Computer Communications 219 (2024) 243–258

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Benchmarking methodology for IPv4aaS technologies: Comparison of the scalability of the Jool implementation of 464XLAT and MAP-T

Gábor Lencse [a,*], Ádám Bazsó [b]

[a] Department of Telecommunications, Faculty of Mechanical Engineering, Informatics and Electrical Engineering, Széchenyi István University, Egyetem tér 1, Győr, H-9026, Hungary
[b] Cybersecurity and Network Technologies Research Group, Faculty of Mechanical Engineering, Informatics and Electrical Engineering, Széchenyi István University, Egyetem tér 1, Győr, H-9026, Hungary

ARTICLE INFO

Keywords:
464XLAT
IPv4aaS
Jool
MAP-T
Scalability

ABSTRACT

A novel method is proposed for the performance and scalability measurements of the IPv4-as-a-Service (IPv4aaS) technologies. It works according to the dual Device Under Test (DUT) setup of RFC 8219 and is suitable for benchmarking any of the five IPv4aaS technologies: Combination of Stateful and Stateless Translation (464XLAT), Dual-Stack Lite (DS-Lite), Lightweight 4over6 (Lw4o6), Mapping of Address and Port with Encapsulation (MAP-E), and Mapping of Address and Port using Translation (MAP-T). The method is based on the reduction of the aggregate of Customer Edge (CE) and Provider Edge (PE) devices to a stateful network address translation from IPv4 to IPv4 (stateful NAT44) gateway. The most important advantage of the novel method is that a stateful NAT44 tester can be used instead of a technology-specific tester, which usually does not exist. The proposed method is validated by the examination of the performance and scalability of the Jool implementation of 464XLAT and MAP-T. Scalability is defined by both (1) how performance increases with the number of active Central Processing Unit (CPU) cores; and (2) how performance decreases with the increasing number of concurrent sessions. Maximum connection establishment rate and throughput are used as performance metrics. The scalability of 464XLAT and MAP-T is measured from 1 to 16 CPU cores and from 1 million to 256 million connections. The measurement details and results are fully disclosed and discussed.

1. Introduction

Even though the public IPv4 address pool of the *Internet Assigned Numbers Authority* (IANA) was depleted in 2011 [1], the transition of the Internet from *Internet Protocol version 4* (IPv4) to *Internet Protocol version 6* (IPv6) has not yet been completed. Several *Internet Service Providers* (ISPs) use *Carrier-grade NAT* (CGN) to mitigate the shortage of public IPv4 addresses, whereas others decided to go ahead and eliminate IPv4 from their access and core networks. However, there are some IPv4-only applications and some users abide by them. Therefore, ISPs still need to provide their customers with IPv4 Internet access. To that end, five *IPv4-as-a-Service* (IPv4aaS) technologies have been developed [2]. In terms of the technology used for access and core network traversal, they can be classified into two categories.

- *Combination of Stateful and Stateless Translation* (464XLAT) and *Mapping of Address and Port using Translation* (MAP-T), which use double translation (first, from IPv4 to IPv6 and then from IPv6 to IPv4)
- *Dual-Stack Lite* (DS-Lite), *Lightweight 4over6* (Lw4o6), *Mapping of Address and Port with Encapsulation* (MAP-E), which encapsulate the IPv4 packets into IPv6 packets and then de-encapsulate them.

The five IPv4aaS technologies have various similarities and differences (e.g., their stateful or stateless nature in the core of the ISP network) and thus they have several advantages and disadvantages as discussed in Ref. [3]. The performance and scalability of the IPv4aaS technologies are important decision factors when network operators select the most suitable IPv4aaS technology for their specific needs. However, the performance analysis of the five IPv4aaS technologies is rather uncharted territory. Al-hamadani in Ref. [4] surveyed several research papers regarding the performance analysis of IPv6 transition technologies and, according to table 2 of that paper, only one of the surveyed research papers addressed any of the five IPv4aaS

* Corresponding author.
E-mail address: lencse@sze.hu (G. Lencse).

https://doi.org/10.1016/j.comcom.2024.03.007
Received 11 December 2023; Received in revised form 4 February 2024; Accepted 7 March 2024
Available online 15 March 2024

technologies, [5]. The authors of [5] covered all of them except Lw4o6. They measured the performance of 464XLAT, DS-Lite, MAP-T, and MAP-E by means of *round-trip-delay*, *jitter*, *throughput*, and *packet loss* using the Asamap Vyatta software executed by Cisco UCS C200 M2 servers. Whereas the authors of this paper acknowledge the significance of their pioneering work, they contend that instead of a performance comparison using some specific hardware (which is rather obsolete at the time of writing this paper), network operators would benefit much more from the *scalability comparison* of the various IPv4aaS solutions. Scalability is defined by both (1) how performance increases with the number of active Central Processing Unit (CPU) cores; and (2) how performance decreases with the increasing number of concurrent sessions. Such measurements were already performed regarding the Jool implementation of the 464XLAT and MAP-T technologies in Ref. [6]. However, that measurement method had several limitations and it also did not comply with RFC 8219 [7] (Please refer to Section 3.2 for the details.).

When this research began, the authors were faced with the following key technical challenges.

1. Except for 464XLAT, *no RFC 8219 compliant Testers existed* for benchmarking the other four IPv4aaS technologies.
2. Although all five IPv4aaS technologies can carry IPv4 traffic, *they cannot be benchmarked using legacy RFC 2544* [8] *compliant Testers,* as shown in Section 2.2.
3. *No RFC 8219-compliant methodology has been defined for measuring the scalability* of the IPv4aaS IPv6 transition technologies as pointed out in Section 3.

It should be noted that RFC 8219 reflects the state of the art in the benchmarking of IPv6 transition technologies. Any measurements that do not comply with it can give valuable insight into the performance of the examined IPv6 transition technologies, but cannot provide its full benefits.

To address these challenges, the authors set the following goals.

1. To provide an RFC 8219-compliant methodology that does not require a technology-specific tester for each technology and is suitable for the performance and scalability measurements of each of the five IPv4aaS technologies.
2. To validate the methodology by actual measurements with two different IPv4aaS technologies.
3. To compare the performance and scalability of the Jool implementation of the 464XLAT and MAP-T technologies in an RFC 8219-compliant way.

To this end, this paper proposed a new methodology that makes it possible to benchmark any of the five IPv4aaS technologies in an RFC 8219-compliant way without the need for technology-specific testers. This was achieved by the reduction of the dual *Device Under Test* (DUT) setup of RFC 8219 to a single DUT setup of a stateful NAT44 gateway, which was then benchmarked according to the Internet-Draft [9]. For measuring scalability, the proposed method was the one defined in the Internet-Draft. Therefore, the authors believed that the proposed measurement methodology for benchmarking IPv4aaS technologies was novel and had never been used by anyone else before. Moreover, by benchmarking the Jool implementation of the 464XLAT and MAP-T technologies, the proposed benchmarking methodology was validated.

The remainder of this paper is structured as follows. In Section 2, the methodological issues of benchmarking the five IPv4aaS technologies are discussed: first, a short high-level summary of the operation of the five IPv4aaS technologies is given and one of their common properties that must be taken into consideration is highlighted; then the relevant requirements of RFC 8219 are mentioned and the methodological gap regarding stateful technologies is pointed out; finally, a novel RFC 8219 compliant benchmarking methodology for the performance and scalability measurement of IPv4aaS technologies is outlined. In Section 3, the results of two previous papers are summarized and the shortcomings of the applied measurement methods are detailed. The rest of the current paper is a case study that demonstrates and validates the proposed methodology. In Section 4, the hardware and software measurement environment and baseline measurements (IPv4 and IPv6 packet forwarding tests) are introduced. In Section 5, the scalability measurements of the Jool implementation of 464XLAT are disclosed, including results and their discussion. In Section 6, the same is done with MAP-T using two different measurement setups. Section 7 contains a discussion of the results and authors' plans for future research. Section 8 is the conclusion of the paper.

2. The problem of a benchmarking methodology for IPv4aaS technologies and its proposed solution

2.1. High-level operation of the five IPv4aaS technologies

In this section, a brief overview of the operation of the five IPv4aaS technologies is given to show why their RFC 8219-compliant benchmarking is a problem to be solved. A common property of all of them is that they all use a Customer Edge (CE) device and Provider Edge (PE) device to facilitate the traversal of the IPv4 traffic of the user over the IPv6-only access and core network of the ISP. Moreover, they have their technology-specific names and operations, which are described as follows.

2.1.1. 464XLAT

The *customer-side translator* (CLAT) of 464XLAT [10] performs a *stateless network address translation from IPv4 to IPv6 (stateless NAT46)* translation (also called *stateless IP/ICMP translation* [SIIT] [11]) to send the IPv4 traffic of the user over the IPv6-only access and core network of the ISP. When the packets arrive at the *provider-side translator* (PLAT), it performs a *network address and protocol translation from IPv6 clients to IPv4 servers* (stateful NAT64) [12] translation and the packets are forwarded to the IPv4 Internet (Fig. 1). The reply packets are translated in the reverse way. As for PLAT, the reverse translation uses the information stored in its so-called *connection tracking table*. There is a state in the central element (PLAT) of 464XLAT that can be a problem regarding its scalability.

2.1.2. DS-lite

The *Basic Bridging BroadBand* (B4) element of DS-Lite [13] encapsulates the IPv4 traffic of the user into IPv6 packets. The *address family transition router* (AFTR) decapsulates the IPv4 packet of the user from the IPv6 packet and (with some simplification) it performs a stateful NAT44 translation. Following this, the packet is forwarded to the IPv4 Internet, as shown in Fig. 2. To be precise, this is not simply a stateful NAT44 translation, because the IPv6 address of the B4 device (called *softwire*-ID) is also stored in the connection tracking table of the AFTR so that it can distinguish the packets of different users that accidently have the same five-tuple (source IPv4 address, source port number, destination IPv4 address, destination port number, and protocol number). Similar to 464XLAT, there is a state in the central element (AFTR) of DS-Lite.

2.1.3. Lw4o6

Lw4o6 [14] is an extension of DS-Lite. The motivation of its design was to remove the state from the central element and thus make the

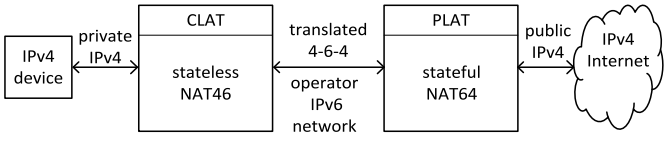


Fig. 1. Overview of the 464XLAT architecture [3].

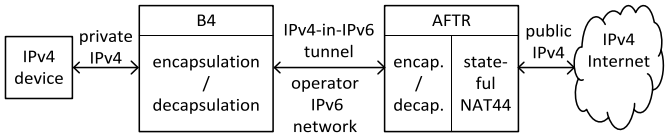


Fig. 2. Overview of the DS-Lite architecture [3].

solution more scalable. It is achieved in a way that each subscriber is given only a specific limited source port range of a public IPv4 address. The *lightweight* B4 (lwB4) element of Lw4o6 first performs a stateful NAT44 translation on the IPv4 packet of the user, transforming its IPv4 source address and source port number to the specific public IPv4 address and the limited source port range assigned to the given subscriber. It then encapsulates the resulting IPv4 packet into an IPv6 packet. The *lightweight AFTR* (lwAFTR) decapsulates the IPv4 packet from the IPv6 packet and forwards it to the IPv4 Internet, as shown in Fig. 3. Thus, the operation of the central element is stateless. In the reverse direction, the packets are forwarded to the proper lwB4 device using *Address plus Port* (A + P) routing [15].

2.1.4. MAP-E

It can be said that MAP-E [16] is a kind of generalization of Lw4o6, although it uses a unique ruleset, called MAP rules. Not considering the mapping rules, its high-level operation is rather similar to that of Lw4o6. The CE element of MAP-E first performs a stateful NAT44 translation on the IPv4 packet of the user. This means that the IPv4 source address and source port number are translated to the specific public IPv4 address and the limited source port range assigned to the CE device. Subsequently, the CE device encapsulates the resulting IPv4 packet into an IPv6 packet. The *Border Relay* (BR) router decapsulates the IPv4 packet from the IPv6 packet and forwards it to the IPv4 Internet, as shown in Fig. 4. Thus, the operation of the central element is stateless.

2.1.5. MAP-T

The operation of MAP-T [17] is similar to that of MAP-E but it uses double translation (like 464XLAT) for access and core network traversal instead of encapsulation and decapsulation. Thus, the CE element of MAP-T first performs a stateful NAT44 translation on the IPv4 packet of the user. (The IPv4 source address and source port number are translated to the specific public IPv4 address and the limited source port range is assigned to the CE device.) Then the CE device performs a stateless NAT46 translation to transform the IPv4 packet into an IPv6 packet. The BR router performs a stateless NAT64 translation (the reverse of the stateless NAT46 translation is performed by the CE device) to transform the IPv6 packet into an IPv4 packet. It then forwards the IPv4 packet to the IPv4 Internet (Fig. 5). Thus, the operation of the central element is stateless.

2.1.6. Summary

The high-level operation of the five IPv4aaS technologies is summarized in Table 1. The 464XLAT and DS-Lite technologies are called "stateful" as they have a state in their PE device. The other three technologies are called "stateless" because they do not have a state in the PE device; however, they do have a state, as well, but it is in the CE device. Regarding the scalability of the technologies, a *state close to the end-user is generally not seen as problematic as a state in the middle of the network* [3]; however, if a state exists anywhere in the system, it causes hardship

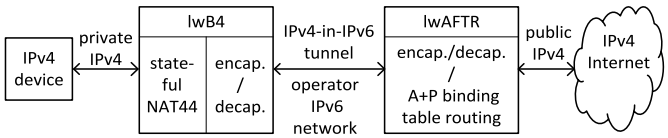


Fig. 3. Overview of the Lw4o6 architecture [3].

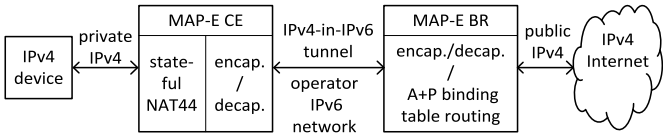


Fig. 4. Overview of the MAP-E architecture [3].

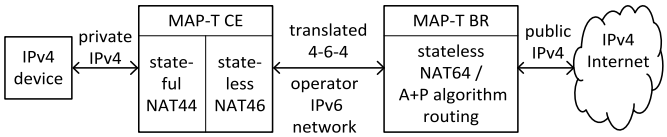


Fig. 5. Overview of the MAP-T architecture [3].

Table 1
Summary of the high-level operation of the five IPv4aaS technologies.

It is stateful in the	Service provider network traversal technology	
	double translation	encapsulation/decapsulation
PE device	464XLAT	DS-Lite
CE device	MAP-T	MAP-E, Lw4o6

for benchmarking, as discussed in Section 2.2.

2.2. Some important requirements of RFC 8219

A comprehensive benchmarking methodology was defined for all kinds of network interconnect devices by RFC 2544 [8] in 1999. Moreover, it still determines how commercial network performance testers work today. Theoretically, it was an IP version independent solution, but its approach and examples reflected IPv4. As time passed, its methods were updated in different ways. Originally, RFC 2544 used a fixed test frame format including port numbers. In 2008, the RFC 4814 [18] recommended using pseudorandom port numbers. An upgrade for IPv6 specificities was given by RFC 5180 [19] in the same year. It explicitly declared that IPv6 transition technologies were outside its scope. RFC 8219 [7] defined a benchmarking methodology for the IPv6 transition technologies in 2017.

RFC 8219 has been built on its predecessors.

- it has reused several measurement procedures, e.g., *throughput*, *frame loss rate*, etc. (unmodified);
- it has also kept the requirement of testing with bidirectional traffic (using the same speed in both directions), although it has added an optional testing with unidirectional traffic.

To be able to handle the high number of IPv6 transition technologies [20] efficiently, RFC 8219 classified them into a small number of categories regarding the method used for access and core network traversal and then defined the appropriate benchmarking methodology for each category. For this research, the relevant categories are *double translation* and *encapsulation* technologies. For these categories, the *Dual DUT setup*

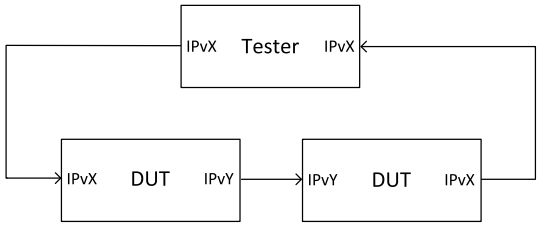


Fig. 6. Dual DUT test setup [7].

is recommended. As shown in Fig. 6, there is a single *Tester* and there are two DUTs. In the current case, *DUT 1* and *DUT 2* were the CE and PE devices, which were benchmarked together. As the usage of this test setup may hide potential asymmetries, the usage of the *Single DUT* test setup is also recommended (Fig. 7). Regarding the benchmarking of the five IPv4aaS technologies, the problem is that the Single DUT test setup requires a Tester that can handle the specific traffic of the given implementation. For example, for testing a MAP-E BR device, the Tester should send IPv6 packets that contain embedded IPv4 packets using IP addresses complying with the mapping rules in one direction, and it should send IPv4 traffic in the other direction. It should also be able to decode the packets to check if they arrived correctly. Not having such a specific tester, the only feasible solution could be to use the Dual DUT setup, where CE and PE devices are benchmarked together, as shown in Fig. 8. Thus, the problem of benchmarking any of the IPv4aaS technologies was simplified to the problem of benchmarking a stateful NAT44 gateway. Moreover, this brings up the problem mentioned in Section 2.1.6; however, in this case, the Tester is expected to use only IPv4 traffic (the aggregate of CE and PE devices is a stateful system and it performs stateful NAT44). Furthermore, this is the same in the case of all five IPv4aaS technologies, as they are all stateful somewhere (either in the CE or PE device). Therefore, testing with bidirectional traffic will not work unless some preliminary arrangements are made and special care is taken (as described in Section 2.3). This is the reason why the aggregate of CE and PE devices cannot be benchmarked by using legacy RFC 2544 compliant Testers, even if it can be called an *IPv4 system* when it is observed from outside.

Regarding the standard frame sizes, RFC 8219 follows the approach of its predecessors and also takes care that translation and encapsulation change the frame size.

2.3. Benchmarking methodology for stateful NAT44 gateways

The Internet-Draft [9] proposed a methodology of how stateful NATxy (x, y are in {4, 6}) gateways might be benchmarked in compliance with RFC 8219 and RFC 4814. Here, only a very brief overview of the method is given focusing on stateful NAT44, which is relevant in the current case. The test setup for benchmarking stateful NAT44 gateways is shown in Fig. 9.

2.3.1. Problems to solve

Literally following the requirements of RFC 4814 and RFC 8219 would result in the following two problems.

1. Using pseudorandom source and destination port numbers from the entire ranges recommended by RFC 4814 in the private to public direction, would result in a DoS (Denial of Service) attack against the connection tracking table of the stateful NAT44 gateway due to the capacity exhaustion. RFC 4814 requires testing with 64,512 different source port numbers and 49,151 different destination port numbers resulting in 3,170,829,312 source port number destination port number combinations.
2. Using pseudorandom source and destination port numbers in the public to private direction would result in a drop of test frames that do not belong to an existing connection, that is, the vast majority of the test frames.

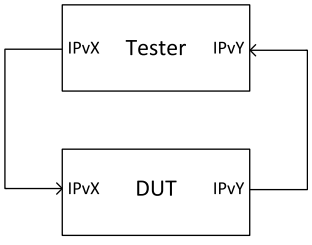


Fig. 7. Single DUT test setup [7].

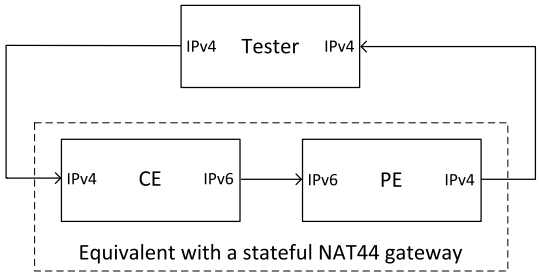


Fig. 8. Proposed test setup for any of the IPv4aaS technologies.

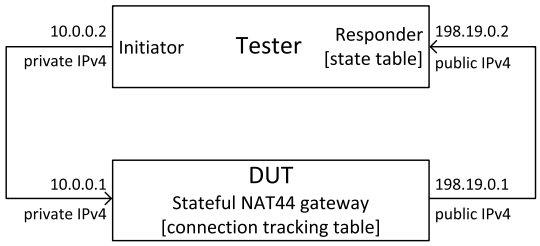


Fig. 9. Test setup for benchmarking stateful NAT44 gateways [9].

Therefore, special care must be taken as described below.

2.3.2. The basic ideas of the solution

To avoid the above-mentioned DoS attack, the source and destination port number ranges for the private to public direction are limited. Their sizes were used as a parameter, which is explained later in Section 5.4.

The method uses two *test phases*. To acquire valid four-tuples (source IP address, source port number, destination IP address, destination port number), which belong to a connection that is present in the connection tracking table, *test phase 1* is used.

Test phase 1 serves two purposes.

- The connection tracking table of the DUT is filled.
- The state table of the Responder is filled with valid four-tuples.

Test phase 1 can be used without *test phase 2* to measure the *maximum connection establishment rate*, which is a new performance metric, specific to stateful devices.

Test phase 2 must always be preceded by test phase 1. The “classic” measurement procedures of RFC 8219 (throughput, frame loss rate, latency, etc.) can be performed in test phase 2.

To ensure clear and repeatable measurements, testing under the following conditions is recommended.

1. During test phase 1, all test frames should create a new connection.
2. During test phase 2, test frames should never create a new connection.
3. Connections should never be deleted (due to timeout or replacement) during test phase 1 or test phase 2. (There is a separate measurement for the connection tear-down rate.)

Condition 1 is ideal for measuring the maximum connection establishment rate and is also optimal for decreasing the duration of test phase 1 when it is followed by test phase 2.

Conditions 2 and 3 are ideal for the throughput, latency, frame loss rate, etc. tests.

These conditions can be easily achieved by.

- using a sufficiently large and empty connection tracking table for each test;
- using pseudorandom enumeration of all possible port number combinations (with the used source and destination port number ranges) in test phase 1;
- using a properly high timeout value in the DUT.

Please refer to the Internet-Draft [9] for all the details.

The stateful extension of the **`siitperf`** measurement tool was developed in parallel with the Internet-Draft and its version used for the measurements in this paper is documented in Ref. [21].

The proposed measurement method was validated by performing its tests with three radically different stateful NAT64 implementations [22].

2.3.3. Method for measuring scalability

Scalability regarding how performance increases with the number of active CPU cores can be expressed by the relative scale-up defined by (1), where the numerator can express any performance characteristic measured using n number of active CPU cores.

$$S_{CPU}(n) = \frac{P_{CPU}(n)}{n} \tag{1}$$

It should be noted that this definition requires a measurement series to be performed, which is the n number active of CPU cores should be increased from 1 to the maximum available number of CPU cores. Using values for n as the powers of 2 (1, 2, 4, 8, 16, etc.) may effectively reduce the number of tests necessary.

Scalability regarding how performance decreases with the increasing number of concurrent sessions is expressed by (2), where the numerator can express any performance characteristic measured using n_i connections, and the denominator is the same performance characteristic measured using n_0 connections.

$$S_C(n_i, n_0) = \frac{P_C(n_i)}{P_C(n_0)} \tag{2}$$

It should be noted that n_0 is the lowest realistic number of connections, which is typically several orders of magnitudes higher than 1. The highest value of n_i depends on the range of interest. In the Internet-Draft [23], it is increased until the hardware limit is reached. The policy of doubling the number of connections in each step can be a suitable approach when fine-grain analysis is needed. In Ref. [22], it was increased 10-fold to reduce the number of tests necessary.

3. Findings and shortcomings of the previous tests

Here, a summary is given about two previous efforts.

3.1. Measuring the scalability of four IPv4aaS technologies

Georgescu et al. [24] claim that no previous studies dealt with the scalability analysis of IPv6 transition technologies. Their paper covers four of the five most important IPv4aaS technologies, namely 464XLAT, DS-Lite, MAP-E, and MAP-T.

3.1.1. Measurement method

In its section 3, the paper contains a survey of methods for measuring scalability using different definitions. The authors call their choice "load scalability" and they measured *how the performance of the examined systems degrades with the increase of the load*. They measured four performance characteristics: round-trip delay, jitter, throughput, and packet loss. The *distributed Internet traffic generator* (D-ITG) [25] was used for packet generation in two different setups. The first setup contained 4 servers to execute the "ITGSend" function to generate traffic, the CE function of the examined technology, the PE function of the same technology, and the "ITGRecv" function to receive the traffic and send it back to the server executing the "ITGSend" function. The second setup contained 31 servers: except for the PE function, the number of servers executing the other three functions was increased tenfold.

3.1.2. Limitations of the method

Although the results can give an important insight into the "load scalability" of the examined IPv4aaS implementations, they give no information about how their performance scales up with the number of CPU cores.

However, as the ongoing development in the hardware sector favors an increasing number of processing units over an increasing speed of a single unit [26], the authors consider it important to measure how the performance of the examined IPv4aaS implementations scales up with the number of CPU cores.

3.2. Measuring the scalability of 464XLAT and MAP-T

The author's team has made a previous attempt towards the scalability comparison of the Jool implementation of 464XLAT and MAP-T technologies as presented in Ref. [6].

3.2.1. Measurement method

As for the measurement tool, the **`dns64perf++`** [27] program was used, which was originally developed for benchmarking *Domain Name System (DNS) extensions for network address translation from IPv6 clients to IPv4 servers* (DNS64) servers at moderate query rates [28]. However, later its performance was significantly increased and it was made suitable for benchmarking authoritative DNS servers up to 3,000,000 queries per second (qps) [29]. To ensure reply packets, a Knot DNS server was set up, as it could produce answers at a sufficiently high rate [29].

As for measuring scalability, the number of active CPU cores was set to 1, 2, 4, 8, and 16 in both CE and PE devices and the performance of the system was measured (by means of the number of successfully forwarded DNS queries and replies).

To determine if the CE or PE device was the bottleneck, their CPU utilization was also measured (in further tests).

3.2.2. Summary of findings

It was found that the Jool implementation of MAP-T scaled up better than the Jool implementation of 464XLAT.

It was discovered from the CPU utilization results that the PLAT was the bottleneck when 464XLAT was tested, and the CE was the bottleneck when MAP-T was tested.

3.2.3. Limitations of the tests and how to overcome them

The measurements had several limitations.

1. Scalability could not be measured regarding the number of concurrent sessions.
2. The maximum connection establishment rate and throughput could not be clearly measured but rather their certain combination.
3. The standard frame sizes required by RFC 8219 could not be used.
4. The authors did not try influencing which device (CE or PE) was the bottleneck, that is, the performance of which device was ultimately measured.

The authors of the current paper believe that these limitations deserve some discussion, especially how serious they are and how they could be eliminated.

Limitation 1 came from the fact that the destination port number of the DNS queries was always the same fixed value (53) due to the nature of the measurement tool. (The source port range could not be widened due to the nature of MAP-T.) The authors consider this limitation as the most serious one, as the scalability regarding the number of sessions is very important for the ISPs, as it depends on the number of active users

and the nature of their applications. By allowing the user to specify both the source and the destination port ranges to be used, **siitperf** can fully eliminate this limitation.

Limitation 2 also came from the nature of the measurement tool. Likely, it is not very serious from an ISP point of view, but it is rather annoying from an analytical point of view. Of course, one can easily measure the maximum connection establishment rate and throughput separately using **siitperf**.

Limitation 3 also came from the nature of the measurement tool; the DNS queries and replies have their specific lengths. Whereas this one seems to have a serious shortcoming at first glance, the first author's experience shows that if the bottleneck is the CPU capacity and not the speed of the network, then packet length does not make a significant impact on the number of transferred packets per second. (This was experienced with various SIIT implementations using 128 bytes and 1280 bytes frame sizes [30] and the Jool stateful NAT64 implementation using 64 bytes and 1024 bytes frame sizes [21].) As for **siitperf**, it supports all Ethernet frame sizes from 64 bytes to 1518 bytes.

Limitation 4 partially came from the decision of the authors of [6] to set the same number of active CPU cores for CE and PE devices. As one PE should serve a high number of CEs in an ISP scenario, *the scalability of the PE device* (in this case: PLAT of 464XLAT and BR of MAP-T) is the important question. Thus, setting the number of CPU cores to the maximum value in the CE device and changing the number of the CPU cores from 1 to 16 in the PE device is a better choice for the current experiments. However, it is still possible that the CE of MAP-T becomes the bottleneck when both devices have 16 active cores, thus it can be said that this limitation partially comes from the Dual DUT setup. This issue is revisited in Section 5.3.

4. Hardware and software measurement environment and baseline measurements

4.1. Hardware and software measurement environment

The measurements were carried out remotely using the resources of the NICT Hokuriku StarBED Technology Center, Japan. In all, seven so-called "P" series nodes were used, which were Dell PowerEdge R430 servers with the following relevant main parameters.

- two Intel Xeon E5-2683v4 2.1 GHz CPUs with 16 cores each;
- twelve 32 GB, 2400 MHz DDR4 RAM modules (in all 384 GB);
- Intel X540 dual-port 10 Gbps NIC (for experimenting).

Based on the previous benchmarking experiments of the authors ([28,29], and [30]), Hyper-Threading and Turbo Boost were switched off in the BIOS of the servers to avoid scattered measurement results. This time the authors went one step further and set the clock frequency of all servers to a fixed 2.1 GHz using the **tlp** Linux package.

The servers were interconnected by a 10 Gbps switch using VLANs.

Debian 9.13 GNU/Linux operating system with its kernel version 4.9.0–14-amd64 was used on those servers that functioned as Testers because the compilation of **siitperf** required the Data Plane Development Kit (DPDK) version 16 contained in Debian 9.

Debian 10.11 GNU/Linux operating system with its kernel version 4.19.0–18-amd64 was used on those servers that functioned as DUTs because it was needed for Jool.

The version of Jool used was 4.1.255.3 (distributed as 4.2.0-rc2). The DPDK version used was 16.11.11–1+deb9u2.

4.2. Baseline measurements

Some "baseline" measurements were performed to check the performance of the measurement environment itself thus avoiding a bottleneck other than the examined IPv4aaS implementations. These were IPv4 and IPv6 packet forwarding tests. The test setup shown in Fig. 10 was used. The MAC addresses are presented in the figure because they had to be explicitly set for **siitperf**, as it was not able to reply to *Address Resolution Protocol* (ARP) or *Neighbor Discovery Protocol* (NDP) requests. IPv4 and IPv6 packet forwarding were enabled in the Linux kernel of DUT1 and DUT2. The *Receive-Side Scaling* (RSS) [31], also called *multi-queue receiving*, was set on all interfaces of the DUTs, so that the port numbers could also be taken into consideration when distributing the interrupts of packet arrivals among the active CPU cores. These used the appropriate versions of the following four commands (please refer to the two brace expansions to get them):

```
ethtool -N enp5s0f{0,1} rx-flow-hash udp{4,6} sdfn
```

The packet forwarding tests were executed using the following number of active CPU cores: 1, 2, 4, 8, and 16. It should be noted that 32 cores were not used because the authors found that the used 4.x Linux kernels did not distribute the interrupts evenly to the second 16 cores.

During the binary search for determining the throughput, 0.01 % packet loss was allowed, that is, the acceptance criterion was 99.99 %. Sometimes the same criterion was used in Refs. [28,29] too. For widespread usage of the non-zero acceptance criterion and its rationale, please refer to Ref. [32].

As required by RFC 8219, bi-directional traffic was used. It should be noted that **siitperf** reports the throughput results as *the number of forwarded frames* per *second* per *direction*. Thus, its results were multiplied by 2 to give *the number of all forwarded frames* per *second* as network performance testers usually do.

As for the IPv4 packet forwarding tests, a 64-byte frame size was used, all experiments were performed 20 times, and the median, minimum and maximum of the 20 results were calculated. The results are shown in Table 2. Moreover, the results are consistent (minimum and maximum values are quite close to each other) and the throughput scaled up quite well with the number of CPU cores. The moderate increase of the median at 2 cores (from 883,553 fps to 1,540,032 fps) can be explained by the cost of multi-core operation and especially by the NUMA[1] architecture of the CPUs: the even number of CPU cores (0, 2, 4, etc.) belong to NUMA node 0, and the odd number of CPU cores (1, 3, 5, etc.) belong to NUMA node 1 (Please refer to Section 4.2.1 of [29] for a comparison of the scale-up of different NUMA architecture CPUs.).

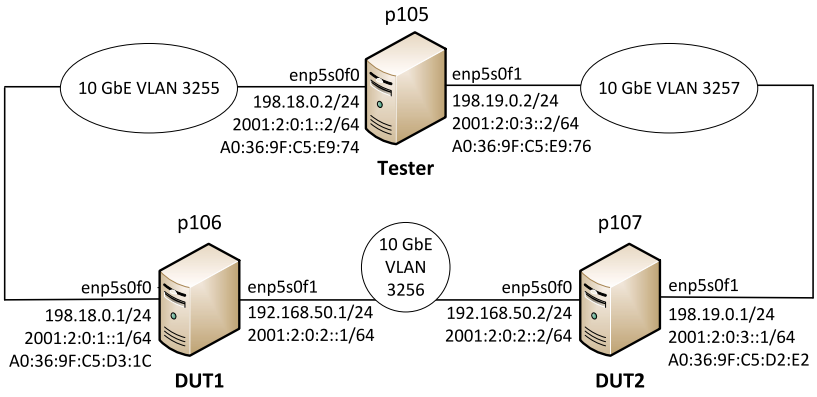
IPv6 packet forwarding tests were also performed with the only difference being that an 84-byte frame size was used. The results are shown in Table 3. From 1 to 8 cores, the IPv6 results were similar to the IPv4 results (they were somewhat lower and the scale-up was also somewhat poorer), but there was a significant drop in the IPv6 results at 16 cores. Its root cause would be interesting for Linux kernel developers, but this is beyond the scope of the current paper. For this research, the point is that the values were sufficiently high in all cases and thus it was ensured that the performance of the test system did not limit the results of the examined IPv4aaS solutions.

5. Scalability of the Jool implementation of 464XLAT

5.1. Measurement setup and tests

The measurement system followed the topology and settings shown in Fig. 11. The setup of Jool was quite straightforward. As for implementing CLAT, the SIIT kernel module of Jool (**jool-siit**) was used. *Explicit Address Mapping* was applied for the source address and an *IPv4-embedded IPv6 Address* was prepared using the NAT64 *Well-Known Prefix* (WKP) (i.e., 64:ff9b/96), for the destination address (for the IPv4 to IPv6 direction). The PLAT implemented stateful NAT64 using the NAT64 WKP. All commands are shown in Fig. 11. There is only one thing that needs an explanation: Jool has two operation modes, called "netfilter

[1] It is a memory system design where the memory access time depends on the location of the memory. A CPU can access its local memory faster than non-local memory. Please refer to Ref. [44] for a full depth explanation.



DUT1 IPv4 routing table

destination	next hop
198.18.0.0/24	-- (direct)
192.168.50.0/24	-- (direct)
198.19.0.0/24	192.168.50.2

DUT1 IPv6 routing table

destination	next hop
2001:2:0:1::/64	-- (direct)
2001:2:0:2::/64	-- (direct)
2001:2:0:3::/64	2001:2:0:2::2

DUT2 IPv4 routing table

destination	next hop
198.18.0.0/24	192.168.50.1
192.168.50.0/24	-- (direct)
198.19.0.0/24	-- (direct)

DUT2 IPv6 routing table

destination	next hop
2001:2:0:1::/64	2001:2:0:2::1
2001:2:0:2::/64	-- (direct)
2001:2:0:3::/64	-- (direct)

Fig. 10. Test setup for baseline measurements (IPv4 and IPv6 packet forwarding).

Table 2
Throughput of IPv4 Packet Forwarding as a Function of Active CPU cores.

Number of active CPU cores	1	2	4	8	16
Median (fps)	883,553	1,540,032	3,122,846	6,093,748	11,484,372
Minimum (fps)	861,314	1,499,874	3,038,012	5,937,488	11,374,998
Maximum (fps)	885,938	1,546,876	3,129,638	6,141,356	11,484,372
Relative scale up	1	0.871	0.884	0.862	0.812

Table 3
Throughput of IPv6 Packet Forwarding as a Function of Active CPU cores.

Number of active CPU cores	1	2	4	8	16
Median (fps)	763,142	1,225,970	2,431,870	4,633,994	5,577,862
Minimum (fps)	749,998	1,203,122	2,390,622	4,559,566	5,554,684
Maximum(fps)	764,064	1,228,516	2,437,502	4,648,468	5,588,866
Relative scale up	1	0.803	0.797	0.759	0.457

Jool" and "iptables Jool". In short, a "netfilter Jool" instance attempts to translate everything it can, whereas the packets to be translated must be explicitly given over to an "iptables Jool" instance by **`iptables`**. Both of them were tested in a few working points and no significant performance differences were found, thus "netfilter Jool" was used because of its somewhat simpler configuration (no need for **`iptables`** rules).

As for performance metrics, *maximum connection establishment rate* and *throughput* were used.

5.2. Scalability against the number of CPU cores

It was examined how the performance of the system scaled up with the number of active CPU cores of the PLAT from 1 to 16, whereas the number of active CPU cores of the CLAT was always 32. However, only the first 16 were used, as mentioned in Section 3.

It was known from the preliminary tests that the performance of the system significantly depended on the number of connections in the connection tracking table of the PLAT. For the scalability test, regarding the number of active CPU cores, 4,000,000 connections were chosen based on the recommendations of Vyacheslav Gapon for a high-loaded NAT server [33]. As the authors wanted to test 464XLAT and MAP-T with identical conditions and the source port number of MAP-T had to be limited due to its nature, 1–4000 and 1-1000 were used as the source and destination port number pools, respectively.

The parameters for this and all following measurements were the same as for the baseline measurements: 64 bytes IPv4 frame size (that is, 84 bytes for IPv6), 99.99 % acceptance criterion, and 20 executions of the tests.

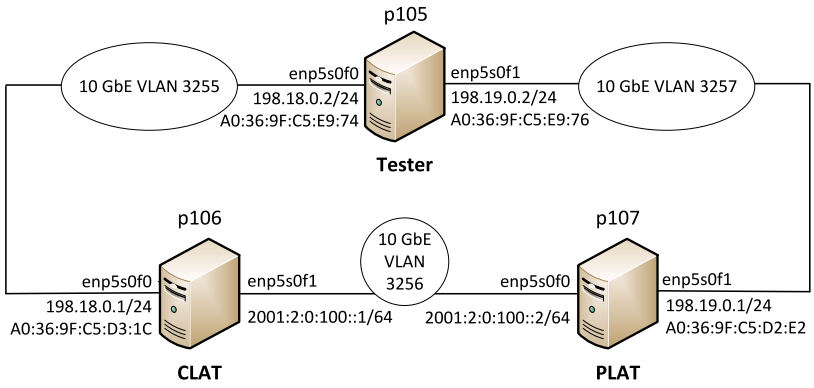
The maximum connection establishment rate and throughput measurement results of the Jool implementation of 464XLAT are shown in Table 4. Both quantities show a moderate scale-up from one to four CPU cores; the maximum connection establishment rate increased from 210,344 *connections* per *second* (cps) to 423,903 cps (relative scale up: 0.504), whereas the throughput increased from 217,946 fps to 472,135 fps (relative scale up: 0.542). Thus, the scale-up is significantly lower than that of the baseline measurements. Above 4 CPU cores, the addition of further active CPU cores did not result in a significant increase in the performance of the system. This is really bad news regarding the scalability of the Jool implementation of 464XLAT.

The somewhat better scalability of the throughput than that of the maximum connection establishment rate (e.g., 0.142 vs. 0.139 relative scale-up at 16 cores) can be explained by the difference that during the throughput measurements, the connection tracking table is mainly read when a test frame is processed (except for the update of the timeout time of the given connection). However, a new connection is registered into the connection tracking table for each test frame during the maximum connection establishment rate measurement.

5.3. Checking the bottleneck

To be able to tell whether the CLAT or PLAT was the bottleneck, the CPU utilization was measured during the tests re-executed using the found maximum in the previous paper [6]. Due to some fluctuations in CPU utilization during the preliminary tests, the authors performed the measurements 100 times and calculated the average for each second. Thus, quite smooth graphs were produced. Subsequently, those graphs were used to determine that the PLAT was the bottleneck.

However, when the authors studied the CPU usage graphs of the individual experiments of the current measurement, they decided not to use averaging (this will be explained later). The CPU idle time of the PLAT as a function of time using 1 CPU core and the measured maximum frame rate of the throughput measurement (218,778 fps) is shown in Fig. 12. It is visible that the idle time varied between 0 % and 20 %



CLAT configuration

```
modprobe jool_siit
ip route add 64:ff9b::/96 via 2001:2:0:100::2
jool_siit instance add --netfilter --pool6 64:ff9b::/96
jool_siit eamt add 198.18.0.2/24 2001:2:0:2::2/120
ip neigh add 198.18.0.2 lladdr a0:36:9f:c5:e9:74 dev \
  enp5s0f0 nud permanent
```

PLAT configuration

```
modprobe jool
ip route add 2001:2:0:2::/64 via 2001:2:0:100::1
jool instance add --netfilter --pool6 64:ff9b::/96
jool pool4 add 198.19.0.1 --udp 1-65535
ip neigh add 198.19.0.2 lladdr a0:36:9f:c5:e9:76 dev \
  enp5s0f1 nud permanent
```

Fig. 11. Test setup for scalability measurements of the Jool implementation of 464XLAT.

Table 4
Performance of the Jool Implementation of 464XLAT as a Function of the Active CPU cores, 4 M Connections.

Number of active CPU cores	1	2	4	8	16
Maximum connection establishment rate - Median (cps)	**210,344**	**328,212**	**423,903**	**448,817**	**459,089**
Maximum connection establishment rate - Minimum (cps)	208,396	324,595	417,039	440,014	451,170
Maximum connection establishment rate - Maximum(cps)	212,501	331,349	428,980	487,501	468,751
Maximum connection establishment rate - Relative scale up	1	0.780	0.504	0.267	0.136
Throughput (bidirectional traffic) - Median (fps)	**217,946**	**372,742**	**472,135**	**490,059**	**495,614**
Throughput (bidirectional traffic) - Minimum (fps)	215,622	365,548	465,608	484,248	487,106
Throughput (bidirectional traffic) - Maximum (fps)	218,778	375,002	478,126	495,314	500,398
Throughput (bidirectional traffic) - Relative scale up	1	0.855	0.542	0.281	0.142

during the 60s long interval of the throughput measurement. The close-to-zero values indicate that the PLAT would not have been able to transfer more packets per second. However, averaging the results of several measurements would produce about 10 % idle time, which could not prove that the PLAT was the bottleneck.

The authors' question is whether CPU utilization can be used to determine if the CLAT or PLAT is the bottleneck when both of them have 16 active CPU cores. To that end, one needs to check the CPU utilization of every single CPU core because an uneven distribution of the load can

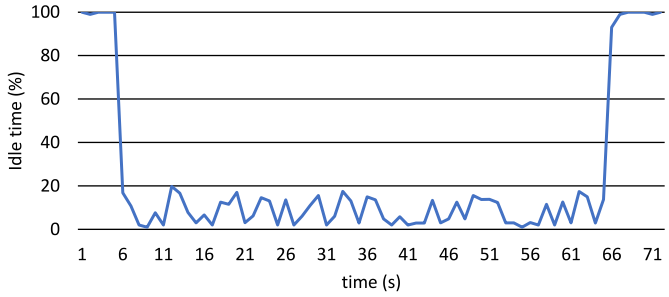


Fig. 12. CPU idle time of the PLAT using 1 CPU core and maximum frame rate of the throughput measurement.

cause frame loss, even if some of the cores have free capacity. The CPU idle time of the CLAT and PLAT as a function of time using 16 CPU cores and measured maximum frame rate of the throughput measurement (500,398 fps) is shown in Figs. 13 and 14, respectively. The fact that the idle time of all CPU cores of the CLAT was above 80 % during the entire test and the idle time of all CPU cores of the PLAT was around 60 % during the entire test makes it likely that the bottleneck was not the CLAT. However, this result does not explain why the PLAT could not handle more packets per second, as it had free CPU capacity. The authors suspect that access to the connection tracking table was the cause of the bottleneck.

It should be noted that the stateful operation does not necessarily

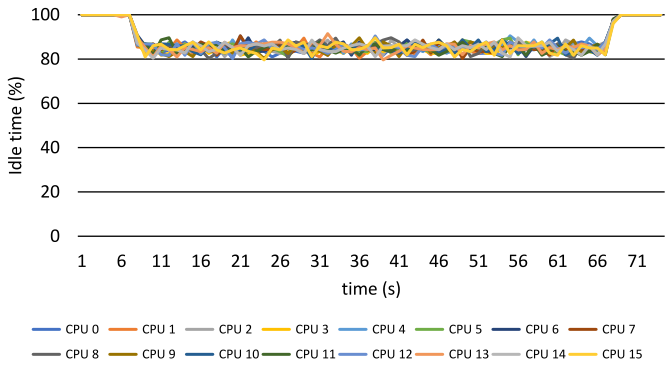


Fig. 13. CPU idle time of the CLAT using 16 CPU cores and maximum frame rate of the throughput measurement.

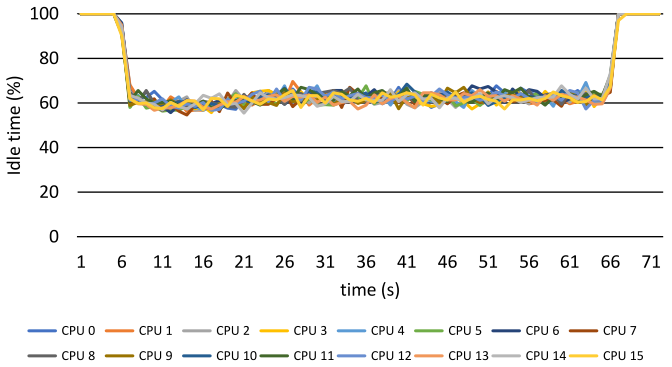


Fig. 14. CPU idle time of the PLAT using 16 CPU cores and maximum frame rate of the throughput measurement.

lead to such mediocre scalability. For example, the **`iptables`** stateful NAT44 implementation scales up much better than Jool. According to Ref. [23], the median throughput of **`iptables`** scaled up from 414,900 fps at 1 CPU core to 4,557,000 fps at 16 CPU cores, which is about an 11-fold increase and 0.686 relative scale-up in contrast to the current 0.136 relative scale-up. Thus, the authors believe that a design or implementation issue caused the poor scalability of Jool.

The CPU utilization of the PLAT was also examined during the maximum connection establishment rate measurement using a single CPU core and 212,501 cps during test phase 1. The CPU idle time of the PLAT as a function of time is displayed in Fig. 15. The filling of 4 million connections into the connection tracking table lasted somewhat less than 20 s. The idle time exhibited a decreasing tendency, which can be explained by the fact the insertion of a new connection requires more work when there are more connections in the connection tracking table.

It is important to note that the CPU utilization measurements were done separately after finishing the maximum connection establishment rate and throughput measurements, as the execution of the used **`dstat`** command also consumed CPU power. The CPU utilization of each DUT was measured separately and the output of the Tester program produced during CPU utilization measurement was discarded.

5.4. Scalability against the number of connections

It was examined how the performance of the 464XLAT system depends on the number of connections in the connection tracking table of the PLAT. To perform the measurements with several different numbers of connections, the source port number range was always 1–4000 and the destination port number range was first 1–250 and its higher limit was doubled 8 times with a final range of 1–64,000. Therefore, the number of connections during the nine consecutive measurement series was increased from 1,000,000 to 256,000,000. The filling up of the connection tracking table with the high number of connections took a significant amount of time, thus the timeout was modified from the default value (300 s) to a value higher than the duration of the entire

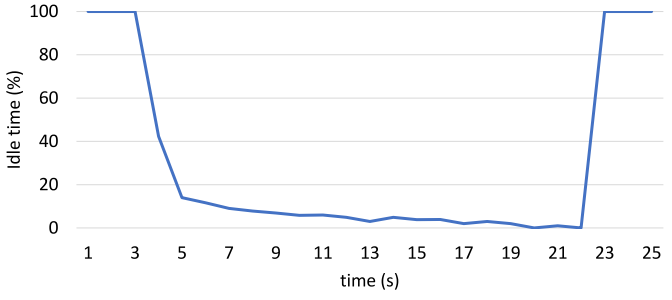


Fig. 15. CPU idle time of the PLAT using 1 CPU core and maximum frame rate of the maximum connection establishment rate measurement.

experiment (test phase 1, gap between the two phases, test phase 2). The number of active CPU cores was always 16.

The results are disclosed in Table 5. Both maximum connection establishment rate and throughput show continuous degradation with an increase in the number of connections. Due to the increase in the number of connections from 1 million to 256 million, the maximum connection establishment rate decreased from 576,790 cps to 272,061 cps, whereas the throughput decreased from 611,141 fps to 279,051 fps. The good news is that there was no sudden drop in the performance of the system at any working point, thus it complies with the *graceful degradation* principle. The bad news is that both performance characteristics decreased to less than half.

6. Scalability of the Jool Implementation of MAP-T

6.1. Original measurement setup and tests

Building a MAP-T system is a much more complex task than building a 464XLAT system. It requires a preliminary design, which includes address calculations. The used design was inspired by the example from the MAP-T setup documentation of Jool [34], which can be recommended as an easy-to-follow introduction to MAP-T for those readers not familiar with MAP-T. It could not be fully followed because they used port sets with 2048 source port numbers, but the authors wanted to use at least 4000 source port numbers to achieve 256 million connections when using 64,000 destination port numbers. Finally, 8192 size port sets were chosen because the authors experienced serious performance degradation when they used 4000 source port numbers by nearly exhausting the 4096-size port set during the preliminary tests. The 8192 large port set size allowed 8 users to share a single IPv4 address and 3 bits were needed to identify a port set. The 192.0.2.0/24 public IPv4 address range was chosen for the measurements, thus the *IPv4 suffix* was 8 bits. Hence, the number of *EA* (Embedded Address) *bits* was $8 + 3 = 11$. From among the possible $2^{11} = 2048$ CEs, the authors chose the one that can be identified by the binary number of "**00100011***011*" (this number will be later referred to as "11b" hexadecimal number). The first 8 bits (**00100011**) are the IPv4 suffix (**35** decimal) and the last 3 bits (*011*) are the *PSID* (Port Set ID, *3* decimal). Thus, the port range is *24,576–32,767*. (This means that the subscriber could use 192.0.2.**35**:*24,576–32,767*.) For an easy calculation, the authors chose 2001:db8:ce/53 as the *IPv6 prefix* for the *Basic Mapping Rule* (BMR), hence the BMR will be as follows: *IPv6 prefix*: 2001:db8:ce/53; *IPv4 prefix*: 192.0.2.0/24; and *number of EA bits*: 11. The *Default Mapping Rule* (DMR) is 64:ff9b/96.

The topology of the measurement system is shown in Fig. 16. However, this is only the high-level topology of the system. The implementation of the CE device required the usage of namespaces for technical reasons specific to the Linux kernel and its Netfilter framework. In short, as described in Section 2.1.5, the stateful NAT44 translation should happen before the stateless NAT46 translation, but the Netfilter framework does stateful NAT44 POSTROUTING and thus the order of the two translations would be incorrect. The required order can be ensured by using two namespaces, as elaborated in Ref. [34]. The internal topology of the CE device is shown in Fig. 17. Here, **`enp5s0f0`** and **`enp5s0f1`** are the physical network interfaces of the p106 server, whereas **`to_global`** and **`to_napt`** are virtual Ethernet interfaces for interconnecting the "napt" namespace and global namespace. The *Network Address and Port Translation* (NAPT), also called *stateful NAT44*, function is implemented by using **`iptables`**.

The detailed settings of the test system are disclosed in Appendix A.1 to support the reproducibility of the measurements. However, one important aspect needs to be mentioned here: the interrupts occur in the "napt" namespace because the packet arrivals were directed to the 16–31 CPU cores using a different method than **`ethtool`**.

Table 5
Performance of the Jool implementation of 464XLAT as a function of the number of connections, 16 CPU cores.

Number of connections (million)	1	2	4	8	16	32	64	128	256
Max. conn. est. rate - Median (cps)	**576,790**	**514,566**	**459,089**	**425,400**	**394,229**	**357,846**	**327,910**	**290,325**	**272,061**
Max. conn. est. rate - Minimum (cps)	549,999	506,233	451,170	419,749	388,181	351,561	324,217	286,861	261,868
Max. conn. est. rate - Maximum(cps)	593,751	519,598	468,751	432,617	398,804	369,690	337,524	303,223	276,563
Throughput (bidir.) - Median (fps)	**611,141**	**554,602**	**495,614**	**453,672**	**421,345**	**384,374**	**340,680**	**304,510**	**279,051**
Throughput (bidir.) - Minimum (fps)	596,846	547,664	487,106	448,238	412,498	379,352	336,758	299,998	276,550
Throughput (bidir.) - Maximum(fps)	615,892	562,502	500,398	459,412	426,270	387,892	347,992	307,104	281,306

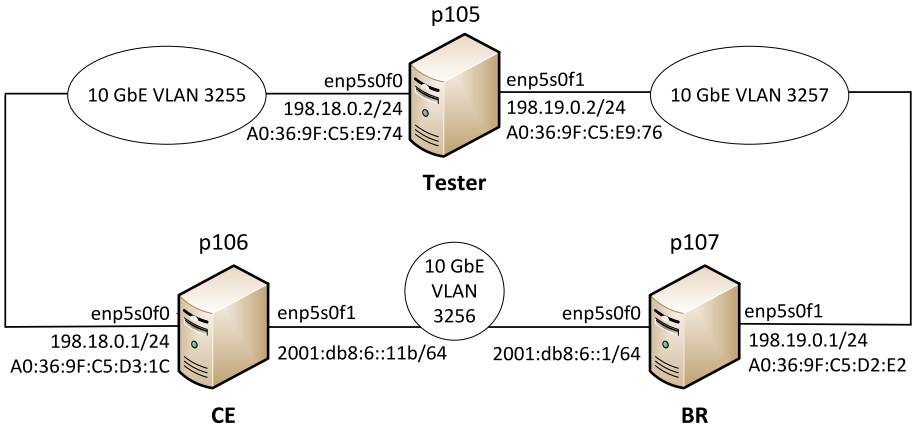


Fig. 16. Original test setup for scalability measurements of the Jool implementation of MAP-T.

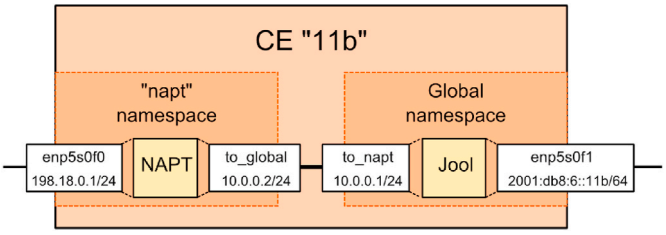


Fig. 17. Namespaces used for implementing MAP-T CE (based on [34]).

6.2. Scalability against the number of CPU cores

The same tests (using the same parameters) were performed as with 464XLAT.

The maximum connection establishment rate and throughput measurement results of the Jool implementation of MAP-T are presented in Table 6. The scale-up of the two quantities was different. Maximum connection establishment rate exhibited a relatively good scale-up from 1 to 4 cores (from 443,388 cps to 1,054,225 cps), where it reached its maximum. (The median shows a slight decrease at 8 cores [to 1,050,862 cps] but it was within measurement error.) Throughput showed a relatively good scale-up from 1 to 8 cores (from 434,220 fps to 2,081,068 fps) and there was only a marginal (less than 4 %) increase at 16 cores (2,162,170 fps).

It should be recalled that only CE is stateful from among the two devices used to implement MAP-T, and BR is stateless. Additionally, the maximum connection establishment rate was a new metric that was introduced in the methodology for benchmarking stateful NATxy gateways [9]. Thus, it was visible that the CE device at 8 and 16 cores limited the maximum connection establishment rate performance of the system. Additionally, it is possible that the CE device at 16 cores also limited the throughput of the system, but currently, this has not been confirmed. The authors were more interested in the scalability of the BR device as stated in Section 3.2.3. Therefore, the measurements were repeated using two separate servers to implement the two sub-functions of the CE device. However, the original setup was not abandoned to have a basis for comparison.

6.3. Scalability against the number of connections

The same tests (using the same parameters) were performed as with 464XLAT.

The results are presented in Table 7. The maximum connection establishment rate results showed two anomalies.

- The median values exhibited a significant increase when the number of connections was raised from 1 million to 8 million.

Table 6
Performance of the Jool implementation of MAP-T as a function of the active CPU cores (original setup), 4 M connections.

Number of active CPU cores	1	2	4	8	16
Maximum connection establishment rate - Median (cps)	**443,388**	**693,165**	**1,054,225**	**1,050,862**	**1,053,726**
Maximum connection establishment rate - Minimum (cps)	439,452	681,249	1,007,805	1,023,418	999,022
Maximum connection establishment rate - Maximum(cps)	445,328	694,922	1,125,001	1,125,065	1,125,123
Maximum connection establishment rate - Relative scale up	1	0.782	0.594	0.296	0.149
Throughput (bidirectional traffic) - Median (fps)	**434,220**	**750,820**	**1,345,462**	**2,081,068**	**2,162,170**
Throughput (bidirectional traffic) - Minimum (fps)	432,806	674,998	1,342,754	2,078,074	2,158,122
Throughput (bidirectional traffic) - Maximum(fps)	434,984	751,336	1,347,658	2,082,098	2,179,988
Throughput (bidirectional traffic) - Relative scale up	1	0.865	0.775	0.599	0.311

Table 7
Performance of the Jool implementation of MAP-T as a function of the number of connections (original setup), 16 CPU cores.

Number of connections (million)	1	2	4	8	16	32	64	128	256
Max. conn. est. r. - Median (cps)	**546,596**	**750,703**	**1,053,726**	**1,989,793**	**1,990,010**	**1,992,191**	**1,992,217**	**1,976,530**	**1,600,486**
Max. conn. est. r. – Min. (cps)	526,609	718,473	999,022	1,499,999	1,312,486	1,874,999	1,499,540	1,749,997	1,546,862
Max. conn. est. r. – Max.(cps)	563,202	782,257	1,125,123	1,996,950	1,993,934	1,994,029	1,998,047	1,994,018	1,620,494
Throughput (bidir.) - Median (fps)	**2,175,004**	**2,173,340**	**2,162,170**	**2,172,770**	**2,172,930**	**2,174,026**	**2,173,252**	**2,171,682**	**2,171,506**
Throughput (bidir.) – Min. (fps)	2,171,778	2,170,886	2,158,122	2,168,942	2,171,872	2,171,626	2,170,996	2,156,248	2,169,368
Throughput (bidir.) – Max.(fps)	2,177,866	2,175,782	2,179,988	2,174,174	2,175,782	2,176,086	2,174,536	2,173,586	2,173,372

- The difference between the minimum and maximum values was extremely high at a certain number of connections (the situation was the worst at 16 million connections), thus the results were rather unreliable.

By studying the measurement log files, it was found that both anomalies had a common root cause, which was that significant frame loss happened during test phase 1. This was in the order of magnitude of 0.01 %. Thus, the increase in the number of connections allowed more frames to be lost while the given step of the binary search was still considered successful. Although the authors could produce "better looking" results using a laxer acceptance criterion (e.g., 99.9 %, which allowed a 0.1 % loss), they did not do so because the maximum connection establishment rate characterizes the CE device and they focused on the scalability of the BR device.

As for the throughput, it showed practically no decrease when the number of connections was raised from 1 million to 256 million. This is an excellent scalability.

6.4. Modified measurement setup and tests

To resolve the potential problem that the insufficient performance of the CE device could be the bottleneck, separate servers were used to implement its two sub-functions. Hence, the usage of namespaces and virtual Ethernet devices was also eliminated.

The topology of the modified measurement system is disclosed in Fig. 18. As for the actual implementation, nearly the same commands were used as for the original measurement setup. However, the namespaces were omitted and the physical interface names were always used. In addition to this, the appropriate `ethtool` command was always used to distribute the interrupts among the CPU cores of the servers.

The same tests (using the same parameters) were performed as with the original setup.

6.5. Scalability against the number of CPU cores

The maximum connection establishment rate and throughput measurement results of the Jool implementation of MAP-T measured with the modified measurement setup are presented in Table 8. The results are fundamentally quite similar to the results using the original setup.

- The maximum connection establishment rate scaled up from 1 to 4 cores, as before. (The results were somewhat higher for all numbers of CPU cores.)
- The throughput scaled up quite well from 1 to 8 cores as before (from 447,262 fps to 1,996,529 fps) and there was only a marginal (less than 3 %) increase at 16 cores (2,051,404 fps).
- Interestingly, the throughput results of the modified setup were somewhat higher than that of the original setup at a single core (as expected), but they were somewhat lower at 2–16 cores (which was unexpected, but the highest difference at 16 cores was only about 5 %).

Unfortunately, the improvement of the scale-up of the modified test system did not meet the expectations of the authors. In the search for root causes of this phenomenon, the following issues were identified.

- When the RSS of the original MAP-T test system was configured, the authors could not handle the distribution of the interrupts caused by packet arrivals to the virtual Ethernet interfaces among the CPU cores using `ethtool`. Furthermore, the solution that was used facilitated the assignment of the interrupts to the 16–31 CPU cores. Thus, the CE of the original test system used all 32 cores of a single server, whereas the two sub-functions of the CE of the modified test system used the first 16 CPU cores of the servers. The moderate increase in the maximum connection establishment rate of the modified test system can be attributed to the fact that the usage of the namespaces and virtual Ethernet interfaces was eliminated.
- The most annoying difference was that the throughput of the modified test system was somewhat lower than that of the original system. This difference could be attributed to the phenomenon that the authors had already experienced during DNS server testing; using different instances of theoretically identical test systems (built up by Dell PowerEdge C6220 servers with the same hardware and software configuration), somewhat different results were produced. For example, the authoritative DNS performance of *Yet Another DNS Implementation For All* (YADIFA) was 180,140qps or 163,641qps, please refer to Table 1 of [28]. To mitigate the problem, the same

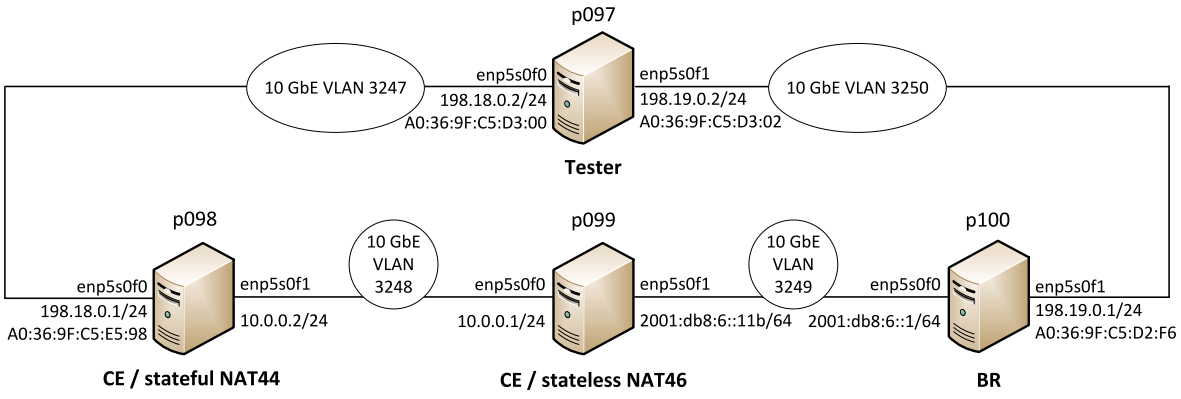


Fig. 18. Modified test setup for scalability measurements of the Jool implementation of MAP-T.

Table 8
Performance of the Jool implementation of MAP-T as a function of the active CPU cores (modified setup), 4 M connections.

Number of active CPU cores	1	2	4	8	16
Maximum connection establishment rate - Median (fps)	**458,582**	**767,221**	**1,187,629**	**1,198,302**	**1,150,865**
Maximum connection establishment rate - Minimum (fps)	451,170	749,999	1,049,999	1,114,940	1,099,217
Maximum connection establishment rate - Maximum (fps)	459,526	768,770	1,235,938	1,246,961	1,225,091
Maximum connection establishment rate - Relative scale up	1	0.837	0.647	0.327	0.157
Throughput (bidirectional traffic) - Median (fps)	**447,262**	**728,366**	**1,303,495**	**1,996,529**	**2,051,404**
Throughput (bidirectional traffic) - Minimum (fps)	445,310	720,700	1,301,950	1,992,606	2,050,654
Throughput (bidirectional traffic) - Maximum (fps)	448,438	730,502	1,305,470	1,998,010	2,052,198
Throughput (bidirectional traffic) - Relative scale up	1	0.814	0.729	0.558	0.287

computers were used for the performance comparison of various DNS64 server and authoritative DNS server implementations in Ref. [28] and Ref. [29], respectively.

The most important achievement of the modified test setup for benchmarking MAP-T was that the authors managed to eliminate the namespaces, which eased CPU utilization measurements and was crucial for checking the bottleneck.

6.6. Checking the bottleneck

As with 464XLAT, the CPU utilization of the servers was measured to point out the bottleneck. The CPU idle time of the BR device as a function of time using 1 CPU core and the measured maximum frame rate of the throughput measurement (448,438 fps) is shown in Fig. 19. The graph is ideal as it shows that the single CPU core was fully utilized during the throughput measurement.

The real question that needs to be asked is what happens when the BR has 16 active cores. The CPU idle time of the CE1 (performing stateful NAT44), CE2 (performing stateless NAT64), and BR devices as a function of time using 16 CPU cores and the measured maximum frame rate of the throughput measurement (2,052,198 fps) is shown in Fig. 20, Fig. 21, and Fig. 22, respectively. The fact that the idle time of all the CPU cores of the BR was 0 % proves that the BR could surely not process more packets.

Of course, the question remains open of why the addition of 8 cores to the 8-core BR results in only a 3 % performance increase. The investigation of this question exceeds the scope of the current paper but could be very useful for the developers of Jool.

6.7. Scalability against the number of connections

The maximum connection establishment rate and throughput measurement results of the Jool implementation of MAP-T measured with the modified measurement setup are shown in Table 9. The results are fundamentally quite similar to the results using the original setup. The most important differences are.

- The maximum connection rate did not show a drop at 256 million connections.

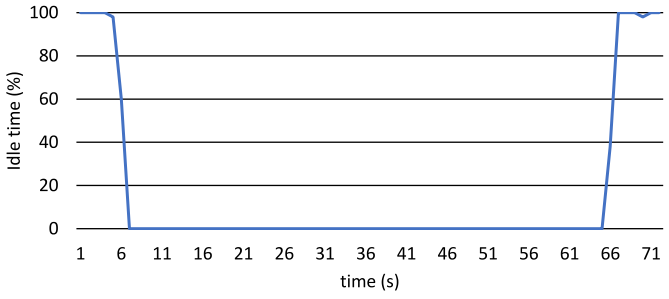


Fig. 19. CPU idle time of the BR using 1 CPU core and maximum frame rate of the throughput measurement.

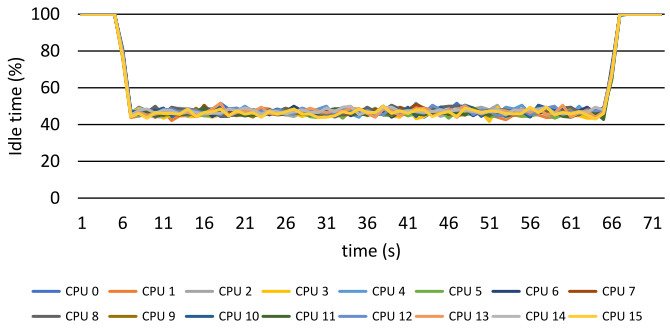


Fig. 20. CPU idle time of the CE1 (stateful NAT44) using 16 CPU cores and maximum frame rate of the throughput measurement.

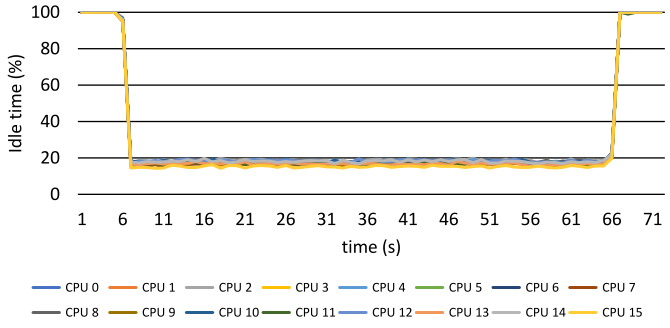


Fig. 21. CPU idle time of the CE2 (stateless NAT46) using 16 CPU cores and maximum frame rate of the throughput measurement.

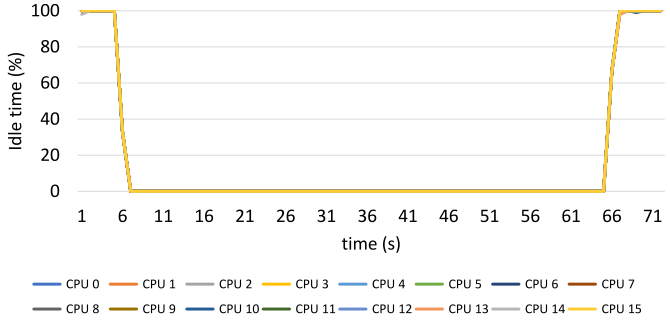


Fig. 22. CPU idle time of the BR using 16 CPU cores and maximum frame rate of the throughput measurement.. (The line expressing the utilization of CPU core 15 hides the other lines, as all 16 are nearly identical.)

Table 9
Performance of the Jool implementation of MAP-T as a function of the number of connections (modified setup), 16 CPU cores.

Number of connections (million)	1	2	4	8	16	32	64	128	256
Max. conn. est. r. - Median (fps)	**534,680**	**754,826**	**1,150,865**	**1,822,311**	**2,004,725**	**2,005,831**	**2,005,818**	**2,006,344**	**2,006,614**
Max. conn. est. r. – Min. (fps)	496,874	687,499	1,099,217	1,756,300	2,003,669	2,004,908	2,005,003	2,005,735	2,006,458
Max. conn. est. r. – Max.(fps)	575,002	812,501	1,225,091	1,876,489	2,005,737	2,006,841	2,006,491	2,006,881	2,007,568
Throughput (bidir.) - Median (fps)	**2,050,620**	**2,051,260**	**2,051,404**	**2,051,378**	**2,051,602**	**2,052,048**	**2,051,474**	**2,051,636**	**2,051,774**
Throughput (bidir.) – Min. (fps)	2,049,000	2,049,654	2,050,654	2,050,236	2,050,916	2,051,360	2,039,060	2,051,336	2,051,198
Throughput (bidir.) – Max.(fps)	2,051,722	2,052,198	2,052,198	2,052,246	2,052,466	2,052,540	2,052,246	2,052,258	2,052,466

- As expected, based on the CPU scale-up results, the throughput values were 5–6 % lower than in the case of the original test system.

7. Discussion and future research

7.1. Achievements

By performing the detailed scalability analysis of two different IPv4aaS technologies (464XLAT and MAP-T), the operation and practical usability of the novel method the authors proposed in Section 2 were successfully demonstrated.

It should be noted that the method does not build on any technology-specific properties. It handles the aggregate of CE and PE devices as a stateful NAT44 gateway, which is true in the case of all five IPv4aaS technologies. Therefore, the method can be used with any of them.

There are two important advantages of the proposed method.

1. It is RFC 8219 compliant and thus it can support all the required measurements providing their full benefits.
2. It works according to the Dual DUT setup; therefore, it does not need a tester specific to the examined IPv4aaS technology, but a stateful NAT44 tester can be used to investigate any of them.

In Section 7.2 it was found (and also confirmed in Section 7.5) that the MAP-T test system showed a significantly different scale-up regarding the maximum connection establishment rate (good scale-up from 1 to 4 cores) compared to the throughput (good scale-up from 1 to 8 cores). This difference justifies the approach of the Internet-Draft [9] to distinguish maximum connection establishment rate and throughput.

In this paper, two performance metrics were used: the maximum connection establishment rate, which is a new, stateful specific metric, and the throughput, which is one of the "classic" metrics of RFC 8219. In Ref. [22], the proposed measurement method for stateful NATxy gateways was validated by performing tests with three radically different stateful NAT64 implementations. It was demonstrated that the further "classic" tests of RFC 8219, e.g., latency, frame loss rate, packet delay variation (PDV), etc. can also be executed in test phase 2. As the aggregate of CE and PE devices is a stateful NAT44 gateway, it also applies to any of the five IPv4aaS technologies in which the above-mentioned measurements can be executed (Please refer to section 7.2 for the limitations regarding their results.).

7.2. Limitations of the method

The Dual DUT setup also has its limitations in that the performance of CE and PE devices of the IPv4aaS technologies are measured together. It has multiple consequences including the following.

1. Additional work is needed to find the bottleneck. To that end, a possible solution based on CPU utilization measurement was also provided.
2. If the bottleneck is not the PE device and one would like to make it the bottleneck, as its scalability is usually the focal point of the research, then an extra attempt is needed to make the PE device the bottleneck. For example, a more powerful CE device or multiple CE devices may be used.
3. The results characterize the aggregate of CE and PE devices, and the metrics for individual CE or PE devices are not trivial to derive.

As for the third limitation, for example, when the latency is measured, the user does not know in what proportion the latency result should be divided between CE and the PE devices, as there is significant asymmetry in them. However, the latency of the aggregate of CE and the PE devices can serve as an upper bound for the latency of the individual devices. The same applies to the frame loss rate, too.

Regarding the equivalence and difference between the dual DUT setup and the single DUT setup of RFC 8219 in the case of stateless, identical devices, theoretical considerations were made and measurements were performed in the case of IPv6 routers and SIIT gateways in Ref. [35].

7.3. Potential alternative solutions

In section 2, the problem of benchmarking the five IPv4aaS technologies according to the Dual DUT setup of RFC 8219 was simplified into the problem of benchmarking a stateful NAT44 gateway according to the single DUT setup of RFC 8219. To solve this simpler problem, the usage of the benchmarking methodology proposed in Internet-Draft [9] was recommended. As far as the authors know, it is the only RFC 8219-compliant solution for the problem. However, there are other solutions for benchmarking a stateful NAT44 gateway. In Ref. [36], it was surveyed how other researchers measured the performance of the **`iptables`** stateful NAT44 solution. In Section 3.2.2 of [36], it is written that "researchers were creative enough to accommodate to the limitation of NAT44 that connections may be initiated only from the private side. They put the iperf or D-ITG server on the public side and thus the measurement was feasible." Of course, these solutions are applicable in the case when IPv4aaS technologies are benchmarked. Moreover, the results can be used to compare their performance. However, using these solutions will result in losing the benefits of the standard RFC 8219 compliant measurements as well as the benefits of the Internet-Draft [9], e.g., distinguishing maximum connection establishment rate and throughput.

7.4. Comparison with other methods

The support for all the "classic" tests of RFC 8219 is the most important distinguishing factor of the proposed method, but it is not the only one.

Similar to its predecessors, RFC 8219 requires testing with bidirectional traffic, however, it also adds optional testing with unidirectional traffic. In the case of the IPv4aaS technologies, the traffic volume in the "download" direction is usually significantly higher than the traffic in the "upload" direction. As demonstrated in Ref. [22], the proposed method can be used with unidirectional traffic in any of the two directions. The alternative solutions mentioned in Sections 3.1, 3.2, and 7.3 cannot use unidirectional traffic in the download direction, because that traffic comes from a "reflector" device that sends back the received packets (or sends answers to the received DNS queries, as elaborated in

Section 3.2).

Another distinguishing factor of the proposed method is that it satisfies the requirement of RFC 4814 for pseudorandom port numbers and also provides a clear way for scalability testing regarding the number of connections (often called network flows).

7.5. Other aspects than performance

This paper has focused on performance and scalability, however, when network operators decide on the selection of the most suitable IPv4aaS solution for their purposes, several other factors need to be considered including security, reliability, documentation, support, experience with the software or hardware solution, references of the vendor, hardware requirement (if a software implementation is used), and price (if a free software solution is not used).

As for the security analysis of the IPv4aaS solutions, there are numerous recent publications of Al-Azzawi covering the security analysis of 464XLAT [37], DS-Lite [38], and Lw4o6 [39] from both theoretical and practical aspects.

D'yab has conducted a comprehensive survey of the IPv4aasS technologies including their implementations [40].

7.6. Plans for future research

In this paper, **siitperf** was used for *stateful NAT44* measurements, but it can do *SIIT* (*stateless NAT46* or *stateless NAT64*) and *stateful NAT64* measurements too. Thus, it can be used to benchmark the two subcomponents of 464XLAT (CLAT and PLAT) separately following the Single DUT setup of RFC 8219. As for the SIIT performance of Jool, the first author has already performed some tests with another measurement tool called **nat64tester**; however, it did not support RFC 4814 pseudorandom port numbers. Instead, it always sent the very same test frames, thus only two CPU cores could be used to process the interrupts of packet arrivals (one for each direction). Therefore, it could not be used for measuring the scalability of the SIIT performance of Jool with the number of active CPU cores [30]. The authors are currently working on it using **siitperf**. The scalability of the stateful NAT64 performance of Jool was already benchmarked to validate the methodology for benchmarking stateful NAT64 gateways [22].

If the two sub-functions of the MAP-T CE are implemented by two separate devices (stateful NAT44 and stateless NAT46) then they can be benchmarked one by one following the Single DUT setup of RFC 8219 and using **siitperf** as the measurement tool.

At the time of performing the above measurements, the authors did not have a MAP-T BR tester, but its implementation was in progress [41]. Since then, Al-hamadani has finished the implementation of **maptperf**, the World's first free software RFC 8219 compliant MAP-T BR tester. It is available under the GPLv3 license from GitHub [42] and it is documented in Ref. [43]. The benchmarking of the scalability of the Jool implementation of MAP-T BR using **maptperf** is already in progress.

The research plans of the team of the first author include the extension of **maptperf** for benchmarking MAP-E BR, as well as the implementation of a Tester for benchmarking the lwAFTR component of Lw4o6.

8. Conclusion

An RFC 8219-compliant benchmarking method was proposed for the performance and scalability analysis of the five most important IPv4aaS technologies. The method works according to the dual DUT setup of RFC 8219 and it uses a stateful NAT44 tester and not a technology-specific tester.

The proposed method was validated and its operation was demonstrated in the example of the Jool implementation of the 464XLAT and MAP-T IPv4aaS solutions.

464XLAT showed a moderate scale-up with the number of CPU cores from one to four cores; the maximum connection establishment rate increased from 210,344 cps to 423,903 cps (relative scale-up: 0.504), whereas the throughput increased from 217,946 fps to 472,135 fps (relative scale up: 0.542). Above 4 CPU cores, the addition of further active CPU cores did not result in a significant increase in the performance of the system. Due to the increase in the number of connections from 1 million to 256 million, the maximum connection establishment rate decreased from 576,790 cps to 272,061 cps, whereas the throughput decreased from 611,141 fps to 279,051 fps. It was checked that these numbers characterize the PLAT part of the system.

As for MAP-T, the scale-up of the maximum connection establishment rate and throughput was different. Maximum connection establishment rate exhibited a relatively good scale-up from 1 to 4 cores (from 443,388 cps to 1,054,225 cps), where it reached its maximum. Throughput showed a relatively good scale-up from 1 to 8 cores (from 434,220 fps to 2,081,068 fps) and there was only a marginal (less than 4 %) increase at 16 cores (2,162,170 fps). However, the maximum connection establishment rate characterizes the CE device, and not the BR device, which is the focal point of the scalability of the system. As for the scalability of the MAP-T system against the number of connections, the throughput exhibited a constant high performance in the entire range, more than 2 million fps, which means both an excellent performance and an excellent scalability.

All in all, it was found that the Jool implementation of MAP-T scaled up much better than that of 464XLAT.

CRediT authorship contribution statement

Gábor Lencse: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Conceptualization. **Ádám Bazsó:** Writing – review & editing, Validation, Investigation, Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgement

The experiments were carried out remotely using the resources of NICT StarBED, 2–12 Asahidai, Nomi-City, Ishikawa 923–1211, Japan.

The authors thank Shuuhei Takimoto for the possibility of using StarBED as well as Satoru Gonno, Makoto Yoshida, and Tsukasa Nishita for their help and advice on StarBED usage related issues.

The authors thank Bertalan Kovács, Sándor Répás, Omar D'yab and Norbert Nagy for reviewing and commenting on the manuscript.

The authors thank John Kowalchuk, Széchenyi István University, for the English language proofreading of the manuscript.

Appendix

A.1. Setup of the Original MAP-T Test System

The CE and BR devices were configured using the commands shown in Figs. 23 and 24, respectively. The interrupts, caused by packet arrivals on the physical interfaces, were distributed among CPU cores 0–15 similar to before, but virtual Ethernet interfaces required a different method. The commands are shown in Fig. 25. Using this method, the interrupts of the packets arriving at the **`to_global`** and **`to_napt`** virtual Ethernet interfaces were processed by CPU cores 16–23 and 24–31, respectively.

```
ip link add to_napt type veth peer name to_global
ip link set enp5s0f0 netns napt
ip link set to_global netns napt
ip netns exec napt ip link set to_global up
ip netns exec napt ip link set enp5s0f0 up
ip link set to_napt up
ip addr add 10.0.0.1/24 dev to_napt
ip netns exec napt ip addr add 10.0.0.2/24 dev to_global
ip netns exec napt ip addr add 198.18.0.1/24 dev enp5s0f0
ip link set enp5s0f1 up
ip addr add 2001:db8:6::11b/64 dev enp5s0f1
ip netns exec napt ip route add default via 10.0.0.1
ip route add 64:ff9b::/96 via 2001:db8:6::1
ip route add 192.0.2.35/32 via 10.0.0.2
ip netns exec napt iptables -t nat -A POSTROUTING -s 198.18.0.0/24 -o to_global -p udp -j SNAT \
  --to-source 192.0.2.35:24576-32767
ip netns exec napt sysctl -w net.ipv4.conf.all.forwarding=1
ip netns exec napt sysctl -w net.ipv6.conf.all.forwarding=1
sysctl -w net.ipv4.conf.all.forwarding=1
sysctl -w net.ipv6.conf.all.forwarding=1
modprobe jool; modprobe jool_mapt
jool_mapt instance add "CE 11b" --netfilter --dmr 64:ff9b::/96
jool_mapt -i "CE 11b" global update end-user-ipv6-prefix 2001:db8:ce:11b::/64
jool_mapt -i "CE 11b" global update bmr 2001:db8:ce::/53 192.0.2.0/24 11 0
jool_mapt -i "CE 11b" global update map-t-type CE
ip netns exec napt ip neighbor add 198.18.0.2 lladdr a0:36:9f:c5:e9:74 dev enp5s0f0 nud permanent
```

Fig. 23. Commands used for configuring the CE device of the original MAP-T test system.

```
modprobe jool; modprobe jool_mapt
jool_mapt instance add "BR" --netfilter --dmr 64:ff9b::/96
jool_mapt -i "BR" fmrt add 2001:db8:ce::/53 192.0.2.0/24 11 0
jool_mapt -i "BR" global update map-t-type BR
ip route add 2001:db8:ce:11b::/64 via 2001:db8:6::11b
ip neighbor add 198.19.0.2 lladdr a0:36:9f:c5:e9:76 dev enp5s0f1 nud permanent
```

Fig. 24. Commands used for configuring the BR device of the original MAP-T test system.

```
ip netns exec napt ethtool -N enp5s0f0 rx-flow-hash udp4 sdfn
ethtool -N enp5s0f1 rx-flow-hash udp6 sdfn
echo "00ff0000" > /sys/class/net/to_global/queues/rx-0/rps_cpus
echo "ff000000" > /sys/class/net/to_napt/queues/rx-0/rps_cpus
```

Fig. 25. Commands used for distributing the interrupts caused by packet arrivals in the case of the original MAP-T test system.

References

[1] NRO, “Free pool of IPv4 address space depleted”, [Online]. Available: https://www.nro.net/ipv4-free-pool-depleted/.

[2] J. Palet Martinez, H.M.-H. Liu, M. Kawashima, Requirements for IPv6 customer edge routers to support IPv4-as-a-Service, IETF RFC 8585 (May 2019), https://doi.org/10.17487/RFC8585.

[3] G. Lencse, J. Palet Martinez, L. Howard, R. Patterson, I. Farrer, Pros and cons of IPv6 transition technologies for IPv4-as-a-Service (IPv4aaS), IETF RFC 9313 (Oct 2022), https://doi.org/10.17487/RFC9313.

[4] A.T.H Al-hamadani, G. Lencse, A survey on the performance analysis of IPv6 transition technologies, Acta Technica Jaurinensis 14 (2) (2021) 186–211, https://doi.org/10.14513/actatechjaur.00577.

[5] M. Georgescu, H. Hazeyama, Y. Kadobayashi, S. Yamaguchi, Empirical analysis of IPv6 transition technologies using the IPv6 network evaluation testbed, EAI Endorsed Transactions on Industrial Networks and Intelligent Systems 2 (2) (2015), https://doi.org/10.4108/inis.2.2.e1.

[6] G. Lencse, N. Nagy, Towards the scalability comparison of the Jool implementation of the 464XLAT and of the MAP-T IPv4aaS technologies, Int. J. Commun. Syst. 35 (18) (2022), https://doi.org/10.1002/dac.5354. Paper ID: e5354.

[7] M. Georgescu, L. Pislaru, G. Lencse, Benchmarking methodology for IPv6 transition technologies, IETF RFC 8219 (Aug. 2017), https://doi.org/10.17487/RFC8219.

[8] S. Bradner, J. McQuaid, Benchmarking methodology for network interconnect devices, IETF RFC 2544 (Mar. 1999), https://doi.org/10.17487/RFC2544.

[9] G. Lencse, K. Shima, “Benchmarking Methodology for Stateful NATxy Gateways Using RFC 4814 Pseudorandom Port Numbers”, Internet-Draft, Jan. 23, 2024. https://datatracker.ietf.org/doc/html/draft-ietf-bmwg-benchmarking-stateful.

[10] M. Mawatri, M. Kawashima, C. Byrne, “464XLAT: combination of stateful and stateless translation”, IETF RFC 6877 (Apr. 2013) https://doi.org/10.17487/RFC6877.

[11] C. Bao, X. Li, F. Baker, T. Anderson, F. Gont, IP/ICMP Translation Algorithm” *IETF RFC 7915*, Jun. 2016, https://doi.org/10.17487/RFC7915.

[12] M. Bagnulo, P. Matthews, I. Beijnum, Stateful NAT64: network address and protocol translation from IPv6 clients to IPv4 servers, IETF RFC 6146 (Apr. 2011), https://doi.org/10.17487/RFC6146.

[13] A. Durand, R. Droms, J. Woodyatt, Y. Lee, Dual-stack lite broadband deployments following IPv4 exhaustion, IETF RFC 6333 (Aug) (2011), https://doi.org/10.17487/RFC6333.

[14] Y. Cui, Q. Sun, M. Boucadair, T. Tsou, Y. Lee, I. Farrer, Lightweight 4over6: an extension to the dual-stack lite architecture, IETF RFC 7596 (Jul. 2015), https://doi.org/10.17487/RFC7596.

[15] The address plus port (A+P) approach to the IPv4 address shortage, in: R. Bush (Ed.), IETF RFC 6346 (Aug. 2011), https://doi.org/10.17487/RFC6346.

[16] Mapping of address and port with encapsulation (MAP-E), in: O. Troan, W. Dec, X. Li, C. Bao, S. Matsushima, T. Murakami, T. Taylor (Eds.), IETF RFC 7597 (July 2015), https://doi.org/10.17487/RFC7597.

[17] X. Li, C. Bao, W. Dec, O. Troan, S. Matsushima, T. Murakami (Eds.), Mapping of Address and Port Using Translation (MAP-T), vol. 7599, IETF RFC, July 2015, https://doi.org/10.17487/RFC7599.

[18] D. Newman, T. Player, Hash and stuffing: overlooked factors in network device benchmarking, IETF RFC 4814 (2008), https://doi.org/10.17487/RFC4814.
[19] C. Popoviciu, A. Hamza, G.V. de Velde, D. Dugatkin, IPv6 benchmarking methodology for network interconnect Devices, IETF RFC 5180 (2008), https://doi.org/10.17487/RFC5180.
[20] G. Lencse, Y. Kadobayashi, Comprehensive survey of IPv6 transition technologies: a subjective classification for security analysis, IEICE Trans. Commun. E102-B (10) (2019) 2021–2035, https://doi.org/10.1587/transcom.2018EBR0002.
[21] G. Lencse, Design and implementation of a software tester for benchmarking stateful NATxy gateways: theory and practice of extending siitperf for stateful tests, Comput. Commun. 172 (1) (Aug. 2022) 75–88, https://doi.org/10.1016/j.comcom.2022.05.028.
[22] G. Lencse, K. Shima, K. Cho, Benchmarking methodology for stateful NAT64 gateways, Comput. Commun. 210 (1) (Oct. 2023) 256–272, https://doi.org/10.1016/j.comcom.2023.08.009.
[23] G. Lencse, "Scalability of IPv6 Transition Technologies for IPv4aaS", Internet-Draft, Oct. 2023. https://datatracker.ietf.org/doc/html/draft-lencse-v6ops-transition-scalability.
[24] M. Georgescu, H. Hazeyama, T. Okuda, Y. Kadobayashi, S. Yamaguchi, Benchmarking the load scalability of IPv6 transition technologies: a black-box analysis, in: Proc. 20th IEEE Symposium on Computers and Communication (ISCC), Larnaca, Cyprus, July 6-9, 2015, pp. 329–334, https://doi.org/10.1109/ISCC.2015.7405536.
[25] S. Avallone, S. Guadagno, D. Emma, A. Pescape, G. Ventre, D-ITG distributed Internet traffic generator, in: Proc. First International Conference on the Quantitative Evaluation of Systems (QEST 2004), 2004, https://doi.org/10.1109/QEST.2004.1348045. Enschede, Netherlands, September 27-30.
[26] G. Kunz, Parallel discrete event simulation, in: Modeling and Tools for Network Simulation, Springer-Verlag, Berlin, 2010, https://doi.org/10.1007/978-3-642-12331-3_8.
[27] G. Lencse, D. Bakai, Design and implementation of a test program for benchmarking DNS64 servers, IEICE Trans. Commun. E100-B (6) (Jun. 2017) 948–954, https://doi.org/10.1587/transcom.2016EBN0007.
[28] G. Lencse, Y. Kadobayashi, Benchmarking DNS64 implementations: theory and practice, Comput. Commun. 127 (1) (Sep. 2018) 61–74, https://doi.org/10.1016/j.comcom.2018.05.005.
[29] G. Lencse, Benchmarking authoritative DNS servers, IEEE Access 8 (Jul. 2020) 130224–130238, https://doi.org/10.1109/ACCESS.2020.3009141.
[30] G. Lencse, K. Shima, Performance analysis of SIIT implementations: testing and improving the methodology, Comput. Commun. 156 (Apr. 2020) 54–67, https://doi.org/10.1016/j.comcom.2020.03.034.
[31] T. Herbert, W. de Bruijn, "Scaling in the Linux networking stack". [Online]. Available: https://www.kernel.org/doc/Documentation/networking/scaling.txt.
[32] G. Lencse, Á. Kovács, K. Shima, Gaming with the throughput and the latency benchmarking measurement procedures of RFC 2544, International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems 9 (2) (2020) 10–17, https://doi.org/10.11601/ijates.v9i2.288.
[33] V. Gapon, "Tuning nf_conntrack", personal blog, [Online], available: https://ixnfo.com/en/tuning-nf_conntrack.html.
[34] NIC México, "MAP-T Run", MAP-T setup documentation of Jool, [Online], available: https://www.jool.mx/en/run-mapt.html.
[35] G. Lencse, Sz Szilágyi, "Equivalence and difference of the dual DUT setup and the single DUT setup of RFC 8219", under review in Int. J. Commun. Syst., review version is available: https://www.hit.bme.hu/~lencse/publications/IJCS-2023-Dual-DUT-for-review.pdf.
[36] G. Lencse, K. Shima, Optimizing the performance of the iptables stateful NAT44 solution, Infocommunications Journal 15 (1) (Mar. 2023) 55–63, https://doi.org/10.36244/ICJ.2023.1.6.
[37] A. Al-Azzawi, G. Lencse, Identification of the possible security issues of the 464XLAT IPv6 transition technology, Infocommunications Journal 13 (4) (December 2021) 10–18, https://doi.org/10.36244/ICJ.2021.4.2.
[38] A. Al-Azzawi, G. Lencse, Analysis of the security challenges facing the DS-Lite IPv6 transition technology, Electronics 12 (10) (May 2023), https://doi.org/10.3390/electronics12102335. Paper ID: 2335.
[39] A. Al-Azzawi, G. Lencse, Lightweight 4over6 test-bed for security analysis, Infocommunications Journal 15 (3) (Sep. 2023) 30–41, https://doi.org/10.36244/ICJ.2023.3.4.
[40] O. D'yab, A comprehensive survey on the most important IPv4aaS IPv6 transition technologies, their implementations and performance analysis, Infocommunications Journal 14 (3) (Sep. 2022) 35–44, https://doi.org/10.36244/ICJ.2022.3.5.
[41] A. Al-hamadani, G. Lencse, Towards implementing a software tester for benchmarking MAP-T devices, Infocommunications Journal 14 (3) (Sep. 2022) 45–54, https://doi.org/10.36244/ICJ.2022.3.6.
[42] A. Al-hamadani. "Maptperf: An RFC 8219 compliant MAP-T BR tester written in C++ using DPDK", https://github.com/alhamadani-ahmed/Maptperf_v1.3.
[43] A. Al-hamadani, G. Lencse "Maptperf: An RFC 8219 compliant tester for benchmarking MAP-T border relay routers", under review in *Computer Networks*, review version is available: https://www.hit.bme.hu/~lencse/publications/COMNET-2024-for-review.pdf.
[44] B. Jacob, S.W. Ng, D.T. Wang, Memory Systems: Cache, DRAM, Disk, Morgan Kaufmann Publishers, 2008, https://doi.org/10.1016/B978-0-12-379751-3.X5001-2.

Gábor Lencse received his MSc and PhD degrees in computer science from the Budapest University of Technology and Economics, Budapest, Hungary in 1994 and 2001, respectively.He has been working for the Department of Telecommunications, Széchenyi István University, Győr, Hungary since 1997. Now, he is a Professor. He has also been a part-time Senior Research Fellow at the Department of Networked Systems and Services, Budapest University of Technology and Economics since 2005. He is the head of the Cybersecurity and Network Technologies Research Group of the Faculty of Mechanical Engineering, Informatics and Electrical Engineering of the Széchenyi István University. His research interests include the performance and security analysis of IPv6 transition technologies. He is a co-author of RFC 8219 and RFC 9313.

Ádám Bazsó received his BSc in electrical engineering from Széchenyi István University, Győr, Hungary, 2022. He is a member of the Cybersecurity and Network Technologies Research Group of the Faculty of Mechanical Engineering, Informatics and Electrical Engineering of the Széchenyi István University.