f7:f2:e5] txq0=c1 rxq0=c1␣
↪PMD: eth_em_tx_queue_setup(): sw_ring=0x7f22acdaa480 hw_ring=0x7f22acdac580 dma_
↪addr=0xaffac580
```

(continues on next page)

(continued from previous page)

```
fp-rte[3660]: PMD: eth_em_rx_queue_setup(): sw_ring=0x7f22acd99f40 hw_
↪ring=0x7f22acd9a440 dma_addr=0xaff9a440
fp-rte[3660]: PMD: eth_em_rx_init(): forcing scatter mode
fp-rte[3660]: PMD: eth_em_start(): <<
fp-rte[3660]: done
...
fp-rte[3660]: fpn_sdk_init: dedicated configuration polling lcore -1
fp-rte[3660]: fpn_dpvi_shmem_mmap: fpn_dpvi_shmem sizeof=80
fp-rte[3660]: fpn_dpvi_ring_shmem_mmap: fpn_dpvi_ring_shmem size=266496
fp-rte[3660]: fpn_per_lcore_dpvi_shmem_init: lcoreid 1 mbufs=768
kernel: [   57.450256] dpvi: kernel_cpumask_display() dpvi: fp_mask = 0x2
kernel: [   57.450259] dpvi: kernel_cpumask_display() dpvi: dpvi_mask = 0x1
kernel: [   57.450260] dpvi: kernel_cpumask_display() dpvi: l_mask = 0x2
kernel: [   57.450261] dpvi: kernel_cpumask_display() dpvi: online_mask = 0xf
kernel: [   57.450263] dpvi: dpvi_init_ring() dpvi: cpu 0 use Tx queue 0  ring 1
kernel: [   57.450264] dpvi: dpvi_init_ring() dpvi: cpu 1 use Tx queue 0  ring 1
kernel: [   57.450264] dpvi: dpvi_init_ring() dpvi: cpu 2 use Tx queue 0  ring 1
kernel: [   57.450265] dpvi: dpvi_init_ring() dpvi: cpu 3 use Tx queue 0  ring 1
kernel: [   57.451284] dpvi: dpvi_sysctl_running_fastpath() Watching PID 3660
fp-rte[3660]: fpn-sdk init finished
fp-rte[3660]: fp-ovs: using accelerated functions(avx[x] sse4.2[x] sse4.1[x])
fp-rte[3660]: Using fp-shared=0x7f22a0069000 size=20390912 (19M)
...
fp-rte[3660]: fp-ovs: using accelerated functions(avx[x] sse4.2[x] sse4.1[x])
fp-rte[3660]: fp-vswitch module loaded
fp-rte[3660]: Init core 1
fp-rte[3660]: entering main loop on lcore 1 (master)
fp-rte[3660]:  RX -- lcoreid=1 queueid=0 portid=0
fp-rte[3660]:  RX -- lcoreid=1 queueid=0 portid=1
fp-rte[3660]:  RX -- lcoreid=1 queueid=0 portid=2
fp-rte[3660]:  TX -- lcoreid=1 queueid=0 portid=0
fp-rte[3660]:  TX -- lcoreid=1 queueid=0 portid=1
fp-rte[3660]:  TX -- lcoreid=1 queueid=0 portid=2
```

If you don't see anything in `/var/log/syslog`, make sure the `rsyslogd` daemon is running:

```
# ps aux | grep syslog
root        83  0.0  0.0 251864  2648 ?        Ssl  13:23   0:00 /usr/sbin/rsyslogd
root       851  0.0  0.0   4676   648 ttyS0    R+   14:32   0:00 grep syslog
```

If `rsyslogd` is not running, refer to your distribution documentation on how to get it started.

Note: Refer to the appropriate manpage (e.g.: `man rsyslog.conf`) for configuration options.

3.5.2 journalctl

With SystemD, logging is handled by the `systemd-journald` daemon. It writes its log in a binary format, and one usually uses `journalctl` to access it.

Use this command to see syslog messages from a given program (providing its path):

```
# journalctl /usr/bin/fpmd
Dec 10 15:56:44 dut-vm fpmd[11412]: bpf module registered
Dec 10 15:56:44 dut-vm fpmd[11412]: inaddr module registered
Dec 10 15:56:44 dut-vm fpmd[11412]: inroute module registered
...
```

You can combine it to follow logs from several programs at once. e.g.:

```
# journalctl /usr/bin/cmgrd /usr/bin/fpmd
...
Dec 10 15:56:44 dut-vm fpmd[11412]: tunnel module registered
Dec 10 15:56:44 dut-vm fpmd[11412]: fpm_netlink_recv: fpn0 found : ifindex 6 status 40
Dec 10 15:56:44 dut-vm cmgrd[11678]: fp-vswitch module loaded
Dec 10 15:56:44 dut-vm cmgrd[11678]: fpm_connect: trying to connect to fpm
Dec 10 15:56:44 dut-vm cmgrd[11678]: fpvs_cm_init_cb: Could not get OVS "ovs_datapath"␣
↪family info
Dec 10 15:56:44 dut-vm fpmd[11413]: add:cannot set flags in FP ens4-vr0: [-95]
...
```

Note: Refer to the appropriate manpage (e.g.: `man journalctl`) for configuration options.

3.5.3 fpmd logs

You can start `fpmd` with `-v` option to have more verbose output. To do that, either:

1. kill and restart (providing `-v`) the `fpmd` process, once the fast path has been started:

   ```
   # killall fpmd
   # fpmd -v
   ```

2. provide `-v` in `FPM_OPTIONS` when starting the fast path (you could set it in `/etc/fpm.env` too):

   ```
   # FPM_OPTIONS='-v' fast-path.sh start
   ```

See also:

Linux - Fast Path Synchronization

3.5.4 cmgrd logs

When facing a synchronization issue, check first the line with `fpm_connection` in the log. It should display `connected to fpm socket` when the connection is established between `cmgrd` and `fpmd`.

You can start `cmgrd` with `-d` option (use `0xffffffff` for maximum debug level) to display more information regarding received netlink messages, messages sent to `fpmd`, etc. To do that, either:

1. kill and restart (providing `-d`) the `cmgrd` process, once the *Linux - Fast Path Synchronization* has been started:

```
# killall cmgrd
# cmgrd -d 0xffffffff
```

2. provide `-d` in `CMGR_OPTIONS` when starting the *Linux - Fast Path Synchronization* (you could set it in `/etc/cmgr.env` too):

```
# CMGR_OPTIONS='-d 0xffffffff' linux-fp-sync.sh start
```

See also:

Linux - Fast Path Synchronization

3.5.5 OpenStack logs

When you have an issue regarding VM spawning, take a look at `/var/log/nova/nova-compute.log` on the compute node hosting the VM.

In particular, look for messages with `error` or `trace` in it. For instance:

```
# grep -iE "(error|trace)" /var/log/nova/nova-compute.log
2016-01-27 11:37:22.286 12945 ERROR nova.network.linux_net [req-b7fdc659-2fd5-4d9e-
↪942c-803f71c2cce1 d82509fae77e41009880defd0bbd829e d9c0a5bd157947bab06d355bf4772db7 -
↪ - -] \
  Unable to execute ['ovs-vsctl', '--timeout=120', '--', '--if-exists', 'del-port', u
↪'tap13d2cb29-d6', '--', 'add-port', 'br-int', u'tap13d2cb29-d6', '--', 'set',
↪'Interface', \
                    u'tap13d2cb29-d6', u'external-ids:iface-id=13d2cb29-d61c-46d9-
↪afe9-98b6aa0a43ea', 'external-ids:iface-status=active', u'external-ids:attached-
↪mac=fa:16:3e:6d:ac:ea', \
                    'external-ids:vm-uuid=1065e38c-e6c6-423f-8140-5d0c021d3af0'].␣
↪Exception: Unexpected error while running command.
2016-01-27 11:37:22.289 12945 ERROR nova.compute.manager [req-b7fdc659-2fd5-4d9e-942c-
↪803f71c2cce1 d82509fae77e41009880defd0bbd829e d9c0a5bd157947bab06d355bf4772db7 - - -
↪] \
  [instance: 1065e38c-e6c6-423f-8140-5d0c021d3af0] Instance failed to spawn
2016-01-27 11:37:22.289 12945 ERROR nova.compute.manager [instance: 1065e38c-e6c6-423f-
↪8140-5d0c021d3af0] AgentError: Error during following call to agent: \
```

(continues on next page)

(continued from previous page)

```
  ['ovs-vsctl', '--timeout=120', '--', '--if-exists', 'del-port', u'tap13d2cb29-d6', '-
↪-', 'add-port', 'br-int', u'tap13d2cb29-d6', '--', 'set', 'Interface', u'tap13d2cb29-
↪d6', \
   u'external-ids:iface-id=13d2cb29-d61c-46d9-afe9-98b6aa0a43ea', 'external-ids:iface-
↪status=active', u'external-ids:attached-mac=fa:16:3e:6d:ac:ea', \
   'external-ids:vm-uuid=1065e38c-e6c6-423f-8140-5d0c021d3af0']
2016-01-27 11:37:22.289 12945 ERROR nova.compute.manager [instance: 1065e38c-e6c6-423f-
↪8140-5d0c021d3af0]
```

There are interesting files regarding running instances available on the compute nodes, in `/var/lib/nova/instances`:

```
# tree /var/lib/nova/instances
/var/lib/nova/instances
|-- 54fff47b-fa5e-4401-8309-e2da66c01d66
|   |-- console.log
|   |-- disk
|   |-- disk.info
|   +-- libvirt.xml
|-- 7fb5ce27-cb7a-4a3b-94d8-afb84ddd3c5b
|   |-- console.log
|   |-- disk
|   |-- disk.info
|   +-- libvirt.xml
|-- _base
|   +-- 775fa67e40ab15538f2f01969e50a38078c09e9b
|-- compute_nodes
+-- locks
    |-- nova-775fa67e40ab15538f2f01969e50a38078c09e9b
    +-- nova-storage-registry-lock
```

For instance, you can find the libvirt domain file used to boot the VM:

```
# head /var/lib/nova/instances/54fff47b-fa5e-4401-8309-e2da66c01d66/libvirt.xml
<domain type="kvm">
  <uuid>54fff47b-fa5e-4401-8309-e2da66c01d66</uuid>
  <name>instance-00000003</name>
  <memory>524288</memory>
  <memoryBacking>
    <hugepages>
      <page size="2048" nodeset="0" unit="KiB"/>
    </hugepages>
  </memoryBacking>
  <numatune>
```

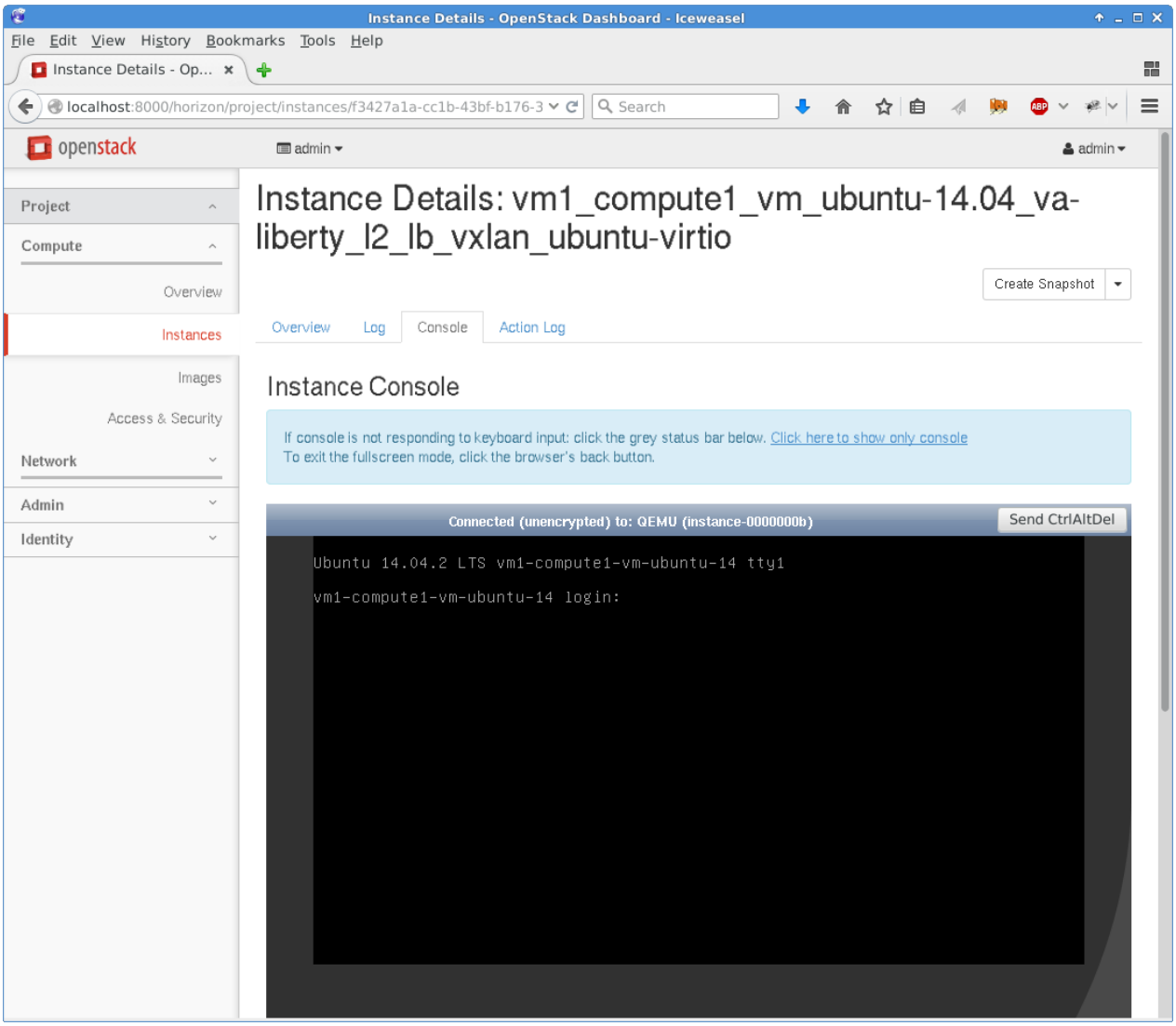

Or you can look at a currently running Nova instance console logs:

```
# tail /var/lib/nova/instances/54fff47b-fa5e-4401-8309-e2da66c01d66/console.log
ec2: #############################################################
-----BEGIN SSH HOST KEY KEYS-----
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYT[...]Osfj0fFcxJvE2Roc= root@compute1-vm
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQ[...]JUYLnAaNV8oNz1AId root@compute1-vm
-----END SSH HOST KEY KEYS-----
Cloud-init v. 0.7.5 finished at Wed, 27 Jan 2016 12:37:49 +0000. Datasource␣
↪DataSourceEc2.
Up 18.67 seconds

Ubuntu 14.04.2 LTS vm1-compute1-vm-ubuntu-14 ttyS0
```

Note:

- For console logs, however, we recommend using `nova console-log <ID>` on the controller node, for a similar result.
- You may also access your VM console itself by accessing horizon (which must be installed and started obviously).

  From the administration panel, access `Compute > Instances > [your instance] > Console`:



If you want Nova to provide you with more information when running, you can configure the `verbose` and `debug` options to `True` in `/etc/nova/nova.conf`:

```
# grep -iE "(verbose|debug)" /etc/nova/nova.conf
verbose = True
debug = True
```

Once configured, restart the `nova-compute` service.

- On Ubuntu Server:

```
# service nova-compute restart
```

- On Red Hat 7:

```
# systemctl restart openstack-nova-compute.service
```

Similarly, if you want Neutron to provide you with more information, configure `verbose` and `debug` options in `/etc/neutron/neutron.conf`:

```
# grep -iE "(verbose|debug)" /etc/neutron/neutron.conf
verbose = True
debug = True
```

Once configured, restart the Neutron service.

Note: Do not keep `verbose` and `debug` options set on production environments, as it is very, very talkative. It makes researching interesting information in the log difficult.

3.6 External Tools

3.6.1 `perf`

For interesting results, `perf` requires the fast path executable to be built with debug info (not stripped).

Use this command to get info on which functions the CPU spends the most time in:

```
# perf top
Samples: 794K of event 'cycles', Event count (approx.): 501635499700
Overhead  Shared Object                 Symbol
  39.71%  fp-rte                        [.] fpn_main_loop
  17.91%  fp-rte                        [.] ixgbe_recv_pkts_lro_bulk_alloc
   8.40%  fp-rte                        [.] fp_ip_input
   7.09%  fp-rte                        [.] ixgbe_xmit_pkts
   2.85%  librte_crypto.so              [.] rte_crypto_poll
   2.66%  fp-rte                        [.] fpn_crypto_generic_poll
   2.31%  fp-rte                        [.] fp_ip_if_send
   2.25%  fp-rte                        [.] fp_ether_input
   2.20%  fp-rte                        [.] fpn_intercore_drain
   2.16%  librte_crypto_multibuffer.so  [.] 0x00000000000052f2
   1.95%  fp-rte                        [.] fp_if_output
   1.79%  fp-rte                        [.] fp_process_input_bulk
   1.68%  fp-rte                        [.] fpn_crypto_poll
   1.60%  ld-2.17.so                    [.] __tls_get_addr
   0.90%  fp-rte                        [.] fpn_recv_exception
   0.89%  fp-rte                        [.] fp_ether_output
   0.66%  librte_crypto_multibuffer.so  [.] 0x00000000000052eb
```

(continues on next page)

(continued from previous page)

```
   0.44%  librte_crypto_multibuffer.so  [.] 0x0000000000005608
   0.34%  librte_crypto_multibuffer.so  [.] 0x00000000000052cc
   0.27%  librte_crypto_multibuffer.so  [.] __tls_get_addr@plt
   0.21%  librte_crypto_multibuffer.so  [.] 0x0000000000005601
```

Note: `perf` can only be used if the fast path is running as a userland process. This is the case for Intel or Arm, but not Octeon, typically.

Refer to the `perf` manpage for specific options.

3.6.2 strace

`strace` displays system calls done by a given program. Use this command to get a first impression on what the program is spending time on. For instance, you can see netlink messages handled by the cache manager:

```
# strace -p $(pidof cmgrd)
Process 5350 attached
setsockopt(11, SOL_SOCKET, SO_SNDBUF, [32768], 4) = 0
setsockopt(11, SOL_SOCKET, SO_RCVBUF, [32768], 4) = 0
bind(11, {sa_family=AF_NETLINK, pid=-2076175130, groups=00000000}, 12) = 0
getsockname(11, {sa_family=AF_NETLINK, pid=-2076175130, groups=00000000}, [12]) = 0
sendmsg(11, {msg_name(12)={sa_family=AF_NETLINK, pid=0, groups=00000000}, msg_iov(1)=[{
↪"\34\0\0\0\20\0\5\0\204\315jV\3
6\24@\204\3\1\0\0\10\0\2\0vrf\0", 28}], msg_controllen=0, msg_flags=0}, 0) = 28
recvmsg(11, {msg_name(12)={sa_family=AF_NETLINK, pid=0, groups=00000000}, msg_iov(1)=[{
↪"\320\0\0\0\20\0\0\0\204\315jV\
46\24@\204\1\2\0\0\10\0\2\0vrf\0\6\0\1\0"..., 16384}], msg_controllen=0, msg_flags=0},␣
↪0) = 208
recvmsg(11, {msg_name(12)={sa_family=AF_NETLINK, pid=0, groups=00000000}, msg_iov(1)=[{
↪"$\0\0\0\2\0\0\0\204\315jV\346\
4@\204\0\0\0\0\34\0\0\0\20\0\5\0\204\315jV"..., 16384}], msg_controllen=0, msg_flags=0}
↪, 0) = 36
sendmsg(11, {msg_name(12)={sa_family=AF_NETLINK, pid=0, groups=00000000}, msg_iov(1)=[{
↪"\24\0\0\0\33\0\5\3\205\315jV\3
6\24@\204\1\0\0\0", 20}], msg_controllen=0, msg_flags=0}, 0) = 20
recvmsg(11, {msg_name(12)={sa_family=AF_NETLINK, pid=0, groups=00000000}, msg_iov(1)=[{
↪",\0\0\0\33\0\2\0\205\315jV\346
24@\204\2\1\0\0\10\0\1\0\0\0\0\0\r\0\2\0"..., 16384}], msg_controllen=0, msg_flags=0},␣
↪0) = 44
epoll_wait(4,
^C
```

(continues on next page)

(continued from previous page)

```
Process 5350 detached
 <detached ...>
```

Note: Refer to the `strace` manpage for specific options.

4. Publicly Available Software List

This document specifies the list of Publicly Available Software (PAS) included in the Virtual Accelerator delivery.

4.1 Processor SDK

4.1.1 6WINDGate DPDK

Drivers and libraries for high performance I/Os on Intel and Arm platforms, based on open source DPDK from dpdk.org.

The fast path component is linked against these DPDK libraries.

Version 20.11

License

- *BSD 3-clause "New" or "Revised" License* (libraries)
- *GNU General Public License v2.0* (kernel modules, dpdk-pmdinfogen tool)

Source code location Included in the 6WIND software delivery.

Summary of changes made to the open source version:

- 6WIND documentation
- Improvement to multiqueue for virtio-user
- LLDP control for i40 driver
- Support to fetch "ethtool" information for i40, ixgbe
- Support to read hw statistics for Mellanox CX-5
- Support to set inner offload for virtio

4.1.2 Mellanox ConnectX-3 EN series PMD

rdma-core

Userspace components for the Linux Kernel *drivers/infiniband* subsystem.

Version v27

License

- *OpenIB.org BSD license (MIT variant)*
- *GNU General Public License v2.0*

Source code location `https://github.com/linux-rdma/rdma-core`

Extracted in `sources/dpdk-pmd-mellanox-rdma-core`

4.1.3 Mellanox ConnectX-4 EN series PMD

rdma-core

Userspace components for the Linux Kernel *drivers/infiniband* subsystem.

Version 16

License

- *OpenIB.org BSD license (MIT variant)*
- *GNU General Public License v2.0*

Source code location Included in the 6WIND software delivery.

4.1.4 Intel Multi-Buffer Crypto

Intel(R) Multi-Buffer Crypto for IPSec

Intel library providing optimized crypto leveraging AES-NI.

The fast path component is linked against this Intel Multi-buffer library.

Version Same as 6WIND software.

License *BSD 3-clause "New" or "Revised" License*

Source code location https://github.com/intel/intel-ipsec-mb

Summary of changes made to the open source version:

- Support to build library outside the source directory

4.2 FPN-SDK

4.2.1 FPN-SDK Baseline

Version Same as 6WIND software.

License

- 6WIND
- *BSD 3-clause "New" or "Revised" License*
- *BSD 4-Clause "Original" or "Old" License*
- Public Domain

Extracted in `sources/fpn-sdk`

4.3 Fast Path Baseline

Version Same as 6WIND software.

License

- 6WIND
- *BSD 2-clause "Simplified" License*
- *BSD 3-clause "New" or "Revised" License*
- *BSD 4-Clause "Original" or "Old" License*
- *ISC License*

Extracted in `sources/fp`

4.3.1 Fast Path Flow Inspection / Packet Capture module

Fast Path Flow Inspection / Packet Capture provides the ability to analyze packets coming in and out of the fast path interfaces (*a la tcpdump*).

Version Same as 6WIND software.

License *BSD 3-clause "New" or "Revised" License*

Extracted in `sources/fp`

4.4 Linux - Fast Path Synchronization

4.4.1 Linux - Fast Path Synchronization

Cache manager daemon implements the listener of Linux networking events.

It includes Linux kernel header UAPI, under license GPL with the exception Linux-syscall-note.

Version Same as 6WIND software.

License

- 6WIND
- *GNU General Public License v2.0* WITH *Linux Syscall Note*
- *GNU General Public License v2.0*

Extracted in `sources/linux-fp-sync`

4.4.2 Linux - Fast Path Synchronization Extension for FPTUN eBPF

The fptun ebpf module transfers the Fast Path messages to the linux kernel.

It includes Linux kernel header UAPI, under license GPL with the exception Linux-syscall-note.

Version Same as 6WIND software.

License

- 6WIND
- *GNU General Public License v2.0* WITH *Linux Syscall Note*

Extracted in `sources/linux-fp-sync-fptun-ebpf`

4.5 Control Plane

4.5.1 Security - IKE (v1 and v2) - strongSwan

Open source IPsec-based VPN solution.

Version 5.9.1

License *GNU General Public License v2.0* or later

Source code location http://download.strongswan.org/; 6WIND patches available on request.

Summary of changes made to the open source version:

- 6WIND documentation

- Bug fixes
- Support to encapsulate and decapsulate DSCP
- Support to enable Path MTU
- Support for automatic route installation
- Support for High Availability

4.5.2 Routing - FRR (https://frrouting.org/)

Open source routing software suite.

Version 7.3.1

License

- *GNU General Public License v2.0* or later
- *GNU Lesser General Public License v2.1* (files copied from glibc project https://www.gnu.org/software/libc/)

Source code location https://github.com/FRRouting/frr/releases/; 6WIND patches available on request.

Summary of changes made to the open source version:

- 6WIND documentation
- Bug fixes
- Improvement to VRF using network namespaces
- Improvement to BGP flow spec
- Add LDP user documentation
- Support for adding the same route in different tables
- Support for ECMP using ‘staticd’ daemon

4.5.3 rtrlib

Version 0.7.0

License *MIT License*

Source code location http://rtrlib.realmv6.org/

4.5.4 OVS - Open vSwitch

Open source virtual switch.

Version 2.13

License See COPYING (https://github.com/openvswitch/ovs/blob/branch-2.4/COPYING).

Source code location http://openvswitch.org; 6WIND patches available on request.

4.6 High Availability

4.6.1 VRRP - keepalived

Open source VRRP solution.

Version 2.0.10

License

- 6WIND
- *GNU General Public License v2.0*

Source code location https://github.com/acassen/keepalived/releases/; 6WIND patches available on request.

4.7 Management

4.7.1 Netopeer2

Version 1.1.44

License *BSD 3-clause "New" or "Revised" License*

Source code location https://github.com/CESNET/Netopeer2

4.7.2 libnetconf2

Version 1.1.32

License *BSD 3-clause "New" or "Revised" License*

Source code location https://github.com/CESNET/libnetconf2

4.7.3 libyang

Version 1.0.198

License *BSD 3-clause "New" or "Revised" License*

Source code location https://github.com/CESNET/libyang

4.7.4 sysrepo

Version 1.4.87

License *Apache License 2.0*

Source code location Included in the 6WIND software delivery.

4.7.5 Telegraf

Version 1.15.3

License *MIT License*

Source code location https://github.com/influxdata/telegraf

4.7.6 kpi-tools bundled Python libraries

asn1crypto

Fast ASN.1 parser and serializer with definitions for private keys, public keys, certificates, CRL, OCSP, CMS, PKCS#3, PKCS#7, PKCS#8, PKCS#12, PKCS#5, X.509 and TSP.

Version 0.24.0

License *MIT License*

Source code location https://github.com/wbond/asn1crypto

bcrypt

Modern password hashing for your software and your servers.

Version 3.1.4

License *Apache License 2.0*

Source code location https://github.com/pyca/bcrypt/

cffi

Foreign Function Interface for Python calling C code.

Version 1.14.2

License *MIT License*

Source code location https://bitbucket.org/cffi/cffi

cryptography

Provides cryptographic recipes and primitives to Python developers.

Version 2.1.4

License *Apache License 2.0*

Source code location https://github.com/pyca/cryptography

idna

Internationalized Domain Names in Applications (IDNA).

Version 2.6

License *BSD 3-clause "New" or "Revised" License*

Source code location https://github.com/kjd/idna

libyang-python

Python bindings to libyang.

Version 1.5.0

License *MIT License*

Source code location https://github.com/CESNET/libyang-python

lxml

Pythonic binding for the libxml2 and libxslt libraries.

Version 4.2.0

License *BSD 3-clause "New" or "Revised" License* with exceptions

Source code location http://lxml.de/files/lxml-4.2.0.tgz

ncclient

Version v0.5.3

License *Apache License 2.0*

Source code location https://github.com/ncclient/ncclient

paramiko

SSH2 protocol library.

Version 2.4.1

License *GNU Lesser General Public License v2.1*

Source code location https://github.com/paramiko/paramiko/

psutil

Cross-platform lib for process and system monitoring in Python.

Version 5.4.7

License *BSD 3-clause "New" or "Revised" License*

Source code location https://github.com/giampaolo/psutil

pyasn1

ASN.1 types and codecs.

Version 0.4.2

License *BSD 2-clause "Simplified" License*

Source code location https://github.com/etingof/pyasn1

pycparser

C parser in Python.

Version 2.18

License *BSD 3-clause "New" or "Revised" License*

Source code location https://github.com/eliben/pycparser

PyNaCl

Python binding to the Networking and Cryptography (NaCl) library.

Version 1.2.1

License *Apache License 2.0*

Source code location https://github.com/pyca/pynacl/

setuptools

Easily download, build, install, upgrade, and uninstall Python packages.

Version 38.5.2

License *MIT License*

Source code location https://github.com/pypa/setuptools

six

Python 2 and 3 compatibility utilities.

Version 1.11.0

License *MIT License*

Source code location http://pypi.python.org/pypi/six/

sysrepo-python

Python bindings to sysrepo.

Version 0.2.0

License *BSD 3-clause "New" or "Revised" License*

Source code location https://github.com/sysrepo/sysrepo-python

xmltodict

Makes working with XML feel like you are working with JSON.

Version 0.11.0

License *MIT License*

Source code location https://pypi.python.org/packages/57/17/a6acddc5f5993ea6eaf792b2e6c3be55e3e11f3b85206c81857258
xmltodict-0.11.0.tar.gz

4.8 Tools

4.8.1 6WINDGate Build Framework

Config tool

The config tool is used only on the host system to configure the build. Not embedded with binaries.

Version 2.6.38

License *GNU General Public License v2.0*

Source code location https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/scripts?h=v2.6.38

nasm

Host tool to build Intel Multi-Buffer Library. Not embedded with binaries.

Version 2.14.02

License *BSD 2-clause "Simplified" License*

Source code location http://www.nasm.us/pub/nasm/releasebuilds/2.14.02/

4.9 6WINDGate Common Libraries

4.9.1 Pyroute2

Pyroute2 is a pure Python netlink and Linux network configuration library.

This library is used in scripts to configure the fast path runtime options.

Version 0.5.0.ad7bb665cbcd

License *GNU General Public License v2.0*

Source code location https://pypi.python.org/packages/91/e7/814f60e355078dc51625cd2e7e715ed4a06111ddf2ac5580f2f10e
pyroute2-0.4.13.tar.gz

4.9.2 iproute2

The iproute2 suite is a collection of utilities for networking and traffic control. These tools communicate with the Linux kernel via the (rt)netlink interface, providing advanced features not available through the legacy net-tools commands 'ifconfig' and 'route'.

Version 5.1.0

License

- 6WIND
- *GNU General Public License v2.0*

Source code location https://www.kernel.org/pub/linux/utils/net/iproute2/iproute2-5.1.0.tar.gz

4.9.3 libconsole

This library helps a daemon to set up a console.

Version Same as 6WIND software.

License *BSD 3-clause "New" or "Revised" License*

Extracted in `sources/tools-common-libs-libconsole`

4.10 Licenses List

Here is the list of licenses used in the product.

4.10.1 Apache License 2.0

Apache License Version 2.0, January 2004 http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

    (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof,

You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

4.10.2 BSD 2-clause "Simplified" License

Copyright (c) <year> <owner> All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

4.10.3 BSD 3-clause “New” or “Revised” License

Copyright (c) <year> <owner>. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

4.10.4 BSD 4-Clause “Original” or “Old” License

Copyright (c) <year> <owner>. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the organization.

4. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY COPYRIGHT HOLDER “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT

SHALL COPYRIGHT HOLDER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

4.10.5 OpenIB.org BSD license (MIT variant)

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

4.10.6 GNU General Public License v2.0

GNU GENERAL PUBLIC LICENSE Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

    a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though

third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> one line to give the program's name and an idea of what it does. Copyright (C) yyyy name of author
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA. Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’. This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.

signature of Ty Coon, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

4.10.7 ISC License

Copyright (c) 2004-2010 by Internet Systems Consortium, Inc. (“ISC”) Copyright (c) 1995-2003 by Internet Software Consortium

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED “AS IS” AND ISC DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL ISC BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

4.10.8 Intel Open Source License

Copyright (c) <year> Intel Corporation All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE INTEL OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

EXPORT LAWS: THIS LICENSE ADDS NO RESTRICTIONS TO THE EXPORT LAWS OF YOUR JURISDICTION. It is licensee's responsibility to comply with any export regulations applicable in licensee's jurisdiction. Under CURRENT (May 2000) U.S. export regulations this software is eligible for export from the U.S. and can be downloaded by or otherwise exported or reexported worldwide EXCEPT to U.S. embargoed destinations which include Cuba, Iraq, Libya, North Korea, Iran, Syria, Sudan, Afghanistan and any other country to which the U.S. has embargoed goods and services.

4.10.9 GNU Lesser General Public License v2.1

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages–typically libraries–of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

   a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

   b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES

OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

one line to give the library's name and an idea of what it does. Copyright (C) year name of author

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

signature of Ty Coon, 1 April 1990 Ty Coon, President of Vice That's all there is to it!

4.10.10 Linux Syscall Note

NOTE! This copyright does *not* cover user programs that use kernel services by normal system calls - this is merely considered normal use of the kernel, and does *not* fall under the heading of "derived work". Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the Linux kernel) is copyrighted by me and others who actually wrote it.

Also note that the only valid version of the GPL as far as the kernel is concerned is _this_ particular version of the license (ie v2, not v2.2 or v3.x or whatever), unless explicitly otherwise stated.

Linus Torvalds

4.10.11 MIT License

Copyright (c) <year> <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

