Towards Implementing a Software Tester for Benchmarking MAP-T Devices

Ahmed Al-hamadani, and Gábor Lencse

***Abstract*—Several IPv6 transition technologies have been designed and developed over the past few years to accelerate the full adoption of the IPv6 address pool. To make things more organized, the Benchmarking Working Group of IETF has standardized a comprehensive benchmarking methodology for these technologies in its RFC 8219. The Mapping of Address and Port using Translation (MAP-T) is one of the most important transition technologies that belong to the double translation category in RFC 8219. This paper aims at presenting our progress towards implementing the world's first RFC 8219 compliant Tester for the MAP-T devices, more specifically, the MAP-T Customer Edge (CE) and the MAP-T Border Relay (BR). As part of the work of this paper, we presented a typical design for the Tester, followed by a discussion about the operational requirements, the scope of measurements, and some design considerations. Then, we installed a testbed for one of the MAP-T implementations, called Jool, and showed the results of the testbed. And finally, we ended up with a brief description of the MAP-T test program and its configuration parameters in case of testing the BR device.**

***Index Terms*—Benchmarking, Border Relay, Customer Edge, IPv6 transition technologies, MAP-T**

I. INTRODUCTION

THE public IPv4 address pool of IANA was depleted in 2011 [1]. However, the full deployment of IPv6 is taking too much time because it faces several significant challenges. As a softening solution to the problem, many transition technologies have been proposed and developed over the past few years to allow IPv4 and IPv6 to coexist and work with other for some time before totally excluding IPv4 from the Internet and fully adopting IPv6 [2]. The IETF's RFC 8219 [3] categorized these transition technologies into four groups, namely, dual stack, single translation, double translation, and encapsulation technologies, and it defines an organized comprehensive methodology for their benchmarking.

In dual stack [4], both IPv4 and IPv6 stacks are included in the network nodes, and the RFC 2544 [5] and RFC 5180 [6] compliant measurement tools can sufficiently benchmark dual stack devices.

A. Al-hamadani is with the Department of Networked Systems and Services, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary. (e-mail: alhamadani@hit.bme.hu).

G. Lencse is with the Department of Networked Systems and Services, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary. (e-mail: lencse@hit.bme.hu). Submitted: June 3, 2022.

The single translation technologies translate the packets traveling from an IPv4 domain to an IPv6 domain and vice versa (i.e., reverse translation) at the edge between these two domains [3]. The single Device Under Test (DUT) setup of RFC 8219 [3] can help in benchmarking the devices of this type of technologies. Siitperf [7], an RFC 8219 compliant Tester, is an example tool that uses this type of setup to benchmark the Stateless IP/ICMP Translation (SIIT), also called stateless NAT64 single translation technology.

The double translation technologies translate the packets traveling from one IPvX domain to another IPvX domain through IPvY core domain, where X and Y are part of the set {4, 6} and $X \neq Y$. The first translation is taken place at the edge between the first IPvX domain and the IPvY core domain, while the second translation is taken place between the IPvY core domain and the second IPvX domain. However, the reply packets will be reversely double translated in the opposite direction. [3] The devices of this type of technologies (e.g. the CLAT and PLAT devices of the 464XLAT technology, or the CE and BR devices of the MAP-T technology) can be benchmarked using the dual DUT setup of RFC 8219 [3], where both interconnecting devices that are located at the two edges are benchmarked together with the same setup. However, when one of the devices forms a bottleneck, this could hide several potential asymmetries. Therefore, RFC 8219 recommends additional separate benchmarking for each one of the two devices using the single DUT setup.

The encapsulation technologies encapsulate the packets traveling from an IPvX domain by an IPvY header at the edge between the IPvX domain and the IPvY core domain and then decapsulate them at the edge between the IPvY core domain and another IPvX domain before reaching their destination. However, the reply packets will experience reverse encapsulation/decapsulation processes in the opposite direction. [3] The devices of this type of technologies (e.g., the B4 and AFTR devices of the DS-Lite technology, or the lwB4 and lwAFTR devices of the lw4o6 technology) can be benchmarked in the same way as benchmarking double translation technologies. [3]

MAP-T technology [8] is one of the most important transition technologies that belong to the double-translation category, and it is also considered an IPv4-as-a-Service (IPv4aaS) technology, which can give the network operators the option to continue providing their customers with IPv4 services while deploying only IPv6 in their core and access network [9]. This paper aims at progressing the implementation of the world's first RFC 8219 compliant MAP-T Tester.

DOI: 10.36244/ICJ.2022.3.6

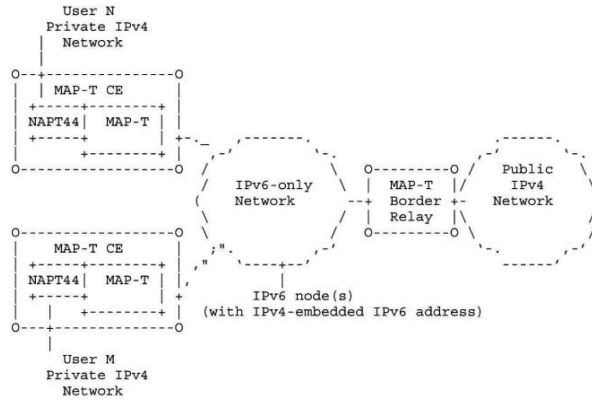

Fig. 1. MAP-T architecture [8]

The remainder of this paper is organized as follows. Section 2 introduces the MAP-T technology. Section 3 discloses the basic operational requirements for the MAP-T Tester based on the recommendations of RFC 8219, followed by a brief description of the scope of measurements. Section 4 discusses the most important design considerations for the Tester. Section 5 gives details on the installation procedure of a testbed for a MAP-T implementation, along with its results. Section 6 describes the MAP-T test programs. Section 7 summarizes further configuration parameters needed by the test programs. Section 8 presents some future plans. Section 9 concludes the paper.

II. Mapping of Address and Port Using Translation (MAP-T) Technology

MAP-T technology [8] helps in accelerating the adoption of IPv6 as it enables service providers to run IPv6-only devices for their operator network while keeping customers run applications that use socket APIs and literal IPv4 addresses. Thus, this technology is considered an *IPv4aaS* technology [10].

MAP-T can be compared to the Mapping of Address and Port using Encapsulation (MAP-E) technology [11] because it also uses mapping rules, but instead of using encapsulation, it deploys a stateless double NAT64 translation, the same as 464XLAT [12] does to perform its function. This procedure targets removing the encapsulation overhead and helps in making IPv4 traffic and IPv6 traffic be treated as similar as possible.

To accomplish its task, MAP-T deploys two types of devices: the Customer Edge (CE) device which is located at the edge of the customer's private network, and the Border Relay (BR) device which is located at the edge of the native IPv4 Internet. Fig. 1 shows the architecture of MAP-T. The CE device connects the privately addressed IPv4 users to the IPv6 network by performing two operations. First, it executes a stateful NAPT to translate the end user's private IPv4 address and port number into the public IPv4 address and port number range assigned to the subscriber, as described by RFC2663 [13]. Then, it performs a stateless NAT46 to map the public IPv4 address and port to IPv6 address. However, the IPv4 destination address will be manipulated differently by embedding it in an IPv6 address, as described by RFC6052 [14], that uses a specific IPv6 prefix to traverse the IPv6 network. When the BR device receives the IPv6 packet, it first performs a similar stateless NAT64 translation to get back the previous public IPv4 address and port and then routes the IPv4 packet into the public IPv4 network. Finally, the CE and BR devices will use the same rules to translate back and forward the reply packets to the user.

One or more CEs and BRs can be connected through an IPv6 network to form a single MAP domain which can be managed by a set of configuration parameters referred to as MAP rules. The CEs and BRs that belong to the same MAP domain will share the same MAP Rules. The service provider can utilize single or multiple MAP domains. The MAP rules are classified as Basic Mapping Rule (BMR), Forwarding Mapping Rule (FMR), and Default Mapping Rule (DMR).

The BMR specifies how the MAP address that all CE devices must share should be built and thus allows assembling MAP addresses out of IPv4 addresses, and vice versa. This rule can be identified by three fields, namely, the Rule IPv6 prefix, the IPv4 prefix assigned for CEs, and the length of Embedded Address (EA) bits. The rule IPv6 prefix is an IPv6 prefix reserved by the service provider for the MAP rule or CE usage, and this means that all CEs belonging to the same MAP domain must use the same rule IPv6 prefix. The IPv4 prefix is the public IPv4 prefix that is assigned by the service provider for the MAP rule or CE usage. The EA bits identify which CE is being used and represent two concatenated subfields: the IPv4 suffix and the Port Set ID (PSID). The PSID specifies which port range is assigned to the CE in case of sharing the IPv4 address among multiple CEs. The format of the MAP address is shown in Fig. 2. The MAP address configured by BMR will represent a customer IPv4 client behind one of the CEs and it is considered the translated form of the source address of the egress packets and the translated form of the destination address of the ingress packets.

The FMR maintains a table of a bunch of BMRs, which will be used by the BR to manage its serviced MAP domain. The BR will use the FMR to translate the source address of the packets received from one of the CEs and to translate the destination address of the packets to be forwarded to one of the CEs.

The DMR is used to form the IPv4-embedded IPv6 addresses for those destinations located outside the MAP domain by adding an IPv6 prefix, which is provisioned from its corresponding BR to the IPv4 public address of the destination. Any CE can use this rule to install an IPv4 default route and to mask the addresses of the devices on the IPv4 internet behind the BR. More precisely, the CEs and BRs will use this rule to translate the destination address of the egress packets and the source address of the ingress packets.

To make things clearer, we refer to the example given by Jool's MAP-T summary [15]. Suppose we have 256 public IP addresses within subnet 192.0.2.0/24 and the number of ports is distributed evenly among customers with 2048 ports each. Thus, we can serve 256*65536/2048= 8192 customers (i.e., we have 65536/2048= 32 port sets within each IP). The PSID will identify each port set. For instance, PSID 0 will identify the set with ports 0-2047, PSID 1 will identify the set with ports 2048-4095, and so on. Note here that the first port set will also include the well-known ports 0-1023. Then, each CE can be identified

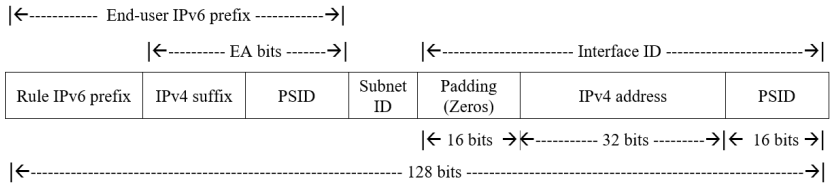

Fig. 2. MAP address format [15]

by the EA-bits, which are composed of a concatenation of the suffix of the public IPv4 address that is assigned to the CE and the PSID. For our example, the suffix is 8-bits because we have 256 IPs and the PSID is 5-bits because we have 32 port sets in each IP. So, the EA-bits are 13-bits. The End-user IPv6 prefix is the only important part of the MAP address, and it can be said that all other fields are essentially cosmetic fields. As the format of the MAP address, shown in Fig. 2, the end-user IPv6 prefix is composed of the Rule IPv6 prefix and the EA-bits. All CEs within the same MAP domain will share the same rule IPv6 prefix. Let us assume it as 2001:db8:ce::/51 for our MAP domain's CEs. Now, the BMR would be the triplet (2001:db8:ce::/51, 192.0.2.0/24, 13) according to its definition stated earlier. An example of the MAP address configured by this BMR, which represents a customer IPv4 client with IPv4 socket 192.0.2.2:2050, could be 2001:db8:ce:41::c000:202:1. Here, 41 represents the EA-bits (00000010 | 00001). The first part (i.e., 00000010) represents the suffix **2** in the last octet of the IPv4 address 192.0.2.**2**, and the second part (i.e., 00001) represents PSID 1 which refers to the second port set because port 2050 is in that port set. The c000:202 is the hexadecimal representation of the public IPv4 address. Whereas the last digit 1 represents the PSID. Now, this address will represent the source address of all IPv6 packets traveling through the related customer's CE to the connected BR via the IPv6 network. While the destination address of these packets is configured by DMR by masking the IPv4 address of the destination, assume it for example 203.0.113.56, by the default IPv6 prefix defined by DMR, assume it for example the NAT64 well-known prefix 64:ff9b::/96. Thus, the IPv4-embedded IPv6 destination address of these packets will be as 64:ff9b::203.0.113.56. However, at the CE, the source address and destination address of the reply packets coming from the BR will adhere to the same rules, but now in a reverse manner, to get their original public IPv4 form. That is, it gets the source address by using DMR and the destination address by using BMR.

On the other side, the source address of the IPv6 packets received by the BR from its connected CE will be translated to its original public IPv4 address according to FMR, which is in our example the same as BMR, while the destination address will be translated according to DMR. Again, the source address and the destination address of the reply packets going to the related CE will be translated according to the same rules, but in a reverse manner, to get their IPv6 form. That is, DMR for the source address and FMR for the destination address.

III. OPERATIONAL REQUIREMENTS AND SCOPE DECISIONS

To emulate, to some extent, the conditions of a production network environment, it is crucial to benchmark IPv6 transition technologies under various operational conditions [3]. This section gives a high-level overview of the operational requirements of the Tester and discloses some considerations about its scope decision.

A. Test and Traffic Setup

As stated earlier, MAP-T is considered a double translation technology. This means that its test setup should follow, in general, the dual DUT test setup as recommended by RFC 8219 [3]. This test setup is briefly described in section 4.2 of this RFC and is exhibited in Fig. 3. The CE should act as DUT1, and the BR should act as DUT 2. However, both have some asymmetries in their behavior. Thus, there should be a separate test based on the single DUT test setup for each one of them according to the recommendations of RFC 8219 [3]. This test setup is briefly described in section 4.1 of this RFC, and it is depicted in Fig. 4. Here, we should also note that the Tester should have translation capabilities as both DUTs to accomplish its task.

There are several test specifications that both test setups should adhere to during testing. These specifications comply with RFC 8219, and they can be summarized as follows:

- Bidirectional traffic must be generated in the tests even though the arrows of the traffic flow are shown as unidirectional in the test setups in Fig. 3 and Fig. 4. However, unidirectional traffic can also be used to get fine-grained performance test results.
- The two IP versions will be used, and they are expressed as IPvX and IPvY, where X=4 and Y=6 in the dual DUT test setup, while X and Y are part of the set {4, 6} and X ≠ Y in the single DUT test setup, that is their exact value depends on what DUT is to be tested (i.e., X is 4 and Y is 6 if the DUT is CE and vice versa if the DUT is BR).
- Although various media types can act as connection media, the tests will rely only on Ethernet.
- Based on RFC 8219, the following frame sizes should be used: 64, 128, 256, 512, 768, 1024, 1280, 1518, 1522, 2048, 4096, 8192, and 9216. In addition, RFC 8219 recommends setting all interfaces of the DUTs and the Tester to use the larger MTU between the physical NICs and virtual translation interfaces to avoid any frame loss due to the MTU mismatch between the two types of interfaces. More specifically, the recommended value to be used is the minimum IPv6 MTU size (i.e., 1280 bytes) plus the translation overhead.

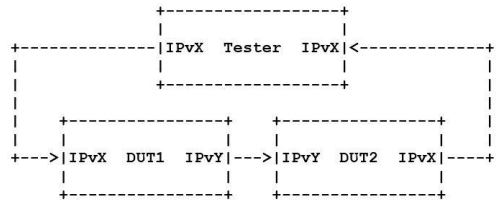

Fig. 3. Dual DUT Test Setup [3]

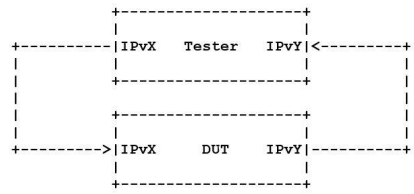

Fig. 4. Single DUT Test Setup [3]

- The IPv4 addresses should be selected based on the recommendations of section 12 of RFC 2544 [5], while the IPv6 addresses should be selected based on the recommendations of section 5 of RFC 5180 [6].
- UDP should be employed as the transport layer protocol for tests.

It is also required that non-translated or native IPv6 traffic, also called "background traffic", must be used besides the translated traffic, and different proportions of the two types must be generated. To accomplish a well-organized testing procedure and to get more precise testing results, we decided to run three types of testing:

1) CE Testing

This test adheres to the single DUT test setup. Here, the CE device will act as the DUT. Both the Tester and the DUT must have one interface configured as IPv4 and another one configured as IPv6. The test starts by sending IPv4 packets from the Tester via its IPv4 interface. The DUT should receive these packets from its IPv4 interface, then performs stateful NAPT translation on their private source address, then translates them into IPv6 packets after forming the IPv6 source address based on BMR (i.e., this address is also called the MAP address) and the IPv6 destination address based on DMR, and then forwards the translated packets via its IPv6 interface to the Tester. When the Tester receives the IPv6 packets from its IPv6 interface, it should be able to get its original IPv4 traffic after translating back the source address based on the same BMR as the CE's one and the destination address based on the same DMR as the CE's one.

To continue testing in the reverse direction, the Tester should first pre-generate IPv6 packets that adhere to the DMR rule (for source address) and BMR rule (for destination address) ahead of starting the test to enhance the speed of the Tester and then send them via its IPv6 interface. When the DUT receives these packets via its IPv6 interface, it should be able to translate them back into IPv4 packets using the same DMR and BMR rules (i.e., to get the IPv4 source address and the public destination address respectively), performs stateful NAPT translation according to the information it has in its local NAPT table, and then forwards the resulted IPv4 packets via its IPv4 interface. Finally, the Tester should, in turn, receive these packets from its IPv4 interface and they should be the same as the original ones before sending them the first time.

2) BR Testing

This test adheres to the single DUT test setup, too. But, here, the BR will act as the DUT. Similarly, both the DUT and the Tester must have one interface configured as IPv4 and another one configured as IPv6. Here, the Tester should first pre-generate templates of IPv6 packets which adhere to BMR (for source address) and DMR (for destination address) ahead of starting the test to enhance the speed of the Tester. The test starts by sending the pre-generated IPv6 packets via its IPv6 interface to the DUT. Once the DUT receives the IPv6 packets from its IPv6 interface, it should be able to translate them back into IPv4 packets after applying the appropriate FMR (i.e., it should select the same BMR as the Tester's one) to get the IPv4 source address and the DMR to get the IPv4 destination address, and then it forwards the IPv4 packets from its IPv4 interface. The Tester should, in turn, receive these packets from its IPv4 interface and they should have similar IPv4 traffic.

To continue testing in the reverse direction, the Tester should first send the IPv4 packets via its IPv4 interface. When the DUT receives these packets, it should translate them into IPv6 packets after applying the DMR to get the IPv6 source address and the appropriate FMR to get the IPv6 destination address, and then it forwards the IPv6 packets via its IPv6 interface. Finally, when the Tester receives these packets via its IPv6 interface, it should translate them back into IPv4 packets after applying the DMR and BMR rules (i.e., to get the IPv4 source and destination addresses respectively) and get its original IPv4 traffic before sending it the first time.

The challenge in the BR Testing is that the Tester should simulate a high number of CEs connected to the BR. This is usually what happens in the production network. Therefore, the Tester should have some approach and configuration settings to make this possible.

3) Overall Testing

Testing both devices under the dual DUT test setup as required by RFC 8219, where the CE device will act as the DUT1 and the BR device will act as the DUT2, can be done with the help of an existing testing tool, which is the stateful branch of `siitperf` [16]. This testing tool follows the benchmarking methodology described in [17].

B. Scope of Measurements

RFC 8219 requires performing different types of performance measurements. Practically, some of these benchmarking measurements are implemented by some existing RFC 2544 testers, while others are either omitted or seldom used such as back-to-back frames, system recovery, and reset. The first two measurement tests require that the Tester must be able to send at the maximum possible rate of the media, which could not be necessarily met by the deployed devices that run the Tester. The latter measurement test requires causing or sensing a DUT reset, and this means we need supplementary hardware. Thus, only those measurement tests that we see as important will be supported by our Tester. In this section, we introduce them and their requirements.

1) Throughput

RFC 8219 reuses the RFC 2544's definition of throughput as it is "the fastest rate at which the count of test frames transmitted by the DUT is equal to the number of test frames sent to it by the test equipment" [5]. That is, no frame loss should occur. Here, the Tester must be able to send frames at any given rate for some period and then count the number of the sent and received frames in that period. It is possible to use a binary search algorithm to find the fastest possible rate and consequently apply this measurement effectively. We should also note that RFC 8219 specified several frame sizes to perform this test.

2) Latency

This measurement is practically based on throughput. Here, a stream of frames, whose duration is at least 120 seconds, should be sent at a particular frame size from the Tester through the targeted DUT at the calculated throughput rate. Some of the frames should be tagged. At least 500 tagged frames should be recognized after 60 seconds from the start of the transmission. For each tagged frame, two timestamps should be recorded, one at the time of fully sending the frame and another at the time of receiving it. The latency will represent the difference value of the two timestamps. Then, the test should calculate two important quantities, the Typical Latency (TL), which represents the median value of the latencies of at least 500 tagged frames, and the Worst-Case Latency (WCL), which represents the 99.9^{th} percentile of them. To get more accurate results, the test must be repeated at least 20 times, and it should eventually record the median value of all TLs and the median value of all WCLs.

3) Packet Delay Variation (PDV)

This measurement includes two variations of tests, Packet Delay Variation (PDV) and Inter Packet Delay Variation (IPDV), both are significantly important, especially for real-time applications. However, our Tester will primarily focus only on calculating PDV as RFC 8219 marks it as recommended, while it marks IPDV as optional for fine-grained analysis of delay variation. In this test, also, a stream of frames should be sent at a particular frame size from the Tester through the targeted DUT at the calculated throughput rate. But the duration of the stream should rather be at least 60 seconds and the one-way latency value of *all* frames should be calculated. Thus, the PDV will represent the difference value between the 99.9^{th} percentile and the minimum delay value in the stream. Similarly, the test must be run at least 20 times and the final recorded value will be calculated from the median of all calculated PDVs.

4) Frame Loss Rate (FLR)

This measurement is similar to the throughput and is also done by sending a stream of frames at some rate through the targeted DUT, but here, we will count the number of received frames by the Tester and then calculate the FLR as in (1):

$$FLR = ((sent - received)/sent) * 100\%. \quad (1)$$

To run this test, a different frame rate will be used at each new trial, starting from the maximum frame rate of the media, and then decreased by some percentage (typically 10%) at each new trial. The test will finish once we find two consecutive trials in which no frames are lost.

IV. DESIGN CONSIDERATIONS

What follows are some important design factors that should be considered when implementing the MAP-T Tester. They are similar to those of the work done in our earlier paper [2] and in the work done in [7], and they follow the same approach of them.

A. Integration or Separation

Building a fully integrated Tester, which can automatically run all measurement tests, can be a desirable solution in case we plan to perform a commodity Tester for routine tests. However, our Tester is intended to be used primarily for research purposes. Thus, we aim to design a more flexible measurement tool that gives the ability to extract some interesting intermediate results by performing only certain important subtasks. For this purpose, the primary functions of our Tester will be implemented using high-performance programs executed by modifiable bash scripts that accept input parameters instead of built-in constants in the programs (e.g., 60 seconds duration or 500 timestamps) as allowed by RFC 8219.

B. Software Architecture and Hardware Requirements

Generally, RFC 8219 requires generating bidirectional traffic in the tests. To build a simple yet efficient program structure, two thread pairs will be used, one for the forward direction (i.e., for processing the packets from the client to the server in Section II) and another for the reverse direction (i.e., for processing the reply packets). Each thread pair consists of a thread for sending and another for receiving. In case each thread will be executed by a single CPU core, this means that we need four CPU cores for communication processes plus an extra CPU core to run the main program. It should also be said that, at any given time, either one of the two directions might be inactive. In addition, each one of the two DUTs and the Tester must have two NICs for testing and an optional extra one for network communication.

C. Input and Output

Building the program structure in such a way that supports separation could help the shell scripts to run the programs multiple times with two forms of parameters, static and dynamic. Those parameters whose values do not change during the execution (e.g., IP addresses, MAC addresses, and so on) will be statically provided in a configuration file. In contrast, those parameters whose values may change during the execution (e.g., frame size, frame rate, and so on) will be provided as command-line arguments.

The results that the shell scripts need to make some decisions during the execution should be printed out on the standard output to be used for further processing. On the other hand, those results which are big or have no longer been used by the shell script should be stored in an output file.

V. MAP-T TESTBED

Before going further with the implementation of the MAP-T Tester, we built a testbed for one of the implementations of MAP-T called Jool [15] to make sure it is working as expected and to check the operation of the implemented Tester on a valid MAP-T implementation. This testbed is shown in Fig. 5, which uses the same IP addresses used in the example of [15]. This example is also discussed in Section II. The testbed is installed using a workstation with the following specifications: Intel(R)

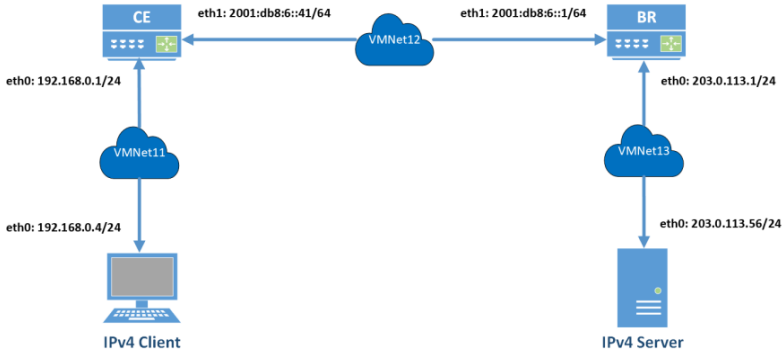

Fig. 5. MAP-T Testbed

Core(TM) i7-3612QM CPU @ 2.10GHz CPU, 8.00 GB RAM:, 1TB HDD, and Windows 10 64-bit Operating System.

Each network node in the testbed is created as a virtual machine using VMware with Debian 10 installed on a single-core processor and supplied with 1GB of RAM and 40GB of Hard Disk. The network nodes are connected via three virtual networks as follows:

- VMNet11 connects the IPv4 Client's eth0 interface to the CE's eth0 interface and it is IPv4 only.
- VMNet12 connects the CE's eth1 interface to the BR's eth1 interface and it is IPv6 only.
- VMNet13 connects the BR's eth0 interface to the IPv4 Server's eth0 interface and it is IPv4 only.

TABLE I shows the Debian and VMware network settings used for each one of the virtual machines.

Furthermore, two shell scripts are written, one is executed at the CE and the other is executed at the BR. Most of the code of the shell scripts follows the configuration steps mentioned in [18]. The **`CE-script.sh`** and **`BR-script.sh`** are available on GitHub [19].

Then, the HTTP service is activated at the IPv4 server by running this Linux command:

```
root@ipv4server# service apache2 start
```

To test the functionality of the network including the CE and the BR, an HTTP request is sent from the IPv4 client to the IPv4 server via this Linux command:

```
root@ipv4client# wget http://203.0.113.56/index.html
```

Consequently, the IPv4 client received the `index.html` page successfully, and this is the result when the "`head`" Linux command was run:

```
root@ipv4client# head -2 index.html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
```

Moreover, when the traffic was captured at eth1 of the CE, as shown in Fig. 6, and eth0 of the BR, as shown in Fig. 7, using tcpdump, it showed proper translations for the HTTP packets.

This gives the result that the Jool's MAP-T implementation of the CE and the BR is working properly and could be tested by our MAP-T Tester.

VI. Description of Test Programs

Three test programs are intended to implement the different four measurements: throughput, latency, PDV, and FLR. These programs are the **maptperf-tp**, which measures the throughput as well as the FLR, the **maptperf-lat**, which measures the latency, and the **maptperf-pdv**, which measures the PDV. The design and implementation of these programs are inspired by the approach followed in [7]. The three programs use the following common parameters, which can be inserted as command-line arguments:

- **IPv6 frame size**: The size of the frames must be according to what is mentioned in Section III.A. The IPv4 frames will automatically be 20 bytes shorter.

TABLE I
Debian and VMware Network Settings

VM Name	Linux Settings			VMware Settings		
	eth0	eth1	eth2	eth0	eth1	eth2
IPv4 Client	Static 192.168.0.4/24	DHCP	N/A	VMNet11	NAT	N/A
CE	Static 192.168.0.1/24	Static 2001:db8:6::41/64	DHCP	VMNet11	VMNet12	NAT
BR	Static 203.0.113.1/24	Static 2001:db8:6::1/64	DHCP	VMNet13	VMNet12	NAT
IPv4 Server	Static 203.0.113.56/24	DHCP	N/A	VMNet13	NAT	N/A

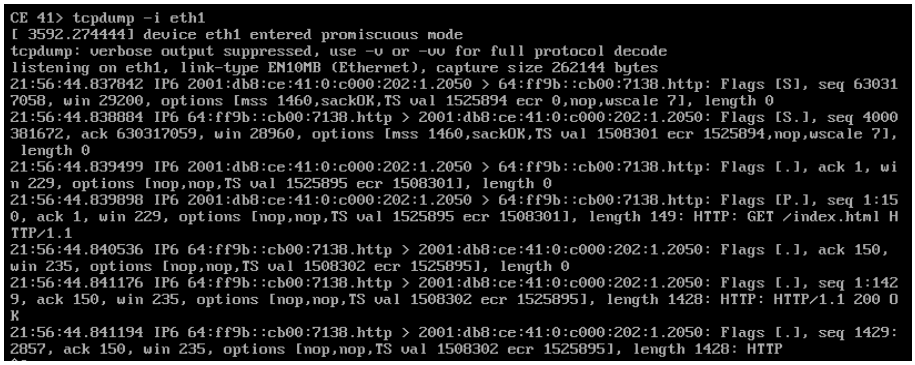

Fig. 6. CE's tcpdump traffic capture at eth1

- **frame rate**: the rate at which the frames will be sent, and it is calculated as frames per second.
- **test duration**: It must be 1-3600 seconds.
- **stream timeout**: the Tester must stop receiving frames when this timeout elapses after sending them completely. This parameter could be compared to the 2000 milliseconds "after sending timeout" recommended by RFC 2544 in its section 23 and complied with RFC 8219 recommendations.
- **n**: a relatively prime number to m, which both help in specifying the proportions of foreground and background frames.
- **m**: a relatively prime number to **n**, which represents the number of foreground frames. Thus, the background traffic will be (**n-m**) frames.

In addition to the abovementioned common parameters, **maptperf-lat** has two further ones:

- **first tagged delay**: The time required to be spent before the first tagged frame can be sent. It must be 0-3600 seconds.
- **tagged**: The number of tagged frames. It must be 1-50,000.

While **maptperf-pdv** has this further one:

- **frame timeout**: In case the value of this parameter is greater than 0 milliseconds, then the delay of every single frame will be checked against it. Then, the frame will be considered "lost" if its delay exceeded the value of this parameter. However, if the value of this parameter is 0, then, the **maptperf-pdv** will calculate PDV as described by RFC 8219.

What follows is the description of each one of these programs:

A. maptperf-tp

This test program sends a stream of frames to the DUT for **test duration** seconds and continues receiving them simultaneously from the DUT until the **stream timeout** elapses. Then, it reports the number of sent frames and the number of received frames for each one of the active directions (i.e. forward and reverse (one of them could be disabled)). The throughput test could be passed if and only if the number of sent frames and the number of received frames are equal for the active directions. However, the FLR could, then, be easily calculated as in (1) (see Section III .B.4).

B. maptperf-lat

This test program also sends a stream of frames to the DUT for **test duration** seconds, but some of these frames are tagged. The first tagged frame will not be sent until the **first tagged delay** elapses. The selected frames to be tagged should be exactly equal to the **tagged** parameter value and should be identified according to the uniform time distribution and sent during the remaining test time (i.e., **test duration** – **first tagged delay**). The test program must continue receiving frames from the DUT until the **stream timeout** expires. For each tagged frame, the test program must report the time at which it completes sending the entire frame and the time at which it finishes receiving it completely. Any tagged frame could be

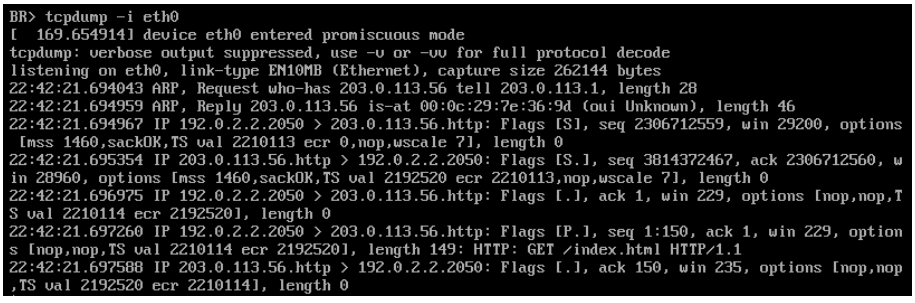

Fig. 7. BR's tcpdump traffic capture at eth0

considered "lost" if not received within a time equal to **test duration - first tagged delay + stream timeout**. Then, the TL and WCL values could be calculated with the help of all reported sending and receiving time values of the tagged frames, as described in Section III.B.2, for the active directions (i.e., forward and reverse).

C. maptperf-pdv

This test program also sends a stream of frames to the DUT for **test duration** seconds, but no tagging will be made here, instead, every frame sent must be recognized by a unique identifier. Similarly, the test program must continue receiving frames from the DUT until the **stream timeout** expires. For *all* frames, the test program must report the time at which it completes sending the entire frame and the time at which it finishes receiving it completely. Then, the test program checks the value of the **frame timeout** parameter. If it is 0, then, it will calculate the PDV as stated in Section III.B.3, for the active directions (i.e., forward and reverse), and any frame will be considered "lost" if not received within a time equal to **test duration + stream timeout**. If the value of the **frame timeout** parameter is greater than 0, then the test program will not calculate PDV, rather it will use the **frame timeout** as a frame loss specifier. That is, if the delay of any single frame exceeds its value, then it will be considered "lost". This gives a more precise approach to measuring throughput and FLR, as recommended in [20]. But the performance penalty would be greater as recording and dealing with timestamps could consume more memory and CPU cycles. Consequently, the **maptperf-pdv** can act as a dual test program. So, it will act as either a PDV tester if the value of the **frame timeout** is 0, or a precise throughput and FLR tester if the value of the **frame timeout** is greater than 0.

VII. FURTHER CONFIGURATION PARAMETERS

Besides the configuration parameters that can be provided to the test programs as command line arguments as described in Section IV, the test programs must also be supplied with some other parameters whose values could not be changed during the execution of the test. These static parameters along with their values are recorded into a configuration file, which will then be used by the test programs during their execution.

In this paper, we will focus on those parameters that must be recorded into the configuration file of the BR testing, as we see that, in practice, a high number of CEs are used together with a single BR, and the scalability of the system will mainly rely on the scalability of the BR itself. What follows is a summary describing them briefly:

A. Basic parameters

The following parameters are the basic ones:

- **Tester-L-IPv6**: The IPv6 address of the left-side interface of the Tester. It should be an address in the same subnet as that of the DUT-L-IPv6. However, it will not represent the source address of the test packets in the forward direction, instead, the source address will be formed by the BMR as the Tester will simulate many CEs. To make things simple and efficient, we deliberately skipped the NAPT function of the CEs and the IPv4 client settings. In addition, the Tester will pseudorandomize the BMR's EA-bits value depending on its length to get various (IPv4 suffix + PSID) values, each of which representing a different CE device. The BMR's IPv4 Prefix, however, is the same for all generated public IPv4 addresses of the CEs.
- **Tester-R-IPv4**: The IPv4 address of the right-side interface of the Tester. It should be an address in the same subnet as that of the DUT-R-IPv4, as Tester will simulate an IPv4 server. This address will also represent the destination IPv4 address for the test packets in the forward direction.
- **Tester-R-IPv6**: The IPv6 address that will be used by the Tester's interface for forwarding background (i.e., non-translated) traffic.
- **DUT-L-IPv6**: The IPv6 address that is currently assigned to the left-side interface of the DUT.
- **DUT-R-IPv4**: The IPv4 address that is currently assigned to the right-side interface of the DUT.
- **DUT-R-IPv6:** The IPv6 address that will be used by the DUT for forwarding background (i.e., non-translated) traffic.
- **Tester-L-MAC**: The MAC address of the left-side interface of the Tester.
- **Tester-R-MAC**: The MAC address of the right-side interface of the Tester.
- **DUT-L-MAC**: The MAC address of the left-side interface of the DUT.
- **DUT-R-MAC**: The MAC address of the right-side interface of the DUT.

The following other basic parameters specify the range of the destination port numbers in the forward direction and the range of the source port numbers in the reverse direction. They are the same as those of the extension of Siitperf [21], which implemented a random port feature originally pointed to by RFC 4814 [22]. There is no specific restriction about the values of these ranges. So, one could use the entire port space, that is 0-65535, as stated in [21]. There are also some other settings recommended by RFC 4814 [22]. We also follow the same approach of [21] to pseudorandomly generate port numbers from these ranges:

- **FW-dport-min**: The lowest number in the range of destination port numbers that can be used by the test packets in the forward direction.
- **FW-dport-max**: The highest number in the range of destination port numbers that can be used by the test packets in the forward direction.
- **RV-sport-min**: The lowest number in the range of source port numbers that can be used by the test packets in the reverse direction.
- **RV-sport-max**: The highest number in the range of source port numbers that can be used by the test packets in the reverse direction.

It may be noticed that the parameters of the source port range in the forward direction and the parameters of the destination port range in the reverse direction are not included with the BR testing parameters because the ports must be

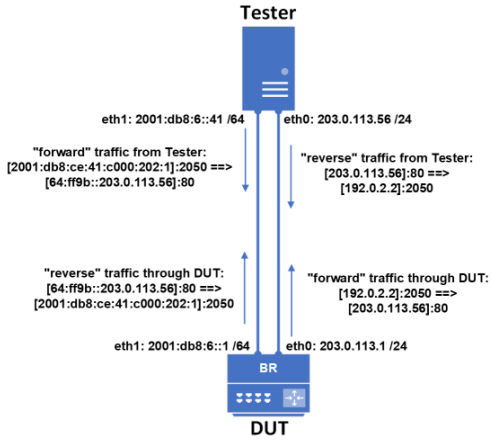

Fig. 8. Translated traffic flow during benchmarking MAP-T BR

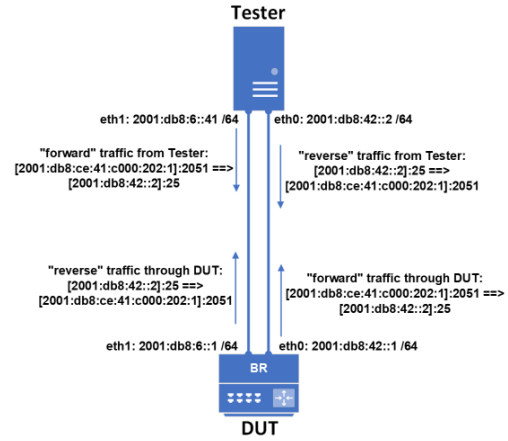

Fig. 9. Background traffic flow during benchmarking MAP-T BR

selected from a specific port set that is assigned to the simulated CE. Therefore, they will be computed by the Tester program and not prerecorded in the configuration file.

The pseudorandomized PSID will determine which port set will be used by the Tester (i.e. the simulated CE). The PSID value can be extracted from the first right "X" bits of the pseudorandomized EA-bits, where X = EA-Len – the IPv4 suffix length. The IPv4 suffix length can be easily determined because the BMR-IPv4-Prefix is set in the configuration file with its length, for example, /24. So, the IPv4 suffix length will equal 32 (i.e., IPv4 address length) minus the BMR-IPv4-Prefix length. Now, to know the range of port numbers in the port set identified by that PSID, we first must know how many port sets are there (we denote this as Y), that is Y = 2 ^ X, and then how many ports can be used in each port set (we denote this as Z), that is Z = (2^16 / Y), where 16 is the number of bits of any port address. Then, the FW-sport-min and the RV-dport-min will be equal to (PSID x Z), while the FW-sport-max and the RV-dport-max will be equal to (((PSID+1) x Z) – 1).

B. MAP rules parameters

The parameters that are related to MAP rules are the followings:

- **BMR-IPv6-Prefix**: The BMR's Rule IPv6 Prefix of the MAP address. As stated earlier, all CEs within the same MAP domain will share this IPv6 Prefix.
- **BMR-IPv4-Prefix**: The BMR's public IPv4 prefix that is reserved for CEs. However, the public IPv4 suffix plus the PSID (i.e., EA bits) will, then, uniquely identify each CE.
- **BMR-EA-Len**: The number of EA bits (i.e., EA length).
- **DMR-IPv6-Prefix**: The IPv6 prefix that will be added by DMR to the public IPv4 address to form the IPv4-embedded IPv6 address.

C. Device hardware parameters

The parameters that are related to the device hardware are the followings: (Some of them are needed by DPDK)

- **CPU-FW-Send**: The CPU core to be used by the thread of sending in the forward direction.
- **CPU-FW-Receive**: The CPU core to be used by the thread of receiving in the forward direction.
- **CPU-RV-Send**: The CPU core to be used by the thread of sending in the reverse direction.
- **CPU-RV-Receive**: The CPU core to be used by the thread of receiving in the reverse direction.
- **Mem-Channels**: The number of memory channels to be used. Setting this parameter is optional. The default value is 1.

D. Network traffic parameters

The parameters that are related to the network traffic are the followings:

- **FW**: The forward direction becomes active if set to 1.
- **RV**: The reverse direction becomes active if set to 1.
- **Promisc**: The promiscuous mode will be active if set to a non-zero value.

What follows are the contents of the *Tester.conf* file for the example MAP-T BR test setup depicted in Fig. 8 and Fig. 9. The figures also show the flow of the translated traffic and the flow of the background traffic, respectively. The IP addresses and the port settings are taken from the example of [15], which is also discussed in Section II.

Tester.conf:

```
#Basic parameters
    Tester-L-IPv6: 2001:db8:6::41/64
    Tester-R-IPv4: 203.0.113.56/24
    Tester-R-IPv6: 2001:db8:42::2/64
    DUT-L-IPv6: 2001:db8:6::1/64
    DUT-R-IPv4: 203.0.113.1/24
    DUT-R-IPv6: 2001:db8:42::1/64
    Tester-L-MAC: 00:0c:29:95:f6:a9
    Tester-R-MAC: 00:0c:29:95:f6:b3
    DUT-L-MAC: 00:0c:29:7f:37:48
    DUT-R-MAC: 00:0c:29:7f:37:52
#Port ranges parameters
    FW-dport-min: 1 #as RFC4814 recommends
    FW-dport-max: 49151 #as RFC4814 recommends
    RV-sport-min: 1024 #as RFC4814 recommends
    RV-sport-max: 65535 #as RFC4814 recommends
#MAP rules parameters
    BMR-IPv6-Prefix: 2001:db8:ce::/51
    BMR-IPv4-Prefix: 192.0.2.0/24
    BMR-EA-Len: 13
```

```
    DMR-IPv6-Prefix: 64:ff9b::/96
#Device hardware parameters
    CPU-FW-Send: 2
    CPU-FW-Receive: 4
    CPU-RV-Send: 6
    CPU-RV-Receive: 8
    Mem-Channels: 2
#Network traffic parameters
    FW: 1
    RV: 1
    Promisc: 0
```

VIII. FUTURE WORK

The next step is to implement the MAP-T Tester using C++ language and the DPDK framework [23], a high-performance user-space networking solution that offers fast packet processing and efficient memory, queue, and buffer management. Next, the Tester should be validated by comprehensive benchmarking tests.

The Tester will be developed as free software under the GPL license, and it could be reused as a model for developing testers for other IPv6 transition technologies, especially those IPv4aaS technologies that have not been benchmarked yet.

IX. CONCLUSION

In this paper, we described a brief design structure for a MAP-T technology Tester that complies with the RFC 8219 guidelines and recommendations, followed by a high-level view of the operational requirements, the scope of measurements, and the design factors that should be considered when implementing the Tester program. Then, we presented the installation steps of a testbed for a MAP-T implementation and showed its results. And finally, we disclosed our planned MAP-T BR test program and its related parameters, accompanied by a discussion about how to set the values of these configuration parameters.

REFERENCES

[1] A. Al-Azzawi, "Towards the security analysis of the five most prominent IPv4aaS technologies," *Acta Technica Jaurinensis*, vol. 13, no. 2, pp. 85–98, Mar. 2020, DOI: 10.14513/actatechjaur.v13.n2.530.

[2] A. Al-hamadani, and G. Lencse, "Design of a Software Tester for Benchmarking Lightweight 4over6 Devices," in *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, 2021, pp. 157–161, DOI: 10.1109/TSP52935.2021.9522607.

[3] M. Georgescu, L. Pislaru, and G. Lencse, "Benchmarking methodology for IPv6 transition technologies," IETF RFC 8219, Aug. 2017, DOI: 10.17487/RFC8219.

[4] E. Nordmark, and R. Gilligan, "Basic transition mechanisms for IPv6 hosts and routers," IETF RFC 4213, Oct. 2005, DOI: 10.17487/RFC4213.

[5] S. Bradner, and J. McQuaid, "Benchmarking methodology for network interconnect devices," IETF RFC 2544, Mar. 1999, DOI: 10.17487/RFC2544.

[6] C. Popoviciu, A. Hamza, G. V. d. Velde, and D. Dugatkin, "IPv6 benchmarking methodology for network interconnect devices," IETF RFC 5180, May. 2008, DOI: 10.17487/RFC5180.

[7] G. Lencse, "Design and implementation of a software tester for benchmarking stateless NAT64 gateways," *IEICE Transactions on Communications*, vol. E104-B, no. 2, pp. 128-140, Feb. 2021, DOI: 10.1587/transcom.2019EBN0010.

[8] X. Li, C. Bao, E. W. Dec, O. Troan, S. Matsushima et al., "Mapping of Address and Port using Translation (MAP-T)," IETF RFC 7599, Jul. 2015, DOI: 10.17487/RFC7599.

[9] G. Lencse, J. P. Martinez, L. Howard, R. Patterson, and I. Farrer, "Pros and cons of IPv6 transition technologies for IPv4aaS," active Internet Draft, Jan. 2021, https://tools.ietf.org/html/draft-ietf-v6ops-transition-comparison-00

[10] A. T. H. Al-hamadani, and G. Lencse, "A survey on the performance analysis of IPv6 transition technologies," *Acta Technica Jaurinensis,* vol. 14, no. 2, pp. 186–211, May. 2021, DOI: 10.14513/actatechjaur.00577.

[11] E. O. Troan, W. Dec, X. Li, C. Bao, S. Matsushima et al., "Mapping of Address and Port with Encapsulation (MAP-E)," IETF RFC 7597, Jul. 2015, DOI: 10.17487/RFC7597.

[12] M. Mawatari, M. Kawashima, and C. Byrne, "464XLAT: Combination of Stateful and Stateless Translation," IETF RFC 6877, Apr. 2013, DOI: 10.17487/RFC6877.

[13] P. Srisuresh, and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations," IETF RFC 2663, Aug. 1999, DOI: 10.17487/RFC2663.

[14] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, and X. Li, "IPv6 addressing of IPv4/IPv6 translators," IETF RFC 6052, Oct. 2010, DOI: 10.17487/RFC6052.

[15] Jool. "Jool MAP-T Summary," [Online]. Available: https://www.jool.mx/en/map-t.html.

[16] G. Lencse, "Design and implementation of a software tester for benchmarking stateful NATxy gateways: Theory and practice of extending siitperf for stateful tests," *Computer Communications*, Jun. 2022, DOI: 10.1016/j.comcom.2022.05.028.

[17] G. Lencse, and K. Shima, "Benchmarking Methodology for Stateful NATxy Gateways using RFC 4814 Pseudorandom Port Numbers," active Internet Draft, Mar. 2022, https://datatracker.ietf.org/doc/html/draft-lencse-bmwg-benchmarking-stateful

[18] Jool. "Jool MAP-T Run," [Online]. Available: https://www.jool.mx/en/run-mapt.html.

[19] A. Al-hamadani., "MAP-T Testbed," [Online]. Available: https://github.com/alhamadani-ahmed/MAP-T-Testbed.

[20] G. Lencse, and K. Shima, "Performance analysis of SIIT implementations: Testing and improving the methodology," *Computer Communications*, vol. 156, pp. 54–67, Apr. 2020, DOI: 10.1016/j.comcom.2020.03.034.

[21] G. Lencse, "Adding RFC 4814 random port feature to siitperf: Design, implementation and performance estimation," *International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems*, vol. 9, no. 3.

[22] D. Newman, and T. Player, "Hash and Stuffing: Overlooked Factors in Network Device Benchmarking," IETF RFC 4814, Mar. 2007, DOI: 10.17487/RFC4814.

[23] D. Scholz, "A look at Intel's dataplane development kit," in *Proc. Seminars Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, Munich, 2014, pp. 115–122, DOI: 10.2313/NET-2014-08-1_15.

Ahmed Al-hamadani graduated from Florida Institute of Technology (FIT), the USA in 2013 with an MSc degree in Computer Science. Since then, he has worked as a lecturer at the Department of Computer Engineering, University of Mosul, Iraq. Ahmed has been awarded the Stipendium Hungaricum scholarship to do his Ph.D. in informatics at Budapest University of Technology and Economics (BME), Hungary. He started his study in the Department of Networked Systems and Services in September 2020. His research field is the benchmarking and performance analysis of IPv6 Transition Technologies.

Gábor Lencse received his MSc and Ph.D. in computer science from the Budapest University of Technology and Economics, Budapest, Hungary in 1994 and 2001, respectively. He has been working full-time for the Department of Telecommunications, Széchenyi István University, Győr, Hungary since 1997. Now, he is a Professor. He has been working part-time for the Department of Networked Systems and Services, Budapest University of Technology and Economics as a Senior Research Fellow since 2005.
His research interests include the performance and security analysis of IPv6 transition technologies. He is a co-author of RFC 8219.