



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)

IETF 117, BMWG, July 26, 2023.

Summary of the Proposal

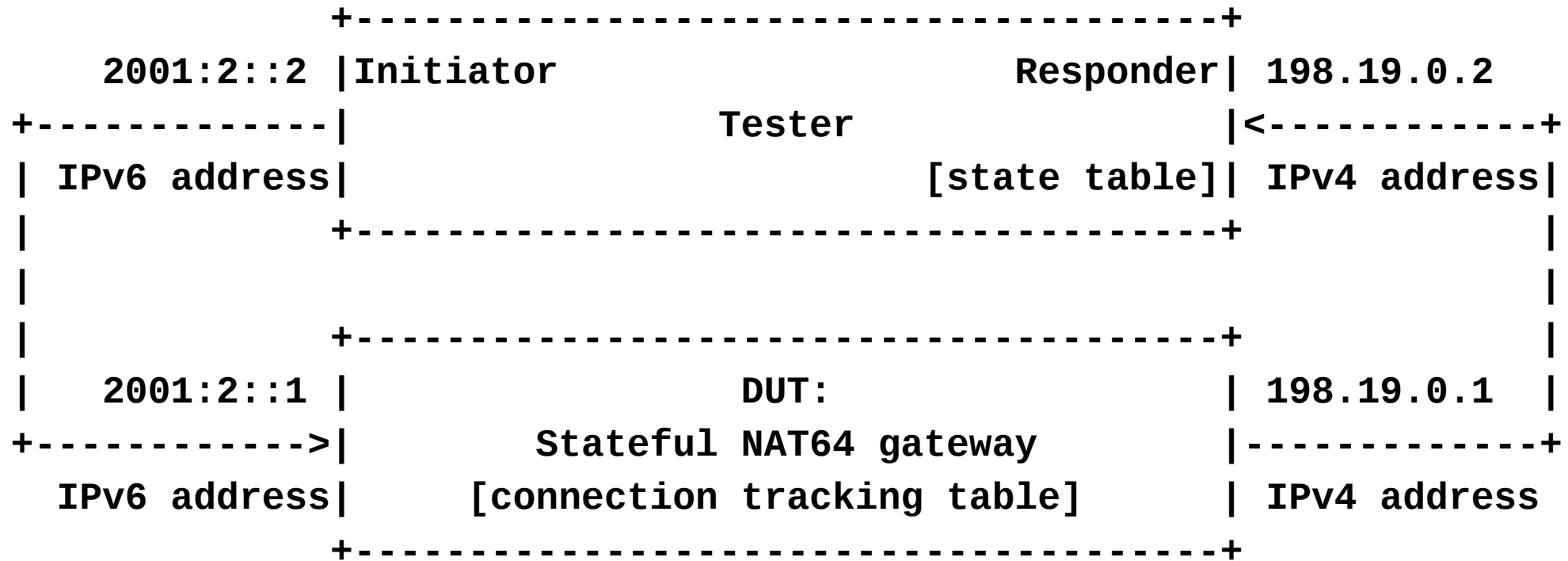
- Guides to achieve reproducible stateful NATxy performance measurements producing meaningful results
 - Facilitating to carry out all the measurement procedures of RFC 2544 / RFC 5180 / RFC 8219 like *throughput, latency, frame loss rate*, etc. to benchmark stateful NATxy (NAT44, NAT64, etc.) gateways
 - 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

- ...
- WG draft “02” (Presented at IETF 116)
 - Added: the usage of multiple IP addresses
 - Section 4.5.1 was removed and split into two Sections: Section 5 about scalability measurements and Section 6 about reporting format.
- WG draft “03” (current version)
 - Updated the usage of multiple IP addresses to have enough of them
 - Test phases were renamed as follows:
 - preliminary test phase --> test phase 1
 - real test phase --> test phase 2.

Reminder: Test Setup

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



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

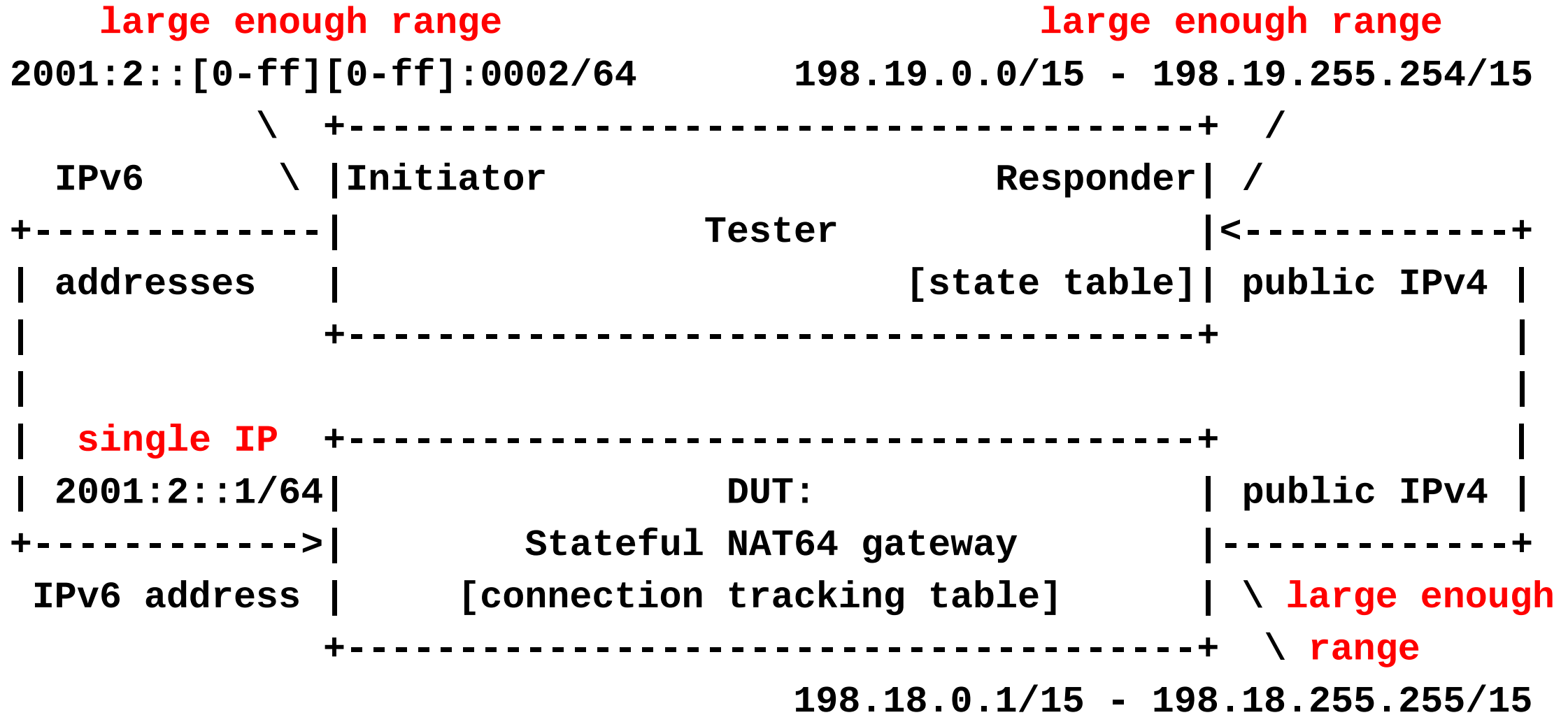
Reminder: Motivation for using multiple IP addresses

- 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!

How to generate multiple IPv4 addresses?

- *When router testing is done*, section 12 of RFC2544 requires testing first using a single source and destination IP address pair, and then using destination IP addresses from 256 different networks.
 - The 16-23 bits of the 198.18.0.0/24 and 198.19.0.0/24 addresses can be used to express the 256 networks. (198.19.{0..255}.0/24)
 - *As we do not do router testing*, we do not need to use multiple destination networks, therefore, these bits are available for expressing multiple IP addresses that belong to the same "/16" network.
 - The two /16 ranges together make a /15 range.
 - And they all can be used on the right side of the test setup! □

Stateful NAT64 Test Setup w/ Multiple IP Addresses



Proof of concept

- The usage of multiple IP addresses has just been implemented in **siitperf**
- Source code is available from GitHub (under GPLv3 license):
<https://github.com/lencsegabor/siitperf>
- Benchmarking tests are being performed...
- Will be documented in a journal paper (under writing)

Question

- Is this draft matured for WGLC?