Scalability of IPv6 Transition Technologies for IPv4aaS

Abstract

Several IPv6 transition technologies have been developed to provide
customers with IPv4-as-a-Service (IPv4aaS) for ISPs with an IPv6-
only access and/or core network. All these technologies have their
advantages and disadvantages, and depending on existing topology,
skills, strategy and other preferences, one of these technologies
may be the most appropriate solution for a network operator.

This document examines the scalability of the five most prominent
IPv4aaS technologies (464XLAT, Dual Stack Lite, Lightweight 4over6,
MAP-E, MAP-T) considering two aspects: (1) how their performance
scales up with the number of CPU cores, (2) how their performance
degrades, when the number of concurrent sessions is increased until
hardware limit is reached.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

IETF has standardized several IPv6 transition technologies [LEN2019]
and occupied a neutral position trusting the selection of the most
appropriate ones to the market. [I-D.ietf-v6ops-transition-
comparison] provides a comprehensive comparative analysis of the
five most prominent IPv4aaS technologies to assist operators with
this problem. This document adds one more detail: measurement data
regarding the scalability of the examined IPv4aaS technologies.

Currently, this document contains only the scalability measurements
of the iptables stateful NAT44 implementation. It serves as a sample
to test if the disclosed results are (1) useful and (2) sufficient
for the network operators.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
"OPTIONAL" in this document are to be interpreted as described in
BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all
capitals, as shown here.

2. Scalability of iptables

2.1. Measurement Method

[RFC8219] has defined a benchmarking methodology for IPv6 transition
technologies. [I-D.lencse-bmwg-benchmarking-stateful] has amended it
by addressing how to benchmark stateful NATxy gateways using
pseudorandom port numbers recommended by [RFC4814]. It has defined a
measurement procedure for maximum connection establishment rate and
reused the classic measurement procedures like throughput, latency,
frame loss rate, etc. from [RFC8219]. We used two of them: maximum
connection establishment rate and throughput to characterize the
performance of the examined system.

The scalability of iptables is examined in two aspects:

   *How its performance scales up with the number of CPU cores?

   *How its performance degrades, when the number of concurrent
    sessions is increased?

```
                +-------------------------------------+
     10.0.0.2  |Initiator                   Responder| 198.19.0.2
 +------------|              Tester                  |<------------+
 | private IPv4|                         [state table]| public IPv4 |
 |            +-------------------------------------+             |
 |                                                               |
 |            +-------------------------------------+             |
 |    10.0.0.1 |                DUT:                  | 198.19.0.1  |
 +----------->|         Sateful NAT44 gateway        |------------+
   private IPv4|      [connection tracking table]      | public IPv4
                +-------------------------------------+
```

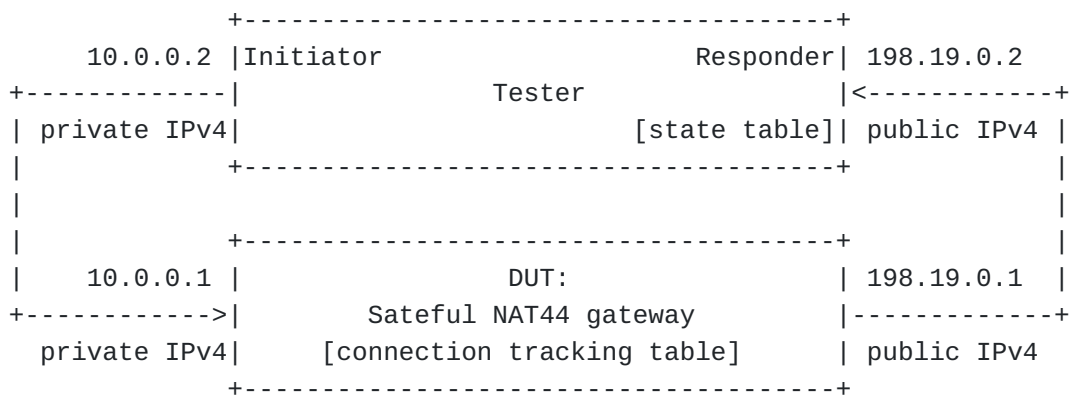Figure 1: Test setup for benchmarking stateful NAT44 gateways

The test setup in Figure 1 was followed. The two devices, the Tester
and the DUT (Device Under Test), were both Dell PowerEdge R430
servers having two 2.1GHz Intel Xeon E5-2683 v4 CPUs, 384GB 2400MHz
DDR4 RAM and Intel 10G dual port X540 network adapters. The NICs of
the servers were interconnected by direct cables, and the CPU clock

frequecy was set to fixed 2.1 GHz on both servers. They had Debian 9.13 Linux operating system with 4.9.0-16-amd64 kernel. The measurements were performed by siitperf [LEN2021] using the "stateful" branch (latest commit Aug. 16, 2021). The DPDK version was 16.11.11-1+deb9u2. The version of iptables was 1.6.0.

The ratio of number of connections in the connection tracking table and the value of the hashsize parameter of iptables significantly influences its performance. Although the default setting is hashsize=nf_conntrack_max/8, we have usually set hashsize=nf_conntrack_max to increase the performance of iptables, which was crucial, when high number of connections were used, because then the execution time of the tests was dominated by the preliminary phase, when several hundereds of millions connections had to be established. (In some cases, we had to use different settings due to memory limitations. The tables presenting the results always contain these parameters.)

The size of the port number pool is an important parameter of the bechmarking method for stateful NATxy gateways, thus it is also given for all tests.

2.2. Performance scale up against the number of CPU cores

To examine how the performance of iptables scales up with the number of CPU cores, the number of active CPU cores was set to 1, 2, 4, 8, 16 using the "maxcpus=" kernel parameter.

The number of connections was always 4,000,000 using 4,000 different source port numbers and 1,000 different destination port numbers. Both the connection tracking table size and the hash table size was set to 2^23.

The error of the binary search was chosen to be lower than 0.1% of the expected results. The experiments were executed 10 times.

Besides the connection establishment rate and the throughput of iptables, also the throughput of the IPv4 packet forwarding of the Linux kernel was measured to provide a basis for comparison.

The results are presented in Figure 2. The unit for the maximum connection establishment rate is 1,000 connections per second. The unit for throughput is 1,000 packets per second (measured with bidirectional traffic, and the number of all packets per second is displayed).

```
num. CPU cores            1          2          4          8         16
src ports            4,000      4,000      4,000      4,000      4,000
dst ports            1,000      1,000      1,000      1,000      1,000
num. conn.       4,000,000  4,000,000  4,000,000  4,000,000  4,000,000
conntrack t. s.      2^23       2^23       2^23       2^23       2^23
hash table size      2^23       2^23       2^23       2^23       2^23
c.t.s/num.conn.     2.097      2.097      2.097      2.097      2.097
num. experiments       10         10         10         10         10
error                 100        100        100      1,000      1,000
cps median          223.5      371.1      708.7      1,341      2,383
cps min             221.6      367.7      701.7      1,325      2,304
cps max             226.7      375.9      723.6      1,376      2,417
cps rel. scale up       1      0.830      0.793      0.750      0.666
throughput median   414.9      742.3      1,379      2,336      4,557
throughput min      413.9      740.6      1,373      2,311      4,436
throughput max      416.1      746.9      1,395      2,361      4,627
tp. rel. scale up       1      0.895      0.831      0.704      0.686
IPv4 packet forwarding (using the same port number ranges)
error                 200        500      1,000      1,000      1,000
throughput median   910.9      1,523      3,016      5,920     11,561
throughput min      874.8      1,485      2,951      5,811     10,998
throughput max      914.3      1,534      3,037      5,940     11,627
tp. rel. scale up       1      0.836      0.828      0.812      0.793
throughput ratio (%) 45.5       48.8       45.7       39.5       39.4
```

Figure 2: Scale up of iptables against the number of CPU cores

Whereas the throughput of IPv4 packet forwarding scaled up from
0.91Mpps to 11.56Mpps showing a relative scale up of 0.793, the
throughput of iptables scaled up from 414.9kpps to 4,557kpps showing
a relative scale up of 0.686 (and the relative scale up of the
maximum connection establishment rate is only 0.666). On the one
hand, this is the price of the stateful operation. On the other
hand, this result is quite good compared to the scale-up results of
NSD (a high performance authoritative DNS server) presented in Table
9 of [LEN2020], which is only 0.52. (1,454,661/177,432=8.2-fold
performance using 16 cores.) And DNS is not a stateful technology.

2.3. Performance degradation caused by the number of sessions

To examine how the performance of iptables degrades with the number
connections in the connection tracking table, the number of
connections was increased fourfold by doubling the size of both the
source port number range and the destination port number range. Both
the connection tracking table size and the hash table size was also
increased four fold. However, we reached the limits of the hardware
at 400,000,000 connections: we could not set the size of the hash
table to 2^29 but only to 2^28. The same value was used at
800,000,000 connections too, when the number of connections was only

doubled, because 1.6 billion connections would not fit into the
memory.

The error of the binary search was chosen to be lower than 0.1% of
the expected results. The experiments were executed 10 times (except
for the very long lasting measurements with 800,000,000
connections).

The results are presented in Figure 3. The unit for the maximum
connection establishment rate is 1,000,000 connections per second.
The unit for throughput is 1,000,000 packets per second (measured
with bidirectional traffic, and the number of all packets per second
is displayed).

num. conn.	1.56M	6.25M	25M	100M	400M	800M
src ports	2,500	5,000	10,000	20,000	40,000	40,000
dst ports	625	1,250	2,500	5,000	10,000	20,000
conntrack t. s.	2^{21}	2^{23}	2^{25}	2^{27}	2^{29}	2^{30}
hash table size	2^{21}	2^{23}	2^{25}	2^{27}	2^{28}	2^{28}
num. exp.	10	10	10	10	10	5
error	1,000	1,000	1,000	1,000	1,000	1,000
n.c./h.t.s.	0.745	0.745	0.745	0.745	1.490	2.980
cps median	2.406	2.279	2.278	2.237	2.013	1.405
cps min	2.358	2.226	2.226	2.124	1.983	1.390
cps max	2.505	2.315	2.317	2.290	2.050	1.440
throughput med.	5.326	4.369	4.510	4.516	4.244	3.689
throughput min	5.217	4.240	3.994	4.373	4.217	3.670
throughput max	5.533	4.408	4.572	4.537	4.342	3.709

Figure 3: Performance of iptables against the number of sessions

The performance of iptables shows degradation at 6.25M connections
compared to 1.56M connections very likely due to the exhaustion of
the L3 cache of the CPU of the DUT. Then the performance of iptables
is fearly constant up to 100M connections. A small performance
decrease can be observed at 400M connections due to the lower hash
table size. A more significant performance decrease can be observed
at 800M connections. It is caused by two factors:

  *on average, about 3 connections were hashed to the same place

  *non NUMA local memory was also used.

We note that the CPU has 2 NUMA nodes, cores 0, 2, ... 14 belong to
NUMA node 0, and cores 1, 3, ... 15 belong to NUMA node 1. The
maximum memory consumption with 400,000,000 connections was below
150GB, thus it could be stored in NUMA local memory.

Therefore, we have pointed out important limitations of the stateful NAT44 technology:

  *there is a performance decrease, when approaching hardware limits

  *there is a hardware limit, beyond which the system cannot handle
   the connections at all (e.g. 1600M connections would not fit into
   the memory).

Therefore, we can conclude that, on the one hand, a well tailored hashing may guarantee an excellent scale-up of stateful NAT44 regarding the number of connections in a wide range, however, on the other hand, stateful operation has its limits resulting both in performance decrease, when approaching hardware limits and also in inability to handle more sessions, when reaching the memory limits.

3. Scalability of Jool

3.1. Measurement Method

The same methodology was used as in Section 2, but now the test setup in Figure 4 was followed. The same Tester and DUT devices were used as before, but the operating system of the DUT was updated to Debian 10.11 with 4.19.0-18-amd64 kernel to meet the requirement of the jool-tools package. The version of Jool was 4.1.6. (The most mature version of Jool at the date of starting the measurements, Relase Date: 2021-12-10.)

```
                 +--------------------------------------+
    2001:2::2 |Initiator                     Responder| 198.19.0.2
 +------------|               Tester                  |<-----------+
 | IPv6 address|                       [state table]| IPv4 address|
 |             +--------------------------------------+             |
 |                                                                  |
 |             +--------------------------------------+             |
 |    2001:2::1 |                 DUT:                 | 198.19.0.1  |
 +------------>|         Sateful NAT64 gateway        |-----------+
   IPv6 address|      [connection tracking table]     | IPv4 address
               +--------------------------------------+
```
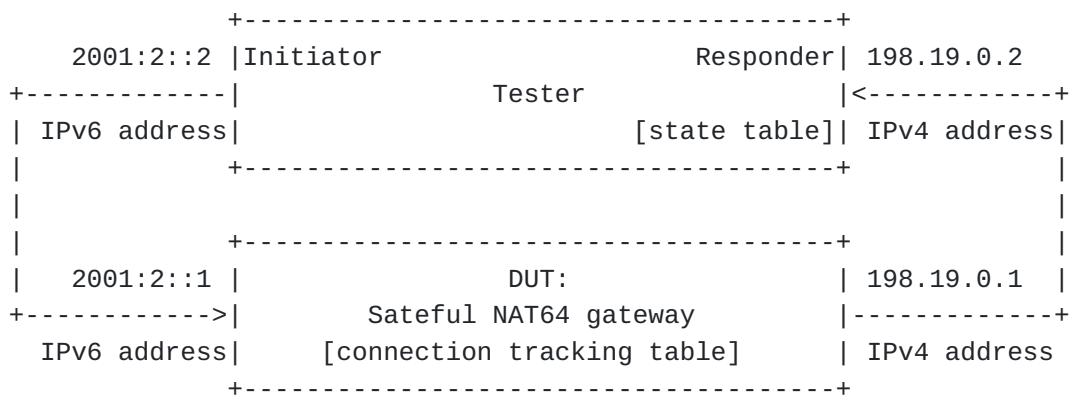
    Figure 4: Test setup for benchmarking stateful NAT64 gateways

Unlike with iptables, we did not find any way to tune the hashsize or any other parameters of Jool.

3.2. Performance scale up against the number of CPU cores

The number of connections was always 1,000,000 using 2,000 different source port numbers and 500 different destination port numbers.

The error of the binary search was chosen to be lower than 0.1% of the expected results. The experiments were executed 10 times.

The results are presented in Figure 5. The unit for the maximum connection establishment rate is 1,000 connections per second. The unit for throughput is 1,000 packets per second (measured with bidirectional traffic, and the number of all packets per second is displayed).

num. CPU cores	1	2	4	8	16
src ports	2,000	2,000	2,000	2,000	2,000
dst ports	500	500	500	500	500
num. conn.	1,000,000	1,000,000	1,000,000	1,000,000	1,000,000
num. experiments	10	10	10	10	10
error	100	100	100	100	100
cps median	228.6	358.5	537.4	569.9	602.6
cps min	226.5	352.5	530.7	562.0	593.7
cps max	230.5	362.4	543	578.3	609.7
cps rel. scale up	1	0.784	0.588	0.312	0.165
throughput median	251.8	405.7	582.4	604.1	612.3
throughput min	249.8	402.9	573.2	587.3	599.8
throughput max	253.3	409.6	585.7	607.2	616.6
tp. rel. scale up	1	0.806	0.578	0.300	0.152

Figure 5: Scale up of Jool against the number of CPU cores

Both the maximum connection establishment rate and the throughput scaled up poorly with the number of active CPU cores. The increase of the performance was very low above 4 CPU cores.

3.3. Performance degradation caused by the number of sessions

To examine how the performance of Jool degrades with the number connections, the number of connections was increased fourfold by doubling the size of both the source port number range and the destination port number range. We did not reach the limits of the hardware regarding the number of connections, because unlike iptables, Jool worked also with 1.6 billion connections.

The error of the binary search was chosen to be lower than 0.1% of the expected results and the experiments were executed 10 times (except for the very long lasting measurements with 800 million and 1.6 billion connections to save execution time).

The results are presented in [Figure 6](#). The unit for the maximum connection establishment rate is 1,000 connections per second. The unit for throughput is 1,000 packets per second (measured with bidirectional traffic, and the number of all packets per second is displayed).

num. conn.	1.56M	6.35M	25M	100M	400M	1600M
src ports	2,500	5,000	10,000	20,000	40,000	40,000
dst ports	625	1,250	2,500	5,000	10,000	40,000
num. exp.	10	10	10	10	5	5
error	100	100	100	100	1,000	1,000
cps median	480.2	394.8	328.6	273.0	243.0	232.0
cps min	468.6	392.7	324.9	269.4	243.0	230.5
cps max	484.9	397.4	331.3	280.6	244.5	233.6
throughput med.	511.5	423.9	350.0	286.5	257.8	198.4
throughput min	509.2	420.3	348.2	284.2	257.8	195.3
throughput max	513.1	428.3	352.5	290.8	260.9	201.6

Figure 6: Performance of Jool against the number of sessions

The performance of Jool shows degradation at the entire range of the number of connections. We did not analyze the root cause of the degradation yet. And we are not aware of the implementation of its connection tracking table. We also plan to check the memory consumption of Jool, what is definitely lower that that of iptables.

4. Acknowledgements

5. IANA Considerations

This document does not make any request to IANA.

6. Security Considerations

TBD.

7. References

7.1. Normative References

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/
            RFC2119, March 1997, <https://www.rfc-editor.org/info/
            rfc2119>.

[RFC4814]   Newman, D. and T. Player, "Hash and Stuffing: Overlooked
            Factors in Network Device Benchmarking", RFC 4814, DOI
            10.17487/RFC4814, March 2007, <https://www.rfc-
            editor.org/info/rfc4814>.

[RFC8174]   Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
            2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
            May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8219]   Georgescu, M., Pislaru, L., and G. Lencse, "Benchmarking
            Methodology for IPv6 Transition Technologies", RFC 8219,
            DOI 10.17487/RFC8219, August 2017, <https://www.rfc-
            editor.org/info/rfc8219>.

7.2. Informative References

[I-D.ietf-v6ops-transition-comparison]
            Lencse, G., Martinez, J. P., Howard, L., Patterson, R.,
            and I. Farrer, "Pros and Cons of IPv6 Transition
            Technologies for IPv4aaS", Work in Progress, Internet-
            Draft, draft-ietf-v6ops-transition-comparison-01, 16
            October 2021, <https://www.ietf.org/archive/id/draft-
            ietf-v6ops-transition-comparison-01.txt>.

[I-D.lencse-bmwg-benchmarking-stateful] Lencse, G. and K. Shima,
            "Benchmarking Methodology for Stateful NATxy Gateways
            using RFC 4814 Pseudorandom Port Numbers", Work in
            Progress, Internet-Draft, draft-lencse-bmwg-benchmarking-
            stateful-02, 10 October 2021, <https://www.ietf.org/
            archive/id/draft-lencse-bmwg-benchmarking-
            stateful-02.txt>.

[LEN2019]   Lencse, G. and Y. Kadobayashi, "Comprehensive Survey of
            IPv6 Transition Technologies: A Subjective Classification
            for Security Analysis", IEICE Transactions on
            Communications, vol. E102-B, no.10, pp. 2021-2035., DOI:
            10.1587/transcom.2018EBR0002, 1 October 2019, <http://
            www.hit.bme.hu/~lencse/publications/e102-b_10_2021.pdf>.

[LEN2020]   Lencse, G., "Benchmarking Authoritative DNS Servers",
            IEEE Access, vol. 8. pp. 130224-130238, DOI: 10.1109/

ACCESS.2020.3009141, July 2020, <https://ieeexplore.ieee.org/document/9139929>.

[LEN2021] Lencse, G., "Design and Implementation of a Software Tester for Benchmarking Stateless NAT64 Gateways", IEICE Transactions on Communications, DOI: 10.1587/transcom.2019EBN0010, 1 February 2021, <http://www.hit.bme.hu/~lencse/publications/IEICE-2020-siitperf-revised.pdf>.

Appendix A. Change Log

A.1. 00

Initial version: scale up of iptables.

A.2. 01

Added the scale up of Jool.

Author's Address

Gabor Lencse
Szechenyi Istvan University
Gyor
Egyetem ter 1.
H-9026
Hungary

Email: lencse@sze.hu