Workgroup: v6ops
Internet-Draft:
draft-lencse-v6ops-transition-scalability-05
Published: 14 October 2023
Intended Status: Informational
Expires: 16 April 2024
Authors: G. Lencse
Széchenyi István University

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 16 April 2024.

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. [RFC9313] 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.

This draft is a collection of various measurement results. Some measurements with the iptables stateful NAT44 implementation and the Jool stateful NAT64 implementation were performed directly for this draft. Some other results published in open access research papers are added gradually.

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. Introduction to iptables

Netfilter [NETFLTR] is a widely used firewall, NAT and packet mangling framework for Linux. It is often called as "iptables" after the name of its user space command line tool. From our point of view, iptables is used as a stateful NAT44 solution. (Also called as NAPT: Network Address and Port Translation.) It is a free and open source software under the GPLv2 license.

This document deals with iptables for multiple considerations:

*To provide a reference for the scalability of various stateful NAT64 implementations. (We use it to prove that a stateful NATxy solution does not need to exhibit a poor scalability.)

*To provide IPv6 operators with a basis for comparison if is it worth using an IPv4aaS solution over Carrier-grade NAT.

*To prove the scalability of iptables, when iptables is used as a part of the CE of MAP-T (see later).

2.2. Measurement Method

[RFC8219] has defined a benchmarking methodology for IPv6 transition technologies. [I-D.ietf-bmwg-benchmarking-stateful] has amended it by addressing how to benchmark stateful NATxy gateways using pseudorandom port numbers recommended by [RFC4814]. It has defined measurement procedures for maximum connection establishment rate, connection tear down rate and connection table capacity measurement, plus it reused the classic measurement procedures like throughput, latency, frame loss rate, etc. from [RFC8219]. Besides the new metrics, we used 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?

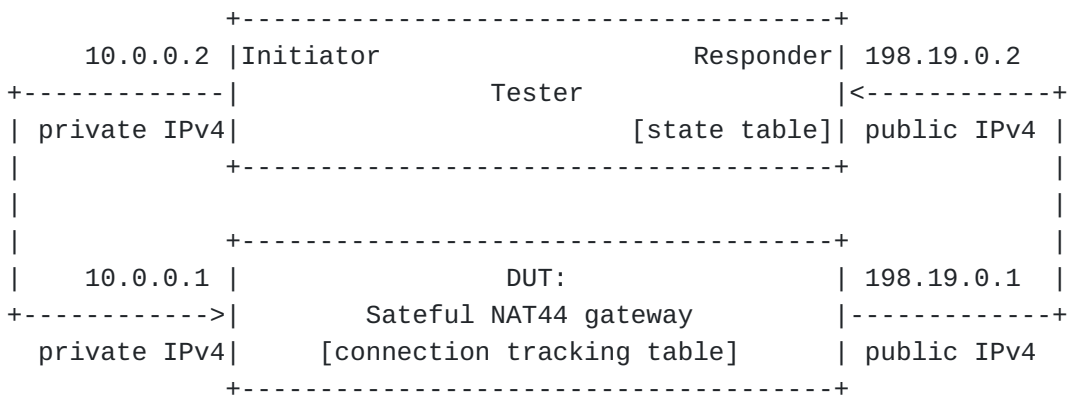

Figure 1: Test setup for benchmarking stateful NAT44 gateways

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.

2.3. 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 (Please refer to the next figure for the explanation of the abbreviations.)

abbreviation	explanation
num. CPU cores	number of CPU cores
src ports	size of the source port number range
dst ports	size of the destination port number range
num. conn.	number of connections = src ports * dst ports
conntrack t. s.	size of the connection tracking table of the DUT
hash table size	size of the hash table of the DUT
c.t.s/num.conn.	conntrack table size / number of connections
num. experiments	number of experiments
error	the difference between the upper and the lower bound of the binary search when it stops
cps (median/min/max)	maximum connection establishment rate (median, minimum, maximum)
cps rel. scale up	the relative scale up of the maximum connection establishment rate against the number of CPU cores
tp. rel. scale up	the relative scale up of the throughput
throughput ratio (%)	the ratio of the throughput of iptables and the throughput of IPv4 packet forwarding

Figure 3: Explanation of the abbreviations for the 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.

2.4. 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 4. 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 4: 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.

2.5. Connection tear down rate

[I-D.ietf-bmwg-benchmarking-stateful] has defined connection tear down rate measurement as an aggregate measurement, that is, N number of connections are loaded into the connection tracking table of the DUT and then the entire content of the connection tracking table is deleted, and its deletion time is measured (T). Finally, the connection tear down rate is computed as: N/T.)

We have observed that the deletion of an empty connection tracking table of iptables my take a significant amount of time depending on its size. Therefore, we made our measurements more accurate by subtracting the deletion time of the empty connection tracking table from that of the filled one, thus we got the time spent with the deleting of the connections.

The same setup and parameters were used as in Section 2.4 and the experiments were executed 10 times (except for the long lasting measurements with 800,000,000 connections).

The results are presented in Figure 5.

num. conn.	1.56M	6.35M	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
n.c./h.t.s.	0.745	0.745	0.745	0.745	1.490	2.980
full contr. del med	4.33	18.05	74.47	305.33	1,178.3	2,263.1
full contr. del min	4.25	17.93	72.04	299.06	1,164.0	2,259.6
full contr. del max	4.38	18.20	75.13	310.05	1,188.3	2,275.2
empty contr. del med	0.55	1.28	4.17	15.74	31.2	31.2
empty contr. del min	0.55	1.26	4.16	15.73	31.1	31.1
empty contr. del max	0.57	1.29	4.22	15.79	31.2	31.2
conn. deletion time	3.78	16.77	70.30	289.59	1,147.2	2,232.0
conn. tear d. rate	413,360	372,689	355,619	345,316	348,690	358,429

Figure 5: Connection tear down rate of iptables against the number of connections

The connection tear down performance of iptables shows significant 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 it shows only a minor degradation up to 100M connections. A small performance increase can be observed at 400M connections due to the relatively lower hash table size. A more visible performance decrease can be observed at 800M connections. It is likely caused by keeping the hash table size constant and doubling the number of connections. The same thing that caused performance degradation of the maximum connection establishment rate and throughput, made now the deletion of the connections faster and thus caused an increase of the connection tear down rate.

We note that according to the recommended settings of iptables, 8 connections are hashed to each place of the hash table on average, but we wilfully used much smaller number (0.745 whenever it was possible) to increase the maximum connection estabilishment rate and thus to speed up experimenting. However, finally this choice significantly slowed down our experiments due to the very low connection tear down rate.

2.6. Connection tracking table capacity

[I-D.ietf-bmwg-benchmarking-stateful] has defined connection tracking table capacity measurement using the following quantities:

*C0: initial safe value for the size of the connection tracking table (the connection tracking table can surely store C0 entries)

*R0: safe connection establisment rate for C0 connections (measured initially)

*CS: safe value for the size of the connection tracking table during the current measurement (taken from the previous iteration step)

*RS: safe connection establisment rate for CS connections during the current measurement (measured during the previous iteration)

*CT: the currently tested size of the connection tracking table during the exponential search; also used in the final binary search.

*RT: the currently used connection establisment rate for testing with CT number of connections during the exponential search

*alpha: safety factor to prevent that connection validation fails due to sending the validation frames at a too high rate

*beta: factor to express a too high drop of the connection establishment rate during the exponential search

*gamma: factor to express a too high drop of the connection establishment rate during the final binary search

First, the order of magnitude of the size of the connection tracking table is determined by an exponential search. When it stops, then the C capacity of the connection tracking table is between CS and CT=2*CS.

Then the C size of the connection tracking table is determined by a binary search within E error.

Measurements were performed with the following parameters: hashsize=nf_conntrack_max=2**22=4,194,304; R0=1,000,000; E=1, alpha=1.0; beta=0.2; gamma=0.4. The measurements were performed 10 times to see the stability of the results.

	C0	R0	CS	RS	CT	C
median	1,000,000	2,562,500	4,000,000	2,250,945	8,000,000	4,194,301
min	1,000,000	2,437,500	4,000,000	2,139,953	8,000,000	4,194,300
max	1,000,000	2,687,500	4,000,000	2,327,269	8,000,000	4,194,302

Figure 6: Connection tracking table capacity measurement resultss for iptables (actual size: 4,194,304)

The results are presented in Figure 6. The exponential search finished at its third step (CS=4,000,000 and CT=8,000,000). And the result of the final binary search was always very close to 4,194,304.

3. Scalability of Jool

3.1. Introduction to Jool

Jool [JOOLMX] is an open source SIIT and stateful NAT64 implementation for Linux. Since its version 4.2 it also supports MAP-T. It has been developed by NIC Mexico in cooperation with ITESM (Monterrey Institute of Technology and Higher Education). Its source code is released under GPLv2 license.

3.2. Measurement Method

The same methodology was used as in Section 2, but now the test setup in Figure 7 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.)

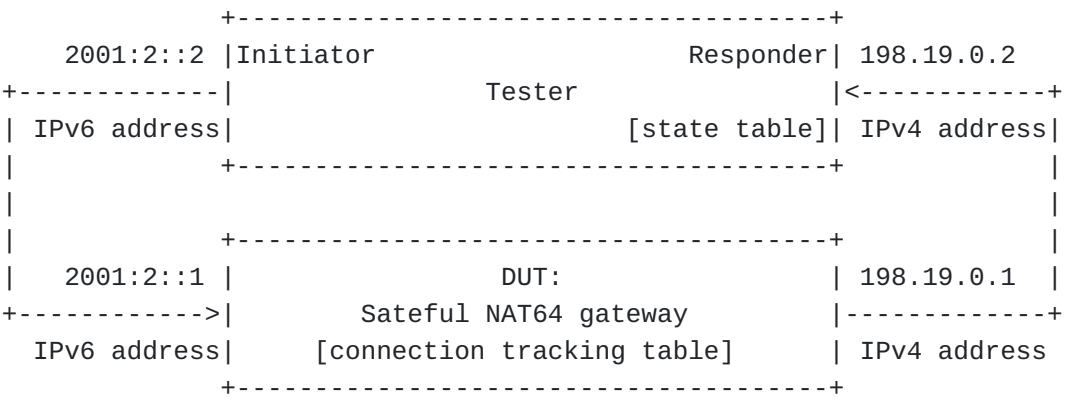

Figure 7: Test setup for benchmarking stateful NAT64 gateways

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

3.3. 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 8. 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 8: 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.

3.4. 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 9. 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 9: 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.

3.5. Connection tear down rate

Basically, the same measurement method was used as in Section 2.5, however having no parameter of Jool to tune, only a single measurement series was performed to determine the deletion time of the empty connection tracking table. The median, minimum and maximum values of the 10 measurements were 0.46s, 0.42s and 0.50s respectively.

The same setup and parameters were used as in Section 3.4 and the experiments were executed 10 times (except for the long lasting measurements with 800,000,000 connections).

The results are presented in Figure 10. The unit for the connection tear down rate is 1,000,000 connections per second.

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	10	5
full contr. del med	0.87	2.05	7.84	36.38	126.09	474.68
full contr. del min	0.80	2.02	7.80	36.27	125.84	473.20
full contr. del max	0.91	2.09	7.94	36.80	127.54	481.38
empty contr. del med	0.46	0.46	0.46	0.46	0.46	0.46
conn. deletion time	0.41	1.59	7.38	35.92	125.63	474.22
conn. t. d. r. (M)	3.811	3.931	3.388	2.784	3.184	3.374

Figure 10: Connection tear down rate of Jool against the number of connections

The connection tear down performance of Jool is excellent at any number of connections. It is about and order of magnitude higher that its connection establishment rate and than the connection tear down rate of iptables. (A slight degradation can be observed at 100M connections.)

3.6. Validation of connection establishment

The measurement of connection establishment rate with validation was performed using different values for the "alpha" parameter.

The results are presented in Figure 11. It is well visible that alpha values 0.8 and 0.6 cause significant decrease of the validated rate, therefore, they are unsuitable. Values 0.5 and 0.25 make no difference compared to the unvalidated connection establishment rate. (The less than 1,000 cps increase of the median is deliberately a measurement error.)

alpha	0.8	0.6	0.5	0.25	no validation
num. conn.	4,000,000	4,000,000	4,000,000	4,000,000	4,000,000
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. exp.	10	10	10	10	10
error	100	100	100	100	100
cps median	323,534	429,491	479,296	479,199	478,417
cps min	322,948	426,464	473,339	474,120	474,902
cps max	325,097	431,542	483,690	483,299	484,667

Figure 11: Connection establishment rate rate of Jool against the alpha parameter

4. Scalability of OpenBSD PF

4.1. Introduction to OpenBSD PF

PF [PFBOOK] is the packet filtering solution of OpenBSD. PF supports statetul NAT64 since May 1, 2012, the release of OpenBSD 5.1. Its connection tracking system uses red-black tree, which is a kind of balanced tree that can ensure O(log(n)) search time, where n is the number of elements in the tree.

It is noted that the main focus of the OpenBSD project is security and not performance.

4.2. Measurement Method

The same methodology was used as in Section 3, and the same test setup shown in Figure 7 was followed. The same Tester and DUT devices were used as before, but the operating system of the DUT was

OpenBSD 7.1 with GENERIC.MP#465 amd64 kernel. All the details of the measurements can be found in [LEN2023]. It is also explained in this open access paper, why there was no point in measuring the scalability of OpenBSD PF against the number of CPU cores.

Similarly to Jool, no parameters of OpenBSD PF were tuned.

4.3. Performance degradation caused by the number of sessions

To examine how the performance of OpenBSD PF degrades with the number connections, the number of connections was increased tenfold by keeping the size of source port number range a fixed value (40,000) and increasing the size of the destination port number range tenfold. This time we only aimed to make a rough assessment of the scalability of the performance of OpenBSD PF againts the number of connections, and we did not aim to reach the limits of the hardware regarding the number of 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.

The results are presented in Figure 12. The unit for the maximum connection establishment rate is 1 connection per second. The unit for throughput is 1 frame per second (measured with bidirectional traffic, and the number of all packets per second is displayed).

The results show a moderate performance degradation, to a similar extent as Jool. However the results themselves are significantly lower than that of Jool.

Number of connections	400,000	4,000,000	40,000,000
Source port numbers	40,000	40,000	40,000
Destination port numbers	10	100	1,000
Error (cps)	50	40	50
Median (cps)	120,214	85,039	74,022
Minimum (cps)	118,701	84,882	73,680
Maximum (cps)	122,411	85,351	74,266
Median / previous median	-	0.71	0.87
Error (fps)	200	80	100
Median (fps)	237,304	198,828	173,338
Minimum (fps)	236,912	198,046	172,946
Maximum (fps)	250,584	199,452	174,120
Median / previous median	-	0.84	0.87

Figure 12: Performance of OpenBSD PF against the number of sessions

4.4. Connection tear down rate

Basically, the same measurement method was used as in Section 3.5.

The same setup and parameters were used as in Section 4.3 and the experiments were executed 10 times.

The results are presented in Figure 13. The unit for the connection tear down rate is 1 connection per second.

Number of connections	400,000	4,000,000	40,000,000
Source port numbers	40,000	40,000	40,000
Destination port numbers	10	100	1,000
Filled table deletion time med. (s)	1.45	11.56	94.20
Filled table deletion time min. (s)	1.36	11.03	91.73
Filled table deletion time max. (s)	1.78	13.81	118.52
Empty table deletion time med. (s)	0.37	0.37	0.37
Empty table deletion time min. (s)	0.36	0.36	0.36
Empty table deletion time max. (s)	0.37	0.37	0.37
Connections deletion time (s)	1.08	11.19	93.83
Connection tear down rate (cps)	370,370	357,622	426,303

Figure 13: Connection tear down rate of OpenBSD PF against the number of connections

The connection tear down performance of OpenBSD PF is moderate at any number of connections. And the results are about an order of magnitude lower than that of Jool.

5. Scalability Comparison of the Jool Implementation of the 464XLAT and of the MAP-T IPv4aaS Technologies using DNS Traffic

This section summarizes the essence of our meeasurements for the scalability comparison of the Jool implementation of the 464XLAT and of the MAP-T IPv4aaS technologies presented in [LEN2022]. The measurements did not comply with the requirements of [RFC8219], but the results give an insight into the scalability of the Jool implementation of the two technologies. Because of the limitations of the measurement method, only their scalability with the number of CPU cores was examined.

5.1. 464XLAT Scalability Measurements and Results

The measurement setup for the scalability analysis of 464XLAT is shown in Figure 14.

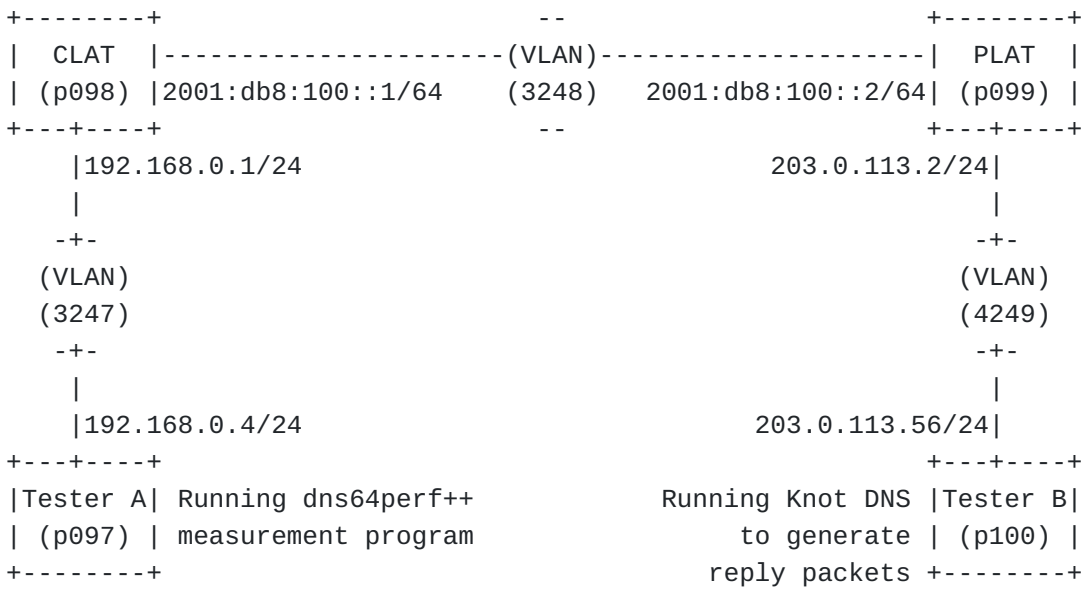

Figure 14: Test setup for benchmarking 464XLAT using DNS traffic

The p097 - p100 devices were the same type of Dell PowerEdge R430 servers residing at NICT StarBED as before, and Debian Linux 10.11 operating system with kernel version 4.19 was used. Both CLAT and PLAT was implemented by Jool [JOOLMX]. To faciliate a fair comparison with MAP-T, Jool version 4.2.0-rc2 was used as Jool supports MAP-T from its 4.2 version.

The measurement traffic was generated by the dns64perf++ program, which sent DNS qeries for "AAAA" records and counted the valid replies. The reverse traffic was generated by the "Knot DNS" authoritative DNS server. As not direct cable connections, but a switch with VLANS was used, we allowed 0.01% packet loss during the binary search to find the highest supported rate. To measure how the performance of the 464XLAT test system scaled up with the number of CPU cores, the number of CPU cores of the CLAT and PLAT devices were set to: 1, 2, 4, 8, and 16, whereas the number of CPU cores of the A and B part of the Tester was always 32.

The number of connections was always 1600. (Dns64perf++ used 16 thread pairs, and the number of source port numbers per sending thread was set to 100. The destination port number was always 53, the well-known port number for DNS.) The reason behind this low number of connections was to use the same number of connections as with MAP-T, which had the limit of 2048 source port numbers per subscriber.

The results are presented in Figure 15. It is well visible that the scalability of the system is moderate, the addition of the last 8 cores results in only 4% performance increase.

num. CPU cores	1	2	4	8	16
Median (qps)	165,403	236,869	425,021	510,155	530,064
Minimum (qps)	163,280	236,185	420,311	499,999	529,881
Maximum (qps)	167,041	237,553	425,782	510,321	530,222
current/previous	-	1.43	1.79	1.20	1.04

Figure 15: The number of resolved queries per second as a function of the number of active CPU cores of the CLAT and PLAT devices of the 464XLAT test system

5.2. MAP-T Scalability Measurements and Results

The measurement setup for the scalability analysis of MAP-T is shown in Figure 16.

The configuration of the test system and the measurement method was the same as with 464XLAT.

The results are presented in Figure 17. It is well visible that the scalability of the system is much better now.

All further details can be found in our open access paper [LEN2022].

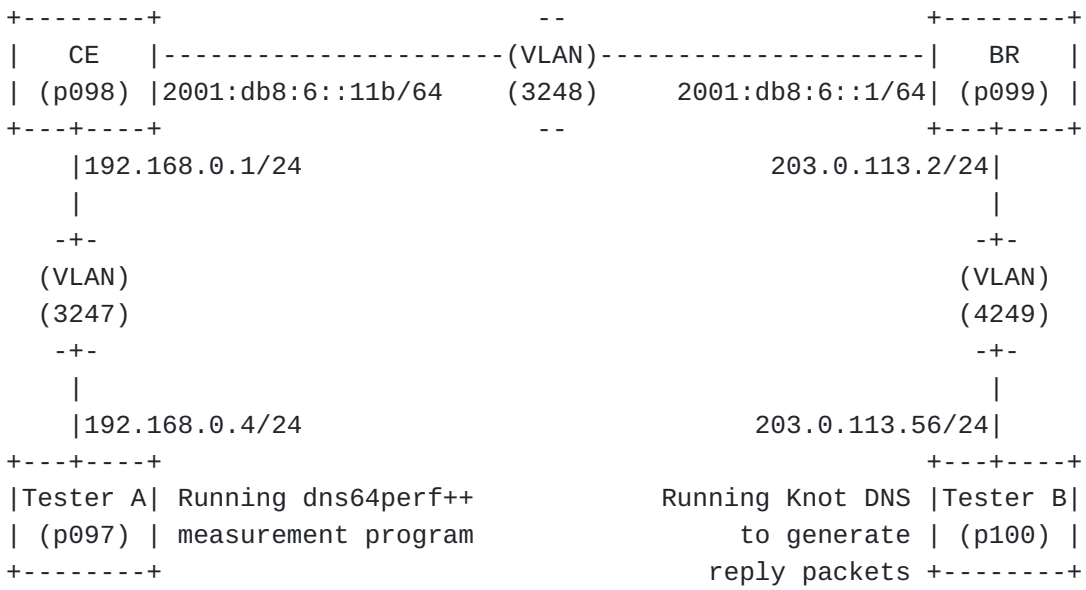

Figure 16: Test setup for benchmarking MAP-T using DNS traffic

num. CPU cores	1	2	4	8	16
Median (qps)	143,498	250,858	475,051	835,180	1,183,615
Minimum (qps)	140,624	245,311	470,311	812,499	1,179,686
Maximum (qps)	145,314	251,953	475,818	837,295	1,184,387
current/previous	-	1.75	1.89	1.76	1.42

Figure 17: The number of resolved queries per second as a function of the number of active CPU cores of the CE and BR devices of the MAP-T test system

6. Acknowledgements

The measurements were carried out by remotely using the resources of NICT StarBED, 2-12 Asahidai, Nomi-City, Ishikawa 923-1211, Japan. The author would like to thank Shuuhei Takimoto for the possibility to use StarBED, as well as to Satoru Gonno and Makoto Yoshida for their help and advice in StarBED usage related issues.

The author would like to thank Ole Troan for his comments on the v6ops mailing list, while the scalalability measurements of iptables were intended to be a part of the draft later published as [RFC9313].

7. IANA Considerations

This document does not make any request to IANA.

8. Security Considerations

TBD.

9. References

9.1. 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>.

[RFC9313] Lencse, G., Palet Martinez, J., Howard, L., Patterson, R., and I. Farrer, "Pros and Cons of IPv6 Transition

Technologies for IPv4-as-a-Service (IPv4aaS)", RFC 9313, DOI 10.17487/RFC9313, October 2022, <https://www.rfc-editor.org/info/rfc9313>.

9.2. Informative References

[I-D.ietf-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-ietf-bmwg-benchmarking-stateful-04, 12 September 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-bmwg-benchmarking-stateful-04>.

[JOOLMX] NIC Mexico, "Jool: SIIT and NAT64", The home page of Jool, 2022, <https://jool.mx/>.

[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>.

[LEN2022] Lencse, G. and N. Nagy, "Towards the scalability comparison of the Jool implementation of the 464XLAT and of the MAP-T IPv4aaS technologies", International Journal of Communication Systems, DOI: 10.1002/dac.5354, 22 September 2022, <https://onlinelibrary.wiley.com/doi/10.1002/dac.5354>.

[LEN2023] Lencse, G., Shima, K., and K. Cho, "Benchmarking methodology for stateful NAT64 gateways", Computer Communications, DOI: 10.1016/j.comcom.2023.08.009, 1

October 2023, <https://www.sciencedirect.com/science/article/pii/S0140366423002931>.

[NETFLTR] The Netfilter's webmasters, "Netfilter: Firewalling, NAT, and packet mangling for Linux", The netfilter.org project home page, 2021, <https://www.netfilter.org/>.

[PFBOOK] Hansteen, P. N. M., "The Book of PF: A No-Nonsense Guide to the OpenBSD Firewall", 3rd ed., San Francisco, No Starch Press, ISBN: 978-1-59327-589-1, 2014, <https://nostarch.com/pf3>.

Appendix A. Change Log

A.1. 00

Initial version: scale up of iptables.

A.2. 01

Added the scale up of Jool.

A.3. 02

Connection tear down rate measurements of iptables and Jool.

A.4. 03

Added: introductions to iptables and Jool, connection tracking table capacity measurement of iptables and connection validation measurement of Jool.

A.5. 04

Added: scalability comparison of the Jool implementation of the 464XLAT and of the MAP-T IPv4aaS technologies using DNS traffic.

A.6. 05

Added: scalability of OpenBSD PF.

Author's Address

Gábor Lencse
Széchenyi István University
Győr
Egyetem tér 1.
H-9026
Hungary

Email: lencse@sze.hu