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

draft-ietf-bmwg-benchmaring-stateful

Gábor LENCSE lencse@sze.hu (Széchenyi István University) – presenter

Keiichi SHIMA keiichi.shima@g.softbank.co.jp (SoftBank)

IETF 115, BMWG, November 10, 2022.

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.

[1] https://datatracker.ietf.org/doc/html/rfc8219#section-10

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

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

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 is 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!

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?