Benchmarking Methodology for Stateful NATxy Gateways using RFC 4814 Pseudorandom Port Numbers

draft-ietf-bmwg-benchmarking-stateful

Gábor LENCSE lencse@sze.hu (Széchenyi István University) – presenter
Keiichi SHIMA keiichi.shima@g.softbank.co.jp (SoftBank)

Summary of the Proposal

• Guides to achieve reproducible stateful NATxy performance measurements producing meaningful results
  – Facilitating to carry out all the measurement procedures of RFC 2544 / RFC 5180 / RFC 8219 like throughput, latency, frame loss rate, etc. to benchmark stateful NATxy (NAT44, NAT64, etc.) gateways
  – Adding new performance metrics specific to stateful testing:
    • Connection setup performance: maximum connection establishment rate
    • Connection tear down performance: connection tear down rate
    • Size of the connection tracking table: connection tracking table capacity
  – Providing guidelines how to use RFC 4814 pseudorandom port numbers with stateful NATxy gateways
Progress of the draft

• Individual draft “04” (presented at IETF 114)
  – Adopted by BMWG as a WG item

• WG draft “00”
  – Added: test setup for stateful NAT64 gateways
  – Consistency checking and corrections

• WG draft “01” (current version)
  – Added: measurements for scalability against
    • the number of connections
    • the number of CPU cores
  – Added: reporting format
Reminder: Test Setup

- Methodology works with any IP versions
  - Now, we use the example of stateful NAT64

```
+--------------------------------------+
| 2001:2::2 | Initiator | Responder | 198.19.0.2 |
+-----------+-----------+-----------+-----------+
| Tester    |           |           |           |
| +---------+-----------+-----------+-----------+
| IPv6 address | [state table] | IPv4 address |
|           | +-----------------+            |
|           | |                       |
|           | +-----------------+            |
|           | |                       |
|           | +-----------------+            |
|           | |                       |
|           | +-----------------+            |
| 2001:2::1 | DUT:      | 198.19.0.1 |
+-----------+-----------+-----------+-----------+
| Stateful NAT64 gateway |            |-----------|
| IPv6 address | [connection tracking table] | IPv4 address |
+-----------------+            +----------------+
```
Reminder: Measurements in two Phases

• Preliminary test phase
  – It serves two purposes:
    • The connection tracking table of the DUT is filled.
    • The state table of the Responder is filled with valid four tuples.
  – It can be used without the real test phase to measure the maximum connection establishment rate.

• Real test phase
  – It MUST be preceded by a preliminary test phase.
  – The “classic” measurement procedures (throughput, frame loss rate, latency, PDV, IPDV) are performed as defined in RFC 8219.
Reminder: To support repeatable measurements

• There are two extreme situations that we can simply ensure
  1. When all test frames create a new connection
     • Ideal for measuring maximum connection establishment rate
  2. When test frames never create a new connection
     • Ideal for the “classic” tests: throughput, latency, frame loss rate, PDV, etc.

• Conditions to achieve them:
  – Large enough and empty connection tracking table for each test
  – Pseudorandom enumeration of all possible port number combinations in the preliminary phase
  – Properly high timeout value in the DUT
Scalability against number of network flows

- Section 10 of RFC 8219 [1] mentions the usage of several network flows, but it does not specify, how to create them.
  - e.g., by using multiple source or destination IP addresses
  - e.g., by using multiple source or destination port numbers

- We recommended to use
  - Single source IP address and destination IP address pair
  - Fixed, larger source port number range (e.g., few times 10,000)
  - Variable size destination port number range, e.g. 10; 100; 1,000; etc.
    - Granularity depends on the purpose.

Scalability against the number of CPU cores

• Stateful NAT64 gateways are often implemented in **software**
  – Examples: Jool, tayga+iptables, OpenBSD PF, FD.io VPP

• Typical view of benchmarking: DUT: **Device Under Test**

• However, software is not bound to a specific hardware!
  – What is not really useful: Performance of X implementation using Y hardware – it does not help when Z hardware is used!
  – What is more useful:
    • Performance of X implementation using a single core of a well-known CPU
    • **Scale up of performance of X implementation with the number of CPU cores**
  – Efficient solution: test with 1, 2, 4, 8, 16, 32, etc. cores
Reporting Format

• Measurements MUST be executed multiple times.
  – The report of the results MUST contain the number of the repetitions

• We RECOMMEND median as the summarizing function plus 1st percentile and the 99th percentile as indices of dispersion
  – Average and standard deviation MAY also be reported.

• All parameters and settings that may influence the performance of the DUT MUST be reported.
  – Some of them may be specific to the given NATxy implementation, e.g.
    • hashsize and nf_conntrack_max for iptables
    • limit of the number of states for OpenBSD PF (set limit states n)
### Reporting Format: Example table

<table>
<thead>
<tr>
<th></th>
<th>0.4M</th>
<th>4M</th>
<th>40M</th>
<th>400M</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>number of sessions (req.)</strong></td>
<td>0.4M</td>
<td>4M</td>
<td>40M</td>
<td>400M</td>
</tr>
<tr>
<td><strong>source port numbers (req.)</strong></td>
<td>40,000</td>
<td>40,000</td>
<td>40,000</td>
<td>40,000</td>
</tr>
<tr>
<td><strong>destination port numbers (req.)</strong></td>
<td>10</td>
<td>100</td>
<td>1,000</td>
<td>10,000</td>
</tr>
<tr>
<td><strong>&quot;hashsize&quot; (i.s.)</strong></td>
<td>2^17</td>
<td>2^20</td>
<td>2^23</td>
<td>2^27</td>
</tr>
<tr>
<td><strong>&quot;nf_conntrack_max&quot; (i.s.)</strong></td>
<td>2^20</td>
<td>2^23</td>
<td>2^26</td>
<td>2^30</td>
</tr>
<tr>
<td><strong>num. sessions / &quot;hashsize&quot; (i.s.)</strong></td>
<td>3.05</td>
<td>3.81</td>
<td>4.77</td>
<td>2.98</td>
</tr>
<tr>
<td><strong>number of experiments (req.)</strong></td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td><strong>error of binary search (req.)</strong></td>
<td>1,000</td>
<td>1,000</td>
<td>1,000</td>
<td>1,000</td>
</tr>
<tr>
<td><strong>connections/s median (req.)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>connections/s 1st perc. (req.)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>connections/s 99th perc. (req.)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3**: Example table: Non-validated maximum connection establishment rate of iptables against the number of sessions
For discussion: multiple network flows

• As for generating **multiple network flows**, we proposed to use
  – a single source IP address destination IP address pair
  – multiple port numbers

• This solution works well with Linux 😊
  – With a proper RSS (Receive-Side Scaling) implementation, it can be set that port numbers are also considered by the hash function to distribute the interrupts of packet arrivals among the CPU cores.

• But it does not work well with OpenBSD 😞
  – Only the IP addresses are considered by the hash function...
  – But there are multiple IP addresses used in the Internet traffic!

Benchmarking Stateful NATxy Gateways
For discussion: multiple network flows

• Shall we add the usage of multiple IP addresses as a requirement?
  – Then measurement results would reflect better the case when a stateful NATxy gateway processes Internet traffic.
For discussion: Any other type of scalability?

• As for scalability, we recommended
  – Scalability against the number of network flows
  – Scalability against the number of CPU cores

• Is there any other type of scalability that would be important to examine?