Notice

The information in this document is provided without warranty of any kind and is subject to change without notice. 6WIND S.A. assumes no responsibility, and shall have no liability of any kind arising from supply or use of this publication or any material contained herein.

©2019, 6WIND S.A. All rights reserved. Company and product names are trademarks or registered trademarks of their respective companies.

No part of this publication may be reproduced, photocopied, or transmitted without express, written consent of 6WIND S.A.
Contents

1. Overview 2

2. Installation and basic configuration 3

3. Use case: NAT444 3
   3.1. Overview 3
   3.2. Configuration 4
      3.2.1. Network Topology 4
      3.2.2. Interfaces configuration 4
      3.2.3. Routing configuration 5
      3.2.4. CG-NAT configuration 5
         3.2.4.1. Pool 5
         3.2.4.2. Rule 5
            3.2.4.2.1. Port algorithm 6
            3.2.4.2.2. Endpoint mapping 6
            3.2.4.2.3. Endpoint filtering 7
            3.2.4.2.4. Hairpinning 8
      3.2.5. Changing parameters 8
   3.3. Understanding Port Block Allocation 9
   3.4. Status 11
      3.4.1. State 11
      3.4.2. Statistics 12
      3.4.3. Listing users 13
   3.5. Monitoring with Grafana 14
   3.6. Logging 14
      3.6.1. On the console 14
      3.6.2. Towards an external framework 15
   3.7. Troubleshooting 19
      3.7.1. Invalid packet state statistics 19
      3.7.2. State/NAT/USER/Block Allocation Failures 20
      3.7.3. No IP Public 20
      3.7.4. NAT port allocation failures 21
         3.7.4.1. Maximum number of blocks reached 22
         3.7.4.2. Full IP Public 22
   3.8. Capability tuning 23
   3.9. Limitations 24

© 2019, 6WIND S.A.
1. Overview

The purpose of this document is to guide the user in deploying the vRouter for a CG-NAT use case. It focuses on the concepts that are relevant to this specific use case, in order to provide a practical example. Exhaustive documentation of the vRouter features that are not covered in the use case can be found in the standard vRouter documentation.

2. Installation and basic configuration

Follow the Getting Started guide to install the software in your environment and get a remote console with SSH.

3. Use case: NAT444

3.1. Overview

One approach to cope with the public IPv4 address exhaustion is to share the remaining or available IPv4 addresses among a larger number of customers. It can be done by using CG-NAT.

CG-NAT, also known as Large Scale NAT, is a highly scalable NAT placed in the ISP core network, between the customer premises equipment (CPE) and the Internet.

In NAT44(4), there are three IPv4 networks:

- A private IPv4 network within the user network (behind the CPE),
- A different private IPv4 network for the user to provider links (between the CPE and the vRouter), known as the Shared address space,
- A public IPv4 network on the outside of the vRouter.
3.2. Configuration

3.2.1. Network Topology

For this use case, we consider the following topology:

3.2.2. Interfaces configuration

Allocate the ports that will be involved in data plane processing to the fast path:

```
vrouter> edit running
vrouter running config# / system fast-path
vrouter running fast-path#! port pci-b0s4
vrouter running fast-path# port pci-b0s5
```

All physical and logical interfaces are configured under the **main** VRF in this example.

```
vrouter running config# vrf main
```

Create Ethernet interfaces, attach them to a port of a NIC and configure IP addresses:

```
vrouter running vrf main# interface physical lan
vrouter running physical lan#! port pci-b0s4
vrouter running physical lan# ipv4 address 8.0.0.1/24
vrouter running physical lan# .. physical wan
vrouter running physical wan#! port pci-b0s5
vrouter running physical wan# ipv4 address 9.0.0.1/24
```

See the User's Guide for more information regarding:

- [CLI basics](#)
- [fast path configuration](#)
- [interfaces configuration](#)
3.2.3. Routing configuration

Configure routes towards the LAN and WAN, plus a blackhole route to drop the incoming public traffic that doesn’t match an existing connection.

```
vrouter running physical wan# / vrf main routing static
vrouter running static# ipv4-route 100.64.0.0/10 next-hop 8.0.0.2
vrouter running static# ipv4-route 32.96.118.0/24 next-hop 9.0.0.2
vrouter running static# ipv4-route 32.96.119.0/24 next-hop blackhole
```

See the User’s Guide for more information regarding:
- Routing configuration

3.2.4. CG-NAT configuration

3.2.4.1. Pool

A CG-NAT pool contains a list of IPv4 addresses used to change the IPv4 source address and port of a packet.

The vRouter implements a feature called Port Block Allocation. Each time a new user sends a packet through the vRouter, a block of ports is allocated to the user from one of the IP addresses in the pool. Each public IP is divided into blocks of ports, whose size and range is defined in the pool configuration. For more information about Port Block Allocation, see below.

Here is an example of pool configuration:

```
vrouter running static# / vrf main cg-nat
vrouter running cg-nat#! pool mypool
vrouter running pool mypool#! address 32.96.119.0/24
vrouter running pool mypool#! port-range 1024 65535
vrouter running pool mypool#! block-size 512
```

**Note**  The ! in the prompt indicates that the current configuration is invalid. This is because a rule is required to complete the CG-NAT configuration.

3.2.4.2. Rule

A CG-NAT rule defines the matching criteria to NAT packets and the pool to use to translate them, replacing the source IP address and port of the packet with an IP address from the pool and a port from the range. It also specifies the number of blocks from the pool to associate to each user.
Here is an example of rule configuration:

```
vrouter running pool mypool#1. rule 1
vrouter running rule 1#1 match
vrouter running match#1 source address 100.64.0.0/10
vrouter running match#1 outbound-interface wan
vrouter running match#1 translate-to
vrouter running translate-to#1 pool-name mypool
vrouter running translate-to#1 max-blocks-per-user 4
```

The ! in the prompt has disappeared, meaning that the configuration is now valid. It can be committed.

```
vrouter running translate-to# commit
Configuration committed.
```

3.2.4.2.1. Port algorithm

The port-algo commands allows choosing between two methods to allocate ports:

- **parity** (default): This algorithm preserves the parity of the port, i.e. an even port will be mapped to an even port, and an odd port will be mapped to an odd port.
- **random**: This algorithm chooses a port randomly.

Here is an example to configure random port allocation:

```
vrouter running translate-to# port-algo random
vrouter running translate-to# commit
```

3.2.4.2.2. Endpoint mapping

There are two endpoint mapping behaviors:

- **independent** (default): The vRouter reuses the same port mapping for subsequent packets sent from the same internal IP address and port to any external IP address and port.

![Diagram showing independent endpoint mapping]

© 2019, 6WIND S.A.
- **dependent**: The vRouter reuses the same port mapping for subsequent packets sent from the same internal IP address and port to the same external IP address and port.

Here is an example to configure dependent endpoint mapping:

```
  vrouter running translate-to# endpoint-mapping dependent
  vrouter running translate-to# commit
```

3.2.4.2.3. Endpoint filtering

There are two endpoint filtering behaviors:

- **independent** (default): Inbound packets from external endpoints are only filtered out if their destination IP address and port don’t match an existing public IP address and port mapping.

Here is an example to configure independent endpoint filtering:

```
  vrouter running translate-to# endpoint-mapping independent
  vrouter running translate-to# commit
```
• **dependent**: Inbound packets from external endpoints are filtered out if they don’t match an existing mapping (source and destination IPs and ports, protocol).

![Diagram of dependent endpoint filtering]

dependent endpoint-filtering

Here is an example to configure dependent endpoint filtering:

```
vrouter running translate-to# endpoint-filtering dependent
vrouter running translate-to# commit
```

3.2.4.2.4. Hairpinning

The hairpinning feature allows two endpoints (user 1 and user 2) on the private network to communicate together using their public IP addresses and ports.

By default, the hairpinning feature is disabled. To enable it, use the following command:

```
vrouter running translate-to# hairpinning true
vrouter running translate-to# commit
```

3.2.5. Changing parameters

Changing some configuration parameters requires to flush users, as summarized in the following table.

<table>
<thead>
<tr>
<th>Category</th>
<th>Parameter</th>
<th>Flush required</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>address</td>
<td>no</td>
<td>New addresses can be added without any impact. If an address is removed, all users assigned to it are flushed.</td>
</tr>
<tr>
<td></td>
<td>block-size</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>port-range</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>rule</td>
<td>match</td>
<td>no</td>
<td>Conntracks not matching the new match criteria are not flushed.</td>
</tr>
</tbody>
</table>
### 3.3. Understanding Port Block Allocation

It is a legal requirement for an ISP to be able to provide the mapping between a user and a public IP address at a given point in time. With classic NAT, it means that each new user session has to be logged. This is obviously not scalable. Additionally with classic NAT, a user can consume all the ports of the public IP.

To reduce the amount of logs and equitably share the ports of the different public IPs, CG-NAT provides the Port Block Allocation (PBA) feature that consists in allocating blocks of ports to each user. As logging is done per block of ports, the amount of logs is reduced. And

#### Flushing a User

Flushing a user is done thanks to the following command:

```bash
vrouter> flush cg-nat user rule-id 1 user-address 100.64.0.1
```

To flush all users of a rule, use the following command:

```bash
vrouter> flush cg-nat user rule-id 1
```

**Note**  Flushing a user resets the associated sessions, resulting in a service interruption.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool-name</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>max-blocks-per-users</td>
<td>no</td>
<td>Extra blocks of users having more blocks than the new max are not flushed, but they cannot be used anymore for new sessions. They are released when all sessions are released.</td>
</tr>
<tr>
<td>active-block-timeout</td>
<td>no</td>
<td>The new timeout value will be taken into account after the current timeout happens.</td>
</tr>
<tr>
<td>user-timeout</td>
<td>no</td>
<td>The new timeout value will be taken into account for a user the next time all his sessions expire.</td>
</tr>
<tr>
<td>port-algo</td>
<td>no</td>
<td>Taken into account for new blocks only.</td>
</tr>
<tr>
<td>endpoint-mapping</td>
<td>no</td>
<td>Taken into account for new mappings only.</td>
</tr>
<tr>
<td>endpoint-filtering</td>
<td>no</td>
<td>Taken into account for new mappings only.</td>
</tr>
<tr>
<td>hairpinning</td>
<td>no</td>
<td>Taken into account for new mappings only.</td>
</tr>
</tbody>
</table>

© 2019, 6WIND S.A.
as the number and size of the blocks can be configured, the user port consumption is controlled. Here is how PBA works.

Port Block Allocation operation

Each time a new user sends a packet through the vRouter, a block of ports is allocated to him from one of the IP addresses in the pool. The public IPs are selected using a round-robin algorithm. Each public IP is divided into blocks of ports, whose size and range is defined in the pool configuration. This prevents a single user from consuming all ports.

Here is an example to change the number of ports per block:

```
vrouter running cg-nat# / vrf main cg-nat pool mypool block-size 256
vrouter running cg-nat# commit
```

For the next session of the same user, a port is allocated from its block of ports, until the block is exhausted. In that case, a new block can be allocated for this user and it becomes the active block. There is only one active block per user. The maximum number of blocks per user is defined in the rule configuration.

Here is an example to change the maximum number of blocks per user:

```
vrouter running cg-nat# / vrf main cg-nat rule 1 translate-to max-blocks-per-user 2
vrouter running cg-nat# commit
```

Since ports are allocated from the same block (until this one is empty), port prediction can potentially happen. To randomize port allocation, it is possible to allocate a new active block if the current active block has been active for too long, even if there are still some ports available in the active block. This feature is called active block timeout. As it can increase the average numbers of blocks allocated per user, it is disabled by default.
Here is an example to configure an active block timeout of 60 seconds:

```plaintext
vrouter running cg-nat# / vrf main cg-nat rule 1 translate-to active-block-timeout 60
vrouter running cg-nat# commit
```

When all the ports are released from a non-active block, this one is released immediately. Regarding the active block, the block is only released when the user subscription times out. The default user timeout is 120 seconds.

Here is an example to change the user timeout:

```plaintext
vrouter running cg-nat# / vrf main cg-nat rule 1 translate-to user-timeout 180
vrouter running cg-nat# commit
```

### 3.4. Status

#### 3.4.1. State

To review the CG-NAT state, use the following command:

```plaintext
vrouter> show state / vrf main cg-nat
cg-nat
    enabled true
    pool mypool
        address 32.96.119.1-32.96.119.254
        block-size 256
        port-range 1024 65535
        ..
    rule 1
        match
            source
                address 100.64.0.0/10
                ..
            outbound-interface wan
                ..
        translate-to
            pool-name mypool
            max-blocks-per-user 2
            active-block-timeout 0
            user-timeout 180
            port-algo random
            endpoint-mapping dependent
            endpoint-filtering dependent
            hairpinning false
            ..
            ..
        options
```
conntrack
  behavior tcp-window-check enabled true
  behavior tcp-rst-strict-order enabled true
  timeouts
    icmp closed 0
    icmp new 30
    icmp established 60
    udp closed 0
    udp new 30
    udp established 120
    gre-pptp closed 0
    gre-pptp new 600
    gre-pptp established 18000
    tcp syn-sent 30
    tcp simsyn-sent 30
    tcp syn-received 60
    tcp established 7440
    tcp fin-sent 120
    tcp fin-received 120
    tcp close-wait 60
    tcp fin-wait 120
    tcp last-ack 30
    tcp time-wait 120
    tcp closed 10
  ...
  ...
logging
  enabled false
  ...
  ...

3.4.2. Statistics

To display the CG-NAT statistics, the following command can be used:

vrouter> show cg-nat statistics
Packets passed:
  0 default pass
  33317355 ruleset pass
  260836153 state pass
Packets blocked:
  0 default block
  0 ruleset block
Hairpinning Stats:
  0 hairpin packets
  0 loop-hairpin drop
State and NAT entries:

- 33077173 state allocations
- 0 state reverse
- 39496338 state destructions
- 0 state allocation failures
- 9726101 NAT entry allocations
- 13127681 NAT entry destructions
- 0 NAT entry allocation failures
- 0 NAT port allocation failures

CGNat entries:

- 0 USER allocations
- 20000 USER destructions
- 0 USER allocation failures
- 120000 Block allocations
- 180000 Block destructions
- 0 Block allocation failures
- 0 No IP Public
- 0 Full IP Public

NAT64 Stats:

- 0 udp null checksum packet drops

Invalid packet state cases:

- 1310 cases in total
- 1310 TCP case invalid first packet
- 0 TCP case RST
- 1310 TCP case invalid transition
- 0 TCP case I
- 0 TCP case II
- 0 TCP case III

Packet race cases:

- 0 USER association race
- 0 USER creation race
- 0 NAT association race
- 0 duplicate state race

State/NAT/BLOCK/USER allocation statistics increase when the vRouter processes traffic properly.

3.4.3. Listing users

The following command can be used to list the current users of the CG-NAT:

```
100.64.0.1 -> 32.96.120.54
```

- 1/2 tcp blocks, 0/2 udp blocks, 0/2 icmp blocks, 0/2 gre blocks
- 0 no port errors, 0 no block errors, 0 full public ip errors

For each user, we can see how many port blocks are used.
The different possible errors are:

- **no port**: A new session has been rejected because no ports were available in the active block.
- **no block**: A new session has been rejected because no blocks are available in the block memory pool.
- **full public IP**: A new session has been rejected because the public IP allocated to this user doesn’t have any more blocks available.

### 3.5. Monitoring with Grafana

Here we will show how to export KPIs to a time-series database which can then be used with a graphical tool like Grafana. This assumes that InfluxDB and Grafana have been installed on 172.16.0.2 following [this documentation](#).

```bash
vrouter> edit running
vrouter running config# / vrf main interface physical mgt
vrouter running physical mgt# / port pci-b0s6
vrouter running physical mgt# / ipv4 address 172.16.0.1/24
vrouter running physical mgt# / system kpi enabled true
vrouter running physical mgt# / vrf main kpi
vrouter running kpi# telegraf influxdb-output url http://172.16.0.2:8086 database telegraf
vrouter running kpi# interface lan
vrouter running kpi# interface wan
vrouter running kpi# commit
```

See the User’s Guide for more information regarding:

- **KPI**

For information on installing Grafana, please visit:

- [6WIND Grafana Setup on github](#)

### 3.6. Logging

#### 3.6.1. On the console

To enable logs, use the following command:

```bash
vrouter running config# vrf main cg-nat logging enabled true
vrouter running config# commit
```

This command displays the CG-NAT logs on the console:

```bash
vrouter running config# show log service cg-nat
Jun 11 08:02:46 vrouter systemd[1]: Started Fast Path cgnat log daemon.
Jun 11 08:02:46 vrouter fp-cgnat-logd[4269]: CGNAT Log listen on 5001
```
See the User’s Guide for more information regarding:
  ●  Logging

3.6.2. Towards an external framework

In this section, we will explain how to export CG-NAT logs to an external logging framework. As an example, we will use Logstash and Kibana from the Elastic Stack to gather the logs and display them in a user-friendly way.

We assume that Elastic, Logstash and Kibana have been installed on a server accessible on the 172.16.0.2 IP address, following the Elastic documentation.

Here is the Logstash configuration, including the IP address and port of the syslog server and filters to parse and format the CG-NAT log messages before storing them in Elastic.

```ruby
input {
  udp {
    host => "172.16.0.2"
    port => 10514
    type => syslog
  }
}
filter {
  if [type] == "syslog" {
    grok |
    match => { "message" => "%{POSINT:syslog_pri}>%{SYLOGTIMESTAMP:syslog_timestamp} %{SYSLOGHOST:syslog_hostname} %{DATA:syslog_program}%{DATA:syslog_id}?:%{POSINT:syslog_pid}:%{GREEDYDATA:syslog_message}" }
    add_field => [ "received_at", "@timestamp" ]
    add_field => [ "received_from", "%{host}" ]
    date {
      match => [ "syslog_timestamp", "MMM d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
  }
}
# Second level of filtering specific for CG-NAT logs
filter {
  if [type] == "syslog" {
    if [syslog_program] == "fp-cgnat-logd" {
      grok |
      match => [ "message", "USER %{IP:prv_ip} (matching rule %{POSINT:rule}): NEW BLOCK (pool "mypool", ip public 32.96.119.1, proto 6, port 1024 - 1536) at Tue Jun 11 08:03:09 2019"
    }
    if "_grokparsefailure" in [tags] { drop {} }
  }
```
On the vRouter, logging to Logstash is enabled using the following command:

```
vrouter running config# / vrf main logging syslog remote-server 172.16.0.2
vrouter running remote-server 172.16.0.2# protocol udp
vrouter running remote-server 172.16.0.2# port 10514
vrouter running remote-server 172.16.0.2# commit
```
Let’s now connect to Kibana using a web browser, pointing at http://172.16.0.2:5601. Click Management, Index Patterns, type logstash in the Index pattern text box, then click Next step.

Select action_date as the Time Filter field name and click Create index pattern.
A logstash indice now appears in the Elasticsearch Index Management page:

Click on Discover in the left menu; some logs are now displayed in the Kibana dashboard.
The final step is to clean the logs output. Add the following available fields: action, prv_ip, sproto, pub_ip, start_port, end_port.

Search can now be used to filter the logs on a public IP and port, for example using "pub_ip:32.96.119.1 AND start_port:[* TO 1200] AND end_port:[1200 TO *]" to search public IP 32.96.119.1 and port 1200.
3.7. Troubleshooting

3.7.1. Invalid packet state statistics
To display the CG-NAT statistics, use the following command:

```bash
vrouter> show cg-nat statistics
...
Invalid packet state cases:
...
  0 TCP case RST
...
  0 TCP case I
  0 TCP case II
  0 TCP case III
...
```

If the TCP case I, II or III statistics are incremented, disable TCP window checks as follows:

```bash
vrouter> edit running
vrouter running config# vrf main cg-nat options conntrack
vrouter running conntrack# behavior tcp-window-check enabled false
vrouter running conntrack# commit
```

If the TCP case RST statistic is incremented, disable TCP RST strict ordering as follows:

```bash
vrouter> edit running
vrouter running config# vrf main cg-nat options conntrack
vrouter running conntrack# behavior tcp-rst-strict-order enabled false
vrouter running conntrack# commit
```

3.7.2. State/NAT/USER/Block Allocation Failures

```bash
vrouter> show cg-nat statistics
...
State and NAT entries:
...
  0 state allocation failures
...
  0 NAT entry allocation failures
  0 NAT port allocation failures
CGNat entries:
...
  0 USER allocation failures
...
  0 Block allocation failures
```
If one of these statistics is incremented, it means that one of the memory pools of the vRouter is full. Memory pool usage can be dumped using the following command:

```
vrouter> show cg-nat mempool-usage
cgnat_user_pool : 2000/10000 (20.00%)
cgnat_block_pool : 8000/80000 (10.00%)
table_pool : 0/1056 (0.00%)
conn_pool : 1056736/1056736 (100.00%)
nat_pool : 1056736/1056736 (100.00%)
```

In the example above, the connection and NAT memory pools are full. Their size must be increased as follows:

```
vrouter running config# / system fast-path limits cg-nat
vrouter running cg-nat# max-conntracks 2000000
vrouter running cg-nat# max-nat-entries 2000000
vrouter running cg-nat# commit
```

Refer to the capability tuning section.

### 3.7.3. No IP Public

```
vrouter> show cg-nat statistics
…
CGNat entries:
…
  0 No IP Public
…
```

If this statistic is incremented, it means there are no blocks available in any public IP. This can be checked using the following command:

```
vrouter> show cg-nat pool-usage pool-name mypool
tcp block usage: 4095/4095 (100.0%)
udp block usage: 4095/4095 (100.0%)
icmp block usage: 4095/4095 (100.0%)
gre block usage: 4095/4095 (100.0%)
```

To solve this issue, add a new public IP to the pool using the following command:

```
vrouter> edit running
vrouter running config# vrf main cg-nat pool mypool
vrouter running pool mypool# address 32.96.120.0/24
vrouter running pool mypool# commit
```
3.7.4. NAT port allocation failures

There are two main reasons for port allocation failures:

- A user has consumed all its port blocks. The maximum number of blocks per user can be increased in the rule using the `max-blocks-per-user` command.
- No blocks are available on the public IP allocated to the user. In this case, the Full IP Public statistic is also incremented.

To list users with allocation failures to understand how many users are impacted, use the following command:

```
vrouter> show cg-nat user rule-id 1 threshold-errors 1
100.64.0.1 -> 32.96.119.108
  2/2 tcp blocks, 0/2 udp blocks, 0/2 icmp blocks, 0/2 gre blocks
  63 no port errors, 0 no block errors, 0 full public ip errors
```

To understand why a specific user has many connections, use the following command:

```
vrouter> show cg-nat conntracks rule-id 1 user-address 100.64.0.1
CON:
  vrfid 0 flags 0x6 alg none tsdiff 47 timeout 120
  forw proto 6 100.64.0.1:1024-> 32.96.118.2:6001 hash:be3505a5
  back proto 6 32.96.118.2:6001-> 32.96.119.108:1216 hash:92e65736
  state 10:
    F { end 0 maxend 0 mwin 0 wscale 0 flags 1}
    T { end 0 maxend 0 mwin 0 wscale 0 flags 0}
  NAT: original address 100.64.0.1 proto 6 oport 1024 tport 1216
CON:
  vrfid 0 flags 0x6 alg none tsdiff 56 timeout 120
  forw proto 6 100.64.0.1:65024-> 32.96.118.2:6000 hash:913f8bf7
  back proto 6 32.96.118.2:6000-> 32.96.119.108:1024 hash:27051895
  state 10:
    F { end 0 maxend 0 mwin 0 wscale 0 flags 1}
    T { end 0 maxend 0 mwin 0 wscale 0 flags 0}
  NAT: original address 100.64.0.1 proto 6 oport 65024 tport 1024
...```

3.7.4.1. Maximum number of blocks reached

If the maximum number of blocks is reached, it probably means that you have not allocated enough blocks per user. You can collect some statistics to get average/percentile block and port usage of all users with the following commands:

```
vrouter> show cg-nat block-statistics rule-id 1
block-usage:
  1 user (with > 1 block = 1, ratio 100.00%) blocks per user: min = 2, max = 2, average = 2.00
```
Then, you can decide to increase the number of blocks per user or the block size. Refer to the Changing parameters section.

3.7.4.2. Full IP Public

The paired address pooling feature ensures the assignment of the same public IP address to all sessions originating from the same internal user, as described in RFC 4787 Req 2.

It means that when a user has started to use one public IP address, all its port blocks will be allocated from this same IP. Adding a new public IP to the pool won’t solve the issue, as the user cannot allocate a block from a new public IP.

A possible way to recover such situation is to add new IP address to the pool, and then flush all the current connections of all users, as follows:

```
vrouter running config# / vrf main cg-nat pool mypool
vrouter running pool mypool# address 32.96.120.0/24
vrouter running pool mypool# commit
Configuration committed.

vrouter running pool mypool# flush cg-nat user rule-id 1
```
3.8. Dimensioning

The maximum numbers for NAT entries, CPEs (users), conntracks (sessions), blocks and block sizes are defined in the configuration. These capabilities can be adjusted to adapt to the amount of memory available in the system.

The following table shows a list of different capability combinations and the corresponding memory requirement. This is empirical and may have to be tuned according to your use case.

<table>
<thead>
<tr>
<th>Max conntracks</th>
<th>Max nat entries</th>
<th>Max cpe</th>
<th>Max blocks</th>
<th>Required memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1M</td>
<td>1M</td>
<td>10K</td>
<td>80K</td>
<td>5 GB</td>
</tr>
<tr>
<td>2M</td>
<td>2M</td>
<td>20K</td>
<td>80K</td>
<td>6 GB</td>
</tr>
<tr>
<td>4M</td>
<td>4M</td>
<td>20K</td>
<td>80K</td>
<td>8 GB</td>
</tr>
<tr>
<td>8M</td>
<td>8M</td>
<td>20K</td>
<td>80K</td>
<td>12 GB</td>
</tr>
<tr>
<td>16M</td>
<td>16M</td>
<td>20K</td>
<td>80K</td>
<td>24 GB</td>
</tr>
<tr>
<td>30M</td>
<td>30M</td>
<td>20K</td>
<td>80K</td>
<td>32 GB</td>
</tr>
</tbody>
</table>

Here is an example to change the maximum number of conntracks:

```
vrouter> edit running
vrouter running config#
vrouter running config# system fast-path limits cg-nat max-conntracks 2000000
vrouter running config# commit
```

Modifying capabilities will automatically restart the fast path and interrupt packet processing. To check that the fast path is back up and running, use the following command:

```
vrouter running config# show state system fast-path
fast-path
    enabled stopping
    ..
vrouter running config# show state system fast-path
fast-path
    enabled starting
    ..
vrouter running config# # show state system fast-path
fast-path
    enabled true
    ...
```
3.9. Limitations

Here are the known CG-NAT limitations of the vRouter.

<table>
<thead>
<tr>
<th>Limitation</th>
<th>Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paired address pooling cannot be disabled.</td>
<td>If a user consumes all its ports on a public IP address, a new public IP must be added to the pool and all the sessions must be flushed for the user to start using the new IP. Refer to the Full IP Public section.</td>
</tr>
<tr>
<td>Pools are not shareable.</td>
<td>A pool cannot be shared by two different rules.</td>
</tr>
<tr>
<td>Endpoint mapping/filtering doesn’t support the Address-Dependent mode.</td>
<td>The supported modes are: Independent, Address-and-Port-Dependent.</td>
</tr>
<tr>
<td>No max-sessions-per-user parameter.</td>
<td>There is no option to limit the number of sessions per user. As a result, when the endpoint mapping/filtering mode is set to independent, a user can consume all the available conntracks.</td>
</tr>
<tr>
<td>Capabilities are not checked against available memory.</td>
<td>Configuring too high capabilities can prevent the system from working properly. Refer to the Capability tuning section.</td>
</tr>
</tbody>
</table>