IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Received 23 May 2025, accepted 27 June 2025, date of publication 10 July 2025, date of current version 21 July 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3587302

RESEARCH ARTICLE

Benchmarking the Mapping of Address and Port With Encapsulation Border Relay Routers

AHMED AL-HAMADANI AND GÁBOR LENCSE
Department of Networked Systems and Services, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, 1111 Budapest, Hungary

Corresponding author: Gábor Lencse (lencse@hit.bme.hu)

ABSTRACT Mapping of Address and Port with Encapsulation (MAP-E) is considered one of the most important IPv6 transition technologies, providing a high level of scalability thanks to its utterly stateless behavior in the operator network. The primary device executing this technology is the Border Relay (BR), which represents the pivot of its scalability. Thus, benchmarking the performance and scalability of this device using a solution based on existing accredited standards is crucial for service providers before its deployment in their core networks. The Benchmarking Working Group (BMWG) of the Internet Engineering Task Force (IETF) has published a comprehensive guide for such a task in its Request for Comments (RFC) 8219. The research work presented in this paper involves conducting comprehensive benchmarking for the MAP-E BR implementation of the FD.io Vector Packet Processing (VPP), a well-known high-performance open-source networking software, and analyzing its performance and scalability, using two different test systems. To accomplish the research, a MAP-E encapsulation capability has been integrated into "Maptperf," an RFC 8219 compliant research Tester developed originally to benchmark the Mapping of Address and Port using Translation (MAP-T) BR. The results of benchmarking the MAP-E BR of VPP demonstrated the functionality and efficiency of the Tester encapsulation plugin as well as the high scalability of the tested implementation, even when serving an increasing number of Customer Edge (CE) devices or deploying a varying number of worker threads in the test. Moreover, they showed that the tested implementation achieved high performance levels in throughput, Frame Loss Rate (FLR), latency, and Packet Delay Variation (PDV) tests, despite utilizing a low number of worker threads.

INDEX TERMS Benchmarking, border relay, IPv6 transition technologies, MAP-E, performance analysis.

I. INTRODUCTION

The IPv4-as-a-Service (IPv4aaS) IPv6 transition technologies [1] support a seamless, gradual transition to full IPv6 address pool use by still providing IPv4 connectivity and services for customers while enabling the network operators to maintain IPv6-only core and access networks. The most prominent ones among these technologies are the Combination of Stateful and Stateless Translation (464XLAT) [2], Dual-Stack Lite (DS-Lite) [3], Lightweight 4 over 6 (Lw4o6) [4], Mapping of Address and Port with Encapsulation (MAP-E) [5], and Mapping of Address and Port using Translation (MAP-T) [6].

MAP-T and MAP-E technologies can be considered the best choices for service providers in terms of scalability due to their entirely stateless behavior within their core network, specifically the functionality of their Border Relay (BR) device, which is relied on using specific MAP rules. Thus, benchmarking the focal point of scalability for these technologies, represented by BR performance, is a key step in facilitating the broad adoption of these technologies in the operation of service providers during the transition to IPv6.

The Benchmarking Working Group (BMWG) of the Internet Engineering Task Force (IETF) presented a comprehensive methodology that builds upon the predecessors, RFC 2544 [7] and RFC 5180 [8]. Still, it specified to benchmark the various IPv6 transition technologies in its RFC 8219 [9]. It puts the MAP-T and MAP-E technologies into two different

The associate editor coordinating the review of this manuscript and approving it for publication was Guangjie Han.

categories. The former is under the double translation category, whereas the latter is under the encapsulation category. Yet, both can be benchmarked using the same methodology guidelines, which involve the dual Device Under Test (DUT) test setup and can be carried out without a technology-specific Tester [10]. However, due to the distinct asymmetry in the behavior of the primary devices, the BR and the Customer Edge (CE), RFC 8219 recommends benchmarking each technology device separately via its single DUT test setup, which requires designing and implementing a specific Tester for each technology. Section VII-B of [10] also points out the limitations of using the dual DUT test setup when benchmarking both devices together as follows:

1) It is not an easy task to reveal which device represented the bottleneck in the test. In contrast, if one deliberately wants to make the BR, for example, form a bottleneck to test its scalability, this rather requires deploying a high number of CE devices simultaneously, which is, in practice, difficult to satisfy.
2) The benchmarking results describe the aggregate performance of both devices together. Thus, extracting the metric values of each individual device is typically a complicated process. For instance, when measuring the latency, assessing the delay proportion each individual device incurred during the test is very unclear, as their function and behavior are significantly non-identical. The same experience applies to the PDV or FLR.

This paper conducts comprehensive benchmarking experiments for analyzing the performance and scalability of the MAP-E BR implementation of the FD.io VPP [11] using two different test systems based on the single DUT test setup of RFC 8219. To accomplish the research work, a MAP-E encapsulation capability was developed and plugged into "Maptperf," an RFC 8219-compliant tester originally built to benchmark the MAP-T BR device. The design and implementation of the former "Maptperf" were presented, and validated via benchmarking the performance of a free MAP-T BR implementation, called Jool [12], in an earlier paper [13]. Hence, the Tester name is updated to "Mapperf" instead. The new tester can run in two operational modes: translation to benchmark the MAP-T BR device and encapsulation to benchmark the MAP-E BR device.

The main contribution of this paper can be summarized as follows:

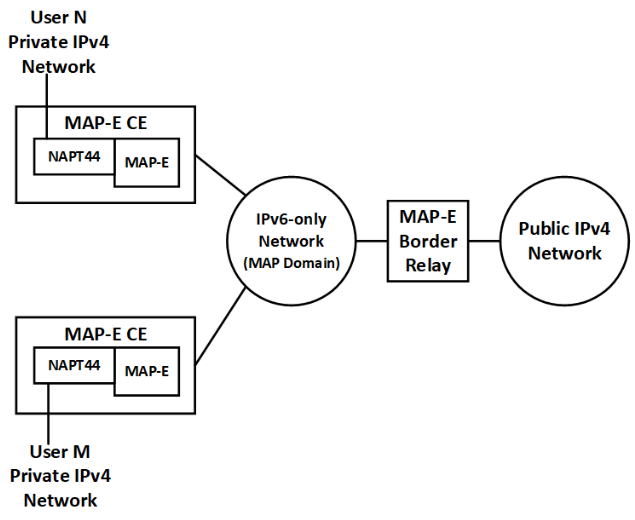
1) Introducing the world's first free software MAP-E BR testing tool that complies with the RFC 8219 standard guidelines.
2) Proving the high scalability of the MAP-E technology, which was assumed by its stateless nature, but has not been proven empirically.
3) Providing network operators with ready-to-use measurement data about the performance and scalability of the VPP MAP-E BR implementation, which was benchmarked using the Tester developed.

The remainder of this paper is structured as follows. Section II introduces the MAP-E technology. Section III provides an overview of the benchmarking methodology outlined in RFC 8219 and how it can be used in measuring the performance of the MAP-E technology. Section IV summarizes the design and implementation of the tester along with the details of its new MAP-E plugin. Section V describes the benchmarking environment, including the test and traffic setup, details of the measurement systems, and the tested implementation. Section VI presents and discusses the benchmarking results. Section VII proposes future plans in the field of benchmarking IPv6 transition technologies. Section VIII concludes the paper.

II. THE MAP-E IPV6 TRANSITION TECHNOLOGY

The MAP-E is one of the most essential IPv4aaS IPv6 transition technologies [1] featured in its scalability, thanks to using different mapping rules that yield stateless routing behavior in the service provider's core network. The Basic Mapping Rule (BMR) and the Forwarding Mapping Rule (FMR) are the two basic rules governing the IP address mappings in the MAP-E network. They will be discussed in detail later in this section.

As shown in Fig. 1, which exhibits the MAP-E architecture, this technology deploys two primary devices to transfer packets from one side of the network to another. These devices are the CE and the BR. The former encapsulates packets sent from the subscriber's IPv4 private network via the IPv6 operator network and decapsulates those received in the opposite direction. In contrast, the latter device encapsulates packets sent from the native IPv4 public network via the IPv6 operator network and decapsulates those received in the opposite direction. A high number of CEs can be connected to one or more BRs to form a MAP domain whose mapping rules are shared among all its devices. The service provider can manage a single or multiple MAP domains.



User N Private IPv4 Network
MAP-E CE
NAPT44
MAP-E
IPv6-only Network (MAP Domain)
MAP-E Border Relay
Public IPv4 Network
MAP-E CE
NAPT44
MAP-E
User M Private IPv4 Network

FIGURE 1. MAP-E architecture [5].

A. THE OPERATION OF MAP-E

The MAP-E technology enables IPv4 address sharing among multiple CEs of a MAP domain, by which they can share the same public IPv4 address by dividing the port range into fixed-sized port sets. Each port set can be identified by a unique Port Set ID (PSID) and must be exclusively assigned to a single CE. Hence, each CE can be distinguished by its public IPv4 address along with the assigned PSID, and it is forcefully required to use its specified range of ports for communication. In contrast, the IPv4 applications of the clients of that CE are not obliged by this limitation, and they are allowed to use the entire port range when transmitting IPv4 packets [14]. The CE is, then, responsible for translating the subscriber's private IPv4 address and port number into the shared public IPv4 address and the port number within the restricted port set by running a stateful Network Address and Port Translation (NAPT) [15]. Therefore, the IPv4 packet to be sent will carry the public IPv4 address as the source address and the restricted port number as the source port. Next, the CE encapsulates the IPv4 packet within an IPv6 packet before transmitting it through the IPv6 operator network.

The source address of the IPv6 packet will be constructed according to the BMR mapping rule; thus, it is referred to as the *MAP address*, whereas its destination address will represent the IPv6 address of the corresponding BR device in the MAP domain. At the other end of the tunnel, when the BR receives the IPv6 packet, it verifies whether the IPv6 source address (i.e., the MAP address of the CE) is valid and that the used source port is within the allowed port range of the CE. If the verification is successful, the BR will decapsulate the packet to obtain the original IPv4 packet and utilize its destination address to route it in the public IPv4 network toward its intended destination. Later, any packet received in the opposite direction (i.e., from the public IPv4 network and intended for a subscriber's application in the private IPv4 network) will experience a reverse (encapsulation/decapsulation) process and the use of the same mapping rules for verification and construction of the MAP address.

Both the CE and the BR maintain a MAP rule table whose entries can be one of two types of MAP rules. The description and purpose of each type can be found in the following subsections.

B. THE BASIC MAPPING RULE

This rule is used to construct the IPv6 MAP address of the CE, which represents either the source address of all outgoing IPv6 packets or the destination address of all incoming IPv6 packets of that CE in its particular MAP domain. Fig. 2 shows the structure of the IPv6 MAP address. Additionally, the BMR is used to verify whether the transport layer source port of the outgoing IPv6 packets or the destination port of the incoming IPv6 packets falls within the allowed port set of the CE.

C. THE FORWARDING MAPPING RULE

This rule is merely similar to the BMR. However, it is used by the CE to derive and verify the IPv6 MAP address of another CE within its MAP domain. It is also checked to validate if the used port number is within the assigned port set of that CE. The rule table of the CE will contain FMR entries only if the "mesh mode" is activated (i.e., a direct "CE-to-CE" connectivity is possible). In other words, there will be no FMR entries if the "hub-and-spoke" mode is used where all traffic must be forwarded directly to the BR. In contrast, the BR will have an FMR entry for each connected MAP domain.

Both rules consist of the following triplet of parameters:

1) *Rule IPv6 prefix* (including its length), which is provisioned by the service provider to all CEs belonging to the same MAP domain to identify their participation in that domain.
2) *Rule IPv4 prefix* (including its length) satisfies the IPv4 address sharing among CEs within their MAP domain.
3) The *Embedded Address* (EA) length represents the number of bits in the IPv6 MAP address that specify the EA field whose value includes both the IPv4 suffix and the PSID assigned to the CE and uniquely identifies it in the MAP domain.

III. THE RFC 8219 BENCHMARKING METHODOLOGY

The BMWG of the IETF provided a comprehensive benchmarking methodology for the various IPv6 transition technologies. First, it categorized them into four classes: *dual stack*, *single translation*, *double translation*, and *encapsulation*, based on the way the technology deploys for traversing the core network. The methodology then defined two types of test setups: the *single DUT test setup* and *the dual DUT*

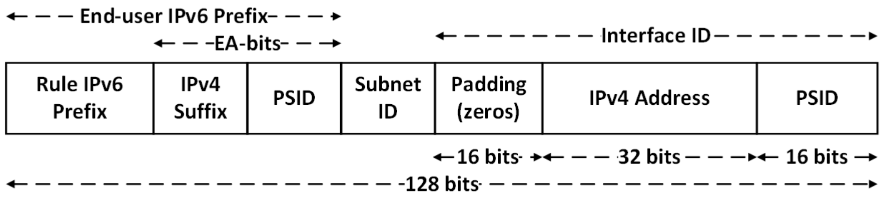


FIGURE 2. MAP address format (based on [12]).

test setup. The former is used when measuring those technologies of a single DUT, such as the single translation technologies (e.g., NAT64 [16], IVI [17]), which translates IPvX packets received by the IPvX interface from the tester to IPvY packets to be sent by the IPvY interface to the tester and vice versa in the opposite direction, where X and Y are part of the set {4,6} and X≠Y. Fig. 3 exhibits the single DUT test setup.

In contrast, the dual DUT test setup is used when measuring those technologies utilizing two DUTs, such as the double translation technologies (e.g., 464XLAT [2] or MAP-T [6]), which translate the IPvX packets received from one interface of the tester to IPvY packets at the first DUT, send them to the second DUT, and then translate the IPvY packets again to IPvX packets at the second DUT before sending them to the other interface of the tester, or encapsulation technologies (e.g., DS-Lite [3], MAP-E [5], or Lightweight 4over6 [4]), which encapsulate the IPvX packets received by one interface of the tester by IPvY header at the first DUT, send them to the second DUT, and then decapsulate the IPvY header at the second DUT to obtain the IPvX packets before sending them back to the other interface of the tester. Fig. 4 exhibits the dual DUT test setup. However, due to the asymmetries in the behavior of both DUTs, which could cause a bottleneck in the benchmarking process, RFC 8219 [9] also recommends testing each DUT separately using the single DUT test setup.

Additionally, some other settings for the test setups are recommended by RFC 8219 [9] to be used in the benchmarking tests. These settings are related to the use of different frame sizes, various proportions of background (i.e., native IPv6) and foreground (i.e., encapsulated) bidirectional traffic, a specific list of IP addresses, and the User Datagram Protocol (UDP) as the transport layer protocol.

Four primary performance measurement procedures are recommended by RFC 8219 [9] during the benchmarking process. A short description of each one is as follows:

1) *Throughput*: The determination of the maximum frame rate at which no frame loss occurs.
2) *Latency*: The difference between the time value of entirely sending the test frame and the time value of receiving it. A subset of the test stream should be tagged for latency calculation, which can be achieved by recording two timestamps: one at the completion of sending the tagged frame and another at receiving it.
3) *Packet Delay Variation (PDV)*: The difference between the 99.9th percentile and the minimum values of the one-way delay for the stream. Unlike the latency measurement, this measurement should be obtained based on the two timestamps of all test frames in the stream.
4) *Frame Loss Rate (FLR)*: The percentage of lost frames in the test stream. It can be calculated as in (1):

$$FLR = \left(\frac{n_{sent} - n_{received}}{n_{sent}}\right) \times 100\% \tag{1}$$

where n_{sent} is the number of sent frames, and $n_{received}$ is the number of received frames.

As MAP-E is considered an encapsulation technology according to the classification of RFC 8219 [9], it could be benchmarked based on the dual DUT test setup. However, this paper focuses on benchmarking the BR device only using the single DUT test setup for two reasons. First, the BR device is considered the focal point of the scalability of MAP-E technology. Hence, its benchmarking is of high importance. Second, measuring the performance and scalability of both devices, the CE and the BR, together could give unpredictable and inaccurate benchmarking results due to the stateful behavior of the CE device, as explained earlier in section I.

IV. RELATED WORK

This section surveys existing research efforts related to benchmarking and performance analysis of IPv6 transition technologies, particularly MAP-E, discusses their findings, and argues their limitations.

Georgescu et al. [18] proposed an IPv6 Network Evaluation Testbed (IPv6NET), a test tool to measure how feasible the examined IPv6 transition technology is for deployment by running a series of scenario-based network situations. As a case study for their research, this article selected one of the enterprise network scenarios presented by the IETF in [19]. The scenario targets enterprises deploying one of the IPv4aaS technologies. Therefore, the authors selected two translation-based technologies, 464XLAT [2] and MAP-T [6], and two encapsulation-based technologies, MAP-E [5] and DS-Lite [3], to implement the scenario.

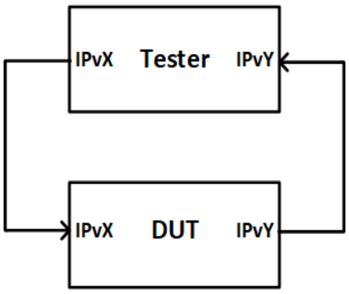


FIGURE 3. Single DUT test setup [9].

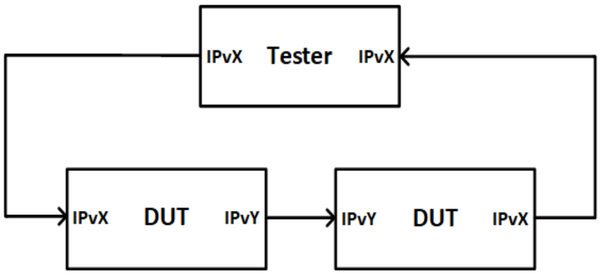


FIGURE 4. Dual DUT test setup [9].

On the other hand, they run the free software *Asamap/Vyatta* Distribution [20] to serve as a technology implementation for the four selected technologies. The Distributed Internet Traffic Generator (D-ITG) [21] was utilized to generate traffic in the experiments.

As a methodology for the experiments, IPv6NET was used in two environments: closed, for obtaining network performance data, and open, for obtaining operational capabilities data. The round-trip delay, jitter, throughput, and packet loss metrics were used to quantify the results of the closed environment. In contrast, an operational capability indicator was proposed for the open environment experiments. The proposed indicator assesses the appropriateness of the tested technology for the given environment and its ability to overcome operational problems. For this purpose, three distinct metrics were defined: *configuration capability*, *troubleshooting capability*, and *applications capability*. The first one measures the capability of the tested implementation in terms of contextual configuration or reconfiguration. The second one measures the capability of the tested implementation to identify and isolate faults. The third one measures the capability of the network device to ensure compatibility with popular end-user network protocols.

The experimental results of IPv6NET showed that, in general, the performance of MAP-E was the best among those of other tested technologies. Furthermore, the translation-based technologies (464XLAT and MAP-T) incurred lower latency than the encapsulation-based technologies (MAP-E and DS-Lite). Conversely, the encapsulation-based technologies achieved higher throughput rates than the translation-based technologies. However, the authors claimed that their results considerably rely on the quality of the implementation used. Therefore, they should be interpreted according to the quality of their chosen implementation, *Asamap*.

It can be said that two limitations should be taken into account regarding the empirical work of this study: first, it relies on the conditions of the scenario used in the test, and there is no benchmarking standard to follow, such as RFC 2455 [7], RFC 5180 [8], or RFC 8219 [9]; Second, it examines the technology as a single entity despite the behavior asymmetries of its devices, the CE and the PE, whose performance should be benchmarked separately to give more valid and accurate results. Section I highlighted some consequences of testing both devices in aggregation.

In [22], Georgescu et al. tested the performance of the same four IPv4aaS technologies against the load scalability. Two different systems were deployed for their experiments: the first involved four servers, two for executing the send and receive functions of the Distributed Internet Traffic Generator (D-ITG) [23] to generate test traffic and two to execute the CE and the Provider Edge (PE) functions of the examined technology. The second system consisted of 31 servers, with one dedicated to the PE function, while the other three functions were run using 10 servers each.

Although the test results of this study revealed interesting information about the impact of increasing traffic load on the overall performance of the examined technology, it lacks exploring how the performance could scale up by increasing the number of active CPU cores of the PE device, as this metric is necessarily important in the benchmarking process because the boost in the processing power of the modern CPUs used in benchmarking mainly resulted from the increase in their number of cores.

G. Lencse et al. proposed an RFC-8219 compliant benchmarking method to measure the performance and scalability of all five most well-known IPv4aaS technologies (including MAP-E) [10]. The method is based on simplifying the dual DUT setup of any examined IPv4aaS technology to the problem of benchmarking a stateful NAT44 gateway; hence, eliminating the need for developing a technology-specific tester. To that end, it aggregates both the CE and PE devices of the technology into a stateful system executing a stateful NAT44. The method was validated by testing the performance and scalability of two Jool implementations, one for 464XLAT technology [24] and another for MAP-T technology [12]. Two metrics were used to measure the performance: the maximum connection establishment rate and the throughput. Two parameters were tuned to measure the scalability: the number of active CPU cores and the number of concurrent connections.

Some of the test results concluded that the Jool implementation of MAP-T scales up much better than that of 464XLAT. In addition, the PE device was the bottleneck in the test experiments of 464XLAT; conversely, the CE device was the bottleneck in the test experiments of MAP-T.

However, the authors referred to the limitations of relying solely on the dual DUT test setup in the benchmarking process in section 7.2. These limitations are summarized in section I of this research paper. The authors emphasized the importance of measuring the performance of each device, the CE and the PE, individually, to overcome the consequences of the hardship of identifying which one is the test bottleneck and specifying the metric results of each one precisely. Moreover, they suggested some potential solutions in this regard for some IPv4aaS technologies in section 7.6. For example, siitperf [25], an RFC 8219-compliant free software tool for benchmarking Stateless IP/ICMP Translation (SIIT) technology (also called stateless NAT64) [26] and for benchmarking stateful NATxy gateways [27], can be utilized to benchmark both the CE device and the PE device of 464XLAT technology (i.e., the customer-side translation CLAT and the Provider-side translation PLAT) separately based on the single DUT test setup of RFC 8219 [9]. Siitperf can also be used to benchmark the two subfunctions of the CE device of MAP-T (i.e., stateful NAT44 and stateless NAT46) based on the single DUT test setup of RFC 8219, in the case where each subfunction is implemented in a separate DUT device. However, MAP rules still needed to benchmark the entire CE performance. In contrast, the PE device of

MAP-T (i.e., the BR) can be benchmarked by Mapperf in the translation mode [13]. On the other hand, benchmarking the CE device or the PE device of MAP-E (i.e., the BR) requires possessing two capabilities: encapsulation/decapsulation of test packets, and adherence to MAP rules. However, siitperf is not designed to have either one of them.

The authors are not aware of any testing tool for benchmarking the PE device of MAP-E. This paper introduces the encapsulation plugin added to Mapperf, which already adheres to MAP rules, to accomplish this task efficiently, and validates it using comprehensive empirical work.

The research effort accomplished throughout this paper, including the MAP-E BR tester and its related analytical methodology, targets a significant gap in benchmarking, no other testing tool has filled in the literature. It is characterized by following a standardized methodology, RFC 8219, for a more efficient and productive benchmarking process.

V. MAPPERF AND THE NEW MAP-E EXTENSION

"*Mapperf*," formerly "*Maptperf*," is an RFC 8219-compliant tester initially developed by the authors to benchmark the MAP-T BR devices. This section introduces a new extension of this tester for benchmarking the MAP-E BR devices.

A. MAPPERF IN A NUTSHELL

The Mapperf tester consists of three main test binaries implemented in C++ with the help of the Intel Data Plane Development Kit (DPDK) [28] API functions. The test binaries are Mapperf-tp, used to measure throughput and the FLR; Mapperf-lat, used to measure latency; and Mapperf-pdv, used to measure PDV. They are supplied by two types of parameters: fixed, whose value does not change during the test experiment and are provided in a configuration file; and varying, whose value could be tuned during the test experiment to reflect different testing conditions, and are provided as command line arguments. Both types of parameters are briefly described in Appendices A and B, respectively. Examples of the configuration file are also provided in Appendices C and D.

In addition, Mapperf utilizes several bash shell scripts to prepare the command-line parameters for the test experiment, run the specific test binaries, and collect the measurement results for evaluation. The benchmarking scheme of Mapperf is shown in Fig. 5.

The workflow of the test experiment of any of the test binaries is typically straightforward. The Tester sends a stream of frames at some frame rate and of some frame size in one or two directions called "forward," which is sourced from the IPv6 interface of the Tester and destined to its IPv4 interface, and "reverse," which is sourced from the IPv4 interface of the Tester and destined to its IPv6 interface. The frames must pass through the DUT, which performs either MAP-T translation or MAP-E encapsulation (in the reverse direction) / decapsulation (in the forward direction) to the frames, depending on the operation mode. Moreover, a subset of the frames could be "native IPv6" frames (also called "background" frames), which must be forwarded smoothly and not be translated or encapsulated/decapsulated by the DUT. Different proportions of the native IPv6 and the "to be translated" or "to be encapsulated/decapsulated" (also called "foreground") test traffic must be sent during the test duration.

1) MAPPERF-TP

The Mapperf-tp is called by two bash shell scripts, one to measure the throughput and another to measure the FLR. They pass the necessary parameter settings such as the

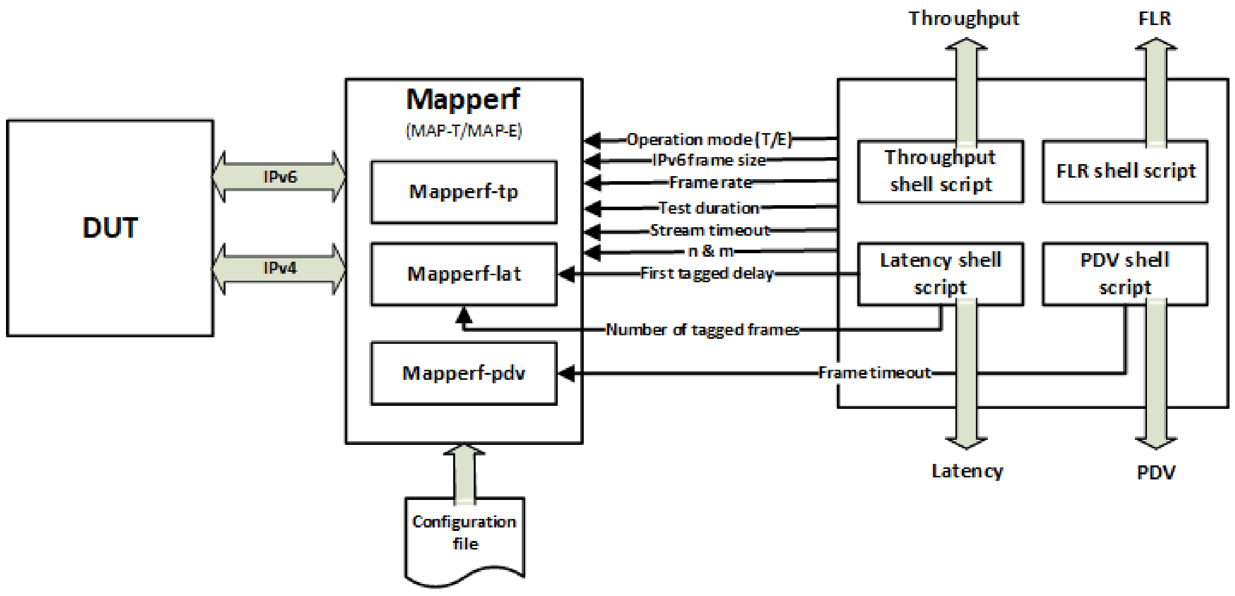


FIGURE 5. The benchmarking scheme of Mapperf.

operation mode (i.e., translation or encapsulation), the frame size, the test duration, the receiving timeout, and proportion of the foreground and background traffic. The throughput shell script considers the test experiment successful if all frames sent from one type of interface (i.e., IPv4 or IPv6) of the Tester were received from its other kind of interface without loss within a predetermined timeout and for all active directions. During the test, the throughput shell script runs a binary search to find the highest frame rate at which the Tester receives all the sent frames without loss. On the other hand, the FLR script computes the FLR after counting the number of lost test frames, which is done at various frame rates. The basic flowchart of Mapperf-tp is shown in Appendix I. The pseudocode of the throughput and FLR shell scripts can be found in Appendices E and F, respectively.

2) MAPPERF-LAT

The Mapperf-lat is called by the latency shell script. In the latency test experiment, all test frames will be sent at the frame rate determined by the throughput test, and only a subset of them will be tagged for recognition in the latency computation. The tagged frames will be identified within the test stream according to the uniform time distribution. Two timestamps will be recorded for each tagged frame, one upon the completion of sending the frame and another upon successful receipt. The difference between these two timestamps represents the frame latency. Next, the Tester will compute two main latency quantities: the Typical Latency (TL), whose value is the median of all latencies, and the Worst-Case Latency (WCL), whose value is the 99.9th percentile of all latencies. The shell script will report these two values for collection. The basic flowchart of Mapperf-lat is shown in Appendix J. The pseudocode of the latency shell script can be found in Appendix G.

3) MAPPERF-PDV

The Mapperf-pdv is mainly called by the PDV shell script. The PDV test experiment will record the sending and receiving timestamps for every test frame, and their one-way latency will be computed and reported accordingly. A unique ID is assigned to the data field of each test frame for identification.

The PDV test experiment can be used either to compute the PDV or to compute the throughput with different criteria, depending on the value of the frame timeout parameter. This parameter specifies the period within which the test frame must be received so as not to be considered a "lost" frame. If its value is 0, the PDV will be computed by subtracting the minimum reported latency from the 99.9th percentile reported latency. In contrast, if the frame timeout is greater than 0, the throughput will be strictly computed because any frame whose one-way latency exceeds the value of the frame timeout will be considered "lost." Therefore, the penalty of using this approach for computing the throughput is more significant, as it consumes more memory and CPU cycles to handle the sending and receiving timestamps for all test frames. The basic flowchart for Mapperf-pdv, which calculates the PDV is shown in Appendix K. The pseudocode of the PDV shell script can be found in Appendix H.

In addition, the MAP-T or MAP-E BR function will be activated on the DUT via a special bash shell script, which installs the necessary MAP rules, sets the appropriate IP addresses for the interfaces, and stores the proper routing and neighboring information.

For more details about the original design and implementation of Mapperf, please refer to an earlier paper by the authors [13].

The source code of Mapperf, along with the shell scripts, configuration file, Makefile, and other files, can be accessed on GitHub [29].

B. THE ADDED ENCAPSULATION CAPABILITY

To enable the Tester to benchmark DUTs running the MAP-E BR implementation, an encapsulation capability is plugged into the Tester, which was initially developed to benchmark only the MAP-T technology. Hence, the Tester can now work in one of two operation modes: translation for MAP-T or encapsulation for MAP-E.

When the Tester runs in the encapsulation mode, it encapsulates the IPv4 test frame with the appropriate IPv6 header according to the MAP rules before sending it through the IPv6 interface in the *forward direction*.[1] The DUT should decapsulate it, and the Tester should then receive the original IPv4 test frame at the end of the tunnel (i.e., its IPv4 interface). Conversely, the Tester sends an IPv4 test frame from its IPv4 interface in the *reverse direction*. The DUT should encapsulate it with the appropriate IPv6 header according to the MAP rules, and the Tester should then receive the original IPv4 test frame encapsulated in the proper IPv6 frame from the other end of the tunnel (i.e., its IPv6 interface).

To validate a successful receive of the test frame, the Tester verifies a special string ("IDENTIFY" in case of the throughput, FLR, or PDV test experiment, or "Identify" in case of the tagged frames during the latency test experiment) in the data field of the IPv4 test frame that will be encapsulated by the IPv6 header either by the Tester itself in the forward direction or the DUT in the reverse direction.

It should also be noted that the size of the IPv6 test frame when the Tester is running in encapsulation mode is 20 bytes larger than when it is running in translation mode, due to the encapsulated IPv4 header.

C. MOST IMPORTANT DESIGN AND IMPLEMENTATION DECISIONS

What follows is a summary of the most critical decisions that were taken during the design and implementation of Mapperf to produce a more robust and efficient testing tool:

1) Buffers of template foreground and background frames were pre-generated, and they can be manipulated

[1]The direction that complies with the arrows in Fig. 3 is called forward direction. The opposite is called reverse direction.

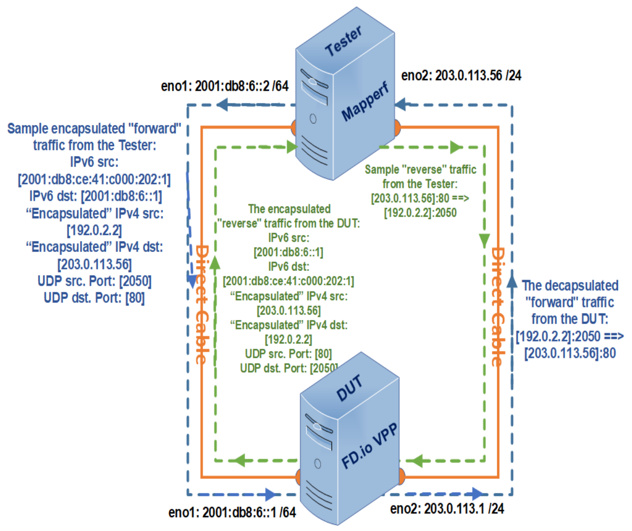

Tester
Mapperf
eno1: 2001:db8:6::2 /64
eno2: 203.0.113.56 /24
Sample encapsulated "forward" traffic from the Tester:
IPv6 src: [2001:db8:ce:41:c000:202:1]
IPv6 dst: [2001:db8:6::1]
"Encapsulated" IPv4 src: [192.0.2.2]
"Encapsulated" IPv4 dst: [203.0.113.56]
UDP src. Port: [2050]
UDP dst. Port: [80]
Sample "reverse" traffic from the Tester:
[203.0.113.56]:80 ==> [192.0.2.2]:2050
The encapsulated "reverse" traffic from the DUT:
IPv6 src: [2001:db8:6::1]
IPv6 dst: [2001:db8:ce:41:c000:202:1]
"Encapsulated" IPv4 src: [203.0.113.56]
"Encapsulated" IPv4 dst: [192.0.2.2]
UDP src. Port: [80]
UDP dst. Port: [2050]
Direct Cable
Direct Cable
The decapsulated "forward" traffic from the DUT:
[192.0.2.2]:2050 ==> [203.0.113.56]:80
DUT
FD.io VPP
eno1: 2001:db8:6::1 /64
eno2: 203.0.113.1 /24

FIGURE 6. Test System 1.

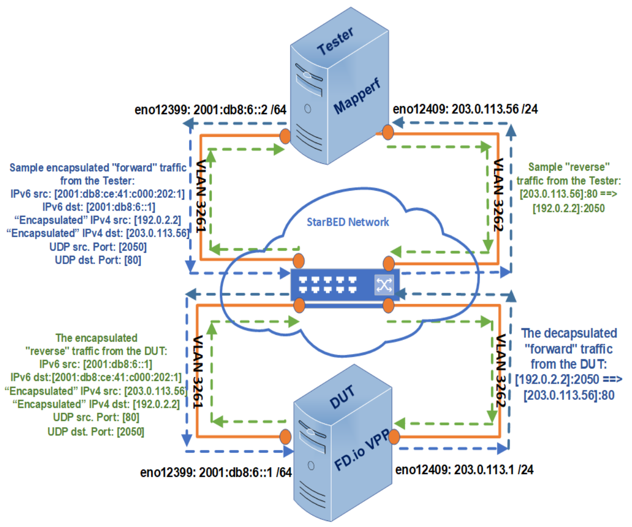

Tester
Mapperf
eno12399: 2001:db8:6::2 /64
eno12409: 203.0.113.56 /24
Sample encapsulated "forward" traffic from the Tester:
IPv6 src: [2001:db8:ce:41:c000:202:1]
IPv6 dst: [2001:db8:6::1]
"Encapsulated" IPv4 src: [192.0.2.2]
"Encapsulated" IPv4 dst: [203.0.113.56]
UDP src. Port: [2050]
UDP dst. Port: [80]
VLAN 3261
VLAN 3262
StarBED Network
Sample "reverse" traffic from the Tester:
[203.0.113.56]:80 ==> [192.0.2.2]:2050
The encapsulated "reverse" traffic from the DUT:
IPv6 src: [2001:db8:6::1]
IPv6 dst:[2001:db8:ce:41:c000:202:1]
"Encapsulated" IPv4 src: [203.0.113.56]
"Encapsulated" IPv4 dst: [192.0.2.2]
UDP src. Port: [80]
UDP dst. Port: [2050]
VLAN 3261
VLAN 3262
The decapsulated "forward" traffic from the DUT:
[192.0.2.2]:2050 ==> [203.0.113.56]:80
DUT
FD.io VPP
eno12399: 2001:db8:6::1 /64
eno12409: 203.0.113.1 /24

FIGURE 7. Test System 2.

(whether for translation or encapsulation) later in the sending cycle to consume less memory and CPU cycles and eventually enhance the performance of the Tester.

2) The Tester should simulate a high number of CEs (to reflect, to some extent, what could happen in the production network). Thus, it creates a CE MAP array for the sender of each direction before sending the test frames. Each array element is a structure of unique CE information consisting of its MAP IPv6 address, public IPv4 address, and PSID. The checksum of the structure fields was also taken into consideration. The elements are pseudorandomly enumerated in the CE array, and the Tester uses them sequentially during the sending cycle of the test frames.
3) The MAP IPv6 address of each CE element is selected from a prepared vector of the pseudorandom enumerations of all possible IPv4 suffix and PSID combinations based on Durstenfeld's random shuffle algorithm [30]. In each iteration, the following combination will be selected according to the 64-bit Mersenne Twister pseudorandom number generator (std::mt19937_64), which was chosen based on the results of Oscar David Arbeláez [31]. Furthermore, this pseudorandom number generator is also used to select the UDP source and destination port numbers of the test frames to be sent.

VI. BENCHMARKING ENVIRONMENT

A. MEASUREMENT SYSTEMS

Two systems were constructed to provide a more accurate evaluation of the scalability and performance of the benchmarked MAP-E BR implementation. The first Test System (TS1) was installed in the Budapest University of Technology and Economics by connecting two 10G network interfaces of a server representing the Tester by two 10G network interfaces of a server representing the DUT via direct cables, as shown in Fig. 6. The second Test System (TS2) was installed using the resources of the NICT StarBED[2] in Japan, connecting two 25G network interfaces of a server representing the Tester to two 25G network interfaces of a server representing the DUT via the StarBED network switch. Each of the connected interface pairs belongs to a dedicated VLAN. TS2 is shown in Fig. 7.

As described in Table 1, which lists the primary specifications of the servers of each test system, the CPU clock frequency of the DUTs of both systems was fixed to ensure stable measurement results. In addition, the CPU cores 2, 4, 6, and 8 of the Tester of TS1 and the CPU cores 4, 8, 48, and 52 of the Tester of TS2 were dedicated to running the sender and receiver threads of both test directions as they belong to the same Non-Uniform Memory Access (NUMA) node that the network interfaces used for sending and receiving the test traffic. Moreover, they were reserved using the isolcpus kernel parameter to prevent other tasks from being scheduled on them.

B. TEST AND TRAFFIC SETUP

The measurement systems were installed according to the guidelines of the single DUT test setup of RFC 8219 [9]. Please refer to section III, which gives details of this test setup. The Tester is responsible for running Mapperf, sending streams of test frames, and reporting the measurement results. In contrast, the DUT is responsible for running the MAP-E BR implementation to be benchmarked. Section V-C gives an overview of this implementation.

Two test directions were established, from the IPv6 interface of the Tester passing through the DUT to the IPv4 interface of the Tester, referred to as "forward," and from the IPv4 interface of the Tester passing through the DUT to the IPv6 interface, referred to as "reverse." During their traversal in the test directions, the test frames must transfer in the

[2]https://starbed.nict.go.jp/en/aboutus/index.html

TABLE 1. Specifications of the test systems.

	Test System 1 (TS1)		Test System 2 (TS2)	
	Tester	**DUT**	**Tester**	**DUT**
Server Type	Dell PowerEdge R720	Dell PowerEdge R730	Dell PowerEdge R650	Dell PowerEdge R650
Number of CPUs	2	2	2	2
Type and speed of CPUs	Intel Xeon E5-2665	Intel Xeon E5-2630 v3	Intel Xeon Gold 6330N	Intel Xeon Gold 6330N
Number of cores per CPU	8	8	28	28
CPU Hyperthreading	Disabled	Disabled	Disabled	Disabled
CPU Turbo Mode	Disabled	Disabled	Enabled	Disabled
CPU clock speed	2.4GHz (fixed)	2.4GHz (fixed) + 1.2GHz (fixed)	0.8-3.4GHz (varying)	2.2GHz (fixed) +0.8GHz (fixed)
Number of NUMA nodes	2	2	4	4
RAM	32GB of 1333MHz DDR3	128GB of 2133MHz DDR4	512GB of 3200MHz DDR4	512GB of 3200MHz DDR4
OS release	Debian 9 (stretch)	Debian 11 (bullseye)	Ubuntu 22.04.3 LTS (Jammy Jellyfish)	Ubuntu 22.04.4 LTS (Jammy Jellyfish)
Kernel version	4.9.0-13-amd64	5.10.0-20-amd64	5.15.0-101-generic	5.15.0-107-generic
DPDK version	16.11.11-1+deb9u2	20.11.9-1~deb11u1	21.11.3-0ubuntu0.22.04.1	21.11.6-0ubuntu0.22.04.1

MAP-E tunnel, whose endpoints are the IPv6 interface of the Tester and its counterpart at the DUT. The IPv6 interfaces are configured within the subnetwork 2001:db8:6::/64, whereas the IPv4 interfaces are configured within the subnetwork 203.0.113.0/24.

Moreover, RFC 8219 recommends using different proportions of foreground and background traffic during the test experiments. However, receiving the background traffic was not possible because the MAP-E implementation at the DUT expects all received traffic to be encapsulated. Therefore, it attempts to decapsulate the native IPv6 traffic, which could ultimately result in dropping the test frames. Therefore, the test experiments were run without background traffic in this research. Later, the authors may investigate the problem more and attempt some solutions.

C. THE TESTED MAP-E IMPLEMENTATION

To the best of the authors' knowledge, there are very few MAP-E BR implementations, and the open-source Vector Packet Processing (VPP) [11] can be considered the most distinguished one. It is the main project of FD.io [32], based on Cisco's VPP technology. That is where the name came from. VPP bypasses the system kernel and handles a vector of packets concurrently in the user space. Thus, it needs some kind of user-space driver, such as the user poll driver of DPDK [28], for its network I/O operations. The used version was 24.02.

Before running the benchmarking experiments, the functionality of VPP was verified at TS2[3] via sending some foreground and background traffic in both directions from the Tester. Fig. 8 exhibits a snapshot of some received and forwarded packets captured at both interfaces of the DUT. It shows how VPP properly encapsulates packets in the reverse direction and decapsulates them in the forward direction. It also demonstrates the valid function of the Tester in terms of the adherence to the MAP rules and the implementation of MAP-E encapsulation. The BMR information used is (Rule IPv6 prefix = 2001:db8:ce::/51, Rule IPv4 prefix = 192.0.2.0/24, EA-length = 13).

As can be noticed, packet 1 is an IPv6 test packet encapsulating IPv4 content sent from the Tester in the forward direction. The source IPv6 address (**2001:db8:ce:17a9:0:c000:2bd:9)** represents the MAP address of the simulated CE at the Tester. In contrast, the destination IPv6 address (**2001:db8:6::1**) represents the corresponding BR address (i.e., the address of the IPv6 interface of the DUT). The encapsulated source IPv4 address (**192.0.2.189**) represents the public IPv4 address of the simulated CE, which is shared with other CEs; however, the assigned port set will uniquely identify the simulated CE. This address is also embedded as the four octets valued as (**c000:2bd)** in the MAP address, whereas the PSID is embedded as the last two octets valued as (9). The rule IPv6 prefix is **2001:db8:ce:/51**, and the EA-bits are **17a9** represented in 13 bits (IPv4 suffix (first 8 bits) + PSID (last 5 bits)).

VPP successfully decapsulated packet 1 and sent the embedded IPv4 packet as packet 2 in the forward direction towards the Tester.

Packet 3 is an IPv4 test packet sent from the Tester in the reverse direction. The source IPv4 address (**203.0.113.56**) represents the IPv4 server simulated by the Tester, whereas the destination IPv4 address (**192.0.2.80**) represents the public IPv4 address of the simulated CE at the Tester.

VPP successfully encapsulated packet 3 content in an IPv6 packet, which is packet 4, and sent it in the reverse direction towards the Tester. It successfully adhered to MAP rules, as seen from the formation of the MAP address to represent the destination IPv6 address (**2001:db8:ce:a11:0:c000:250:11**).

[3]The same version of VPP was installed at TS1; Thus, there was no need to test its functionality there, too.

```
1 05.300760 IP6 2001:db8:ce:17a9:0:c000:2bd:9 > 2001:db8:6::1: IP 192.0.2.189.19722 > 203.0.113.56.46521
2 05.300785 IP 192.0.2.189.19722 > 203.0.113.56.46521
3 05.300789 IP 203.0.113.56.41999 > 192.0.2.80.35336
4 05.300801 IP6 2001:db8:6::1 > 2001:db8:ce:a11:0:c000:250:11: IP 203.0.113.56.41999 > 192.0.2.80.35336
5 05.799651 IP6 2001:db8:ce:17a9:0:c000:2bd:9 > 2001:db8:6::1: IP 192.0.2.189.19313 > 203.0.113.56.19912
6 05.799651 IP 203.0.113.56.10408 > 192.0.2.80.36536
7 05.799654 IP 192.0.2.189.19313 > 203.0.113.56.19912
8 05.799654 IP6 2001:db8:6::1 > 2001:db8:ce:a11:0:c000:250:11: IP 203.0.113.56.10408 > 192.0.2.80.36536
9 06.298549 IP6 2001:db8:ce:607:0:c000:230:7 > 2001:db8:6::1: IP 192.0.2.48.15196 > 203.0.113.56.21046
10 06.298550 IP 203.0.113.56.65174 > 192.0.2.41.41811
11 06.298553 IP 192.0.2.48.15196 > 203.0.113.56.21046
12 06.298553 IP6 2001:db8:6::1 > 2001:db8:ce:534:0:c000:229:14: IP 203.0.113.56.65174 > 192.0.2.41.41811
.
.
.
13 52.118679 IP6 2001:db8:6::2.23457 > 2001:db8:7::2.13707
14 52.118699 IP6 2001:db8:7::2.32172 > 2001:db8:6::2.36445
15 52.617568 IP6 2001:db8:6::2.20916 > 2001:db8:7::2.41725
16 52.617568 IP6 2001:db8:7::2.8786 > 2001:db8:6::2.5099
```

FIGURE 8. Sample VPP capture of foreground and background packets at both interfaces of the DUT.

Regarding the investigation of port number selection, the Tester pseudo-randomly selected PSID 9 for packet 1. To specify the range of port numbers for this port set, we know that the EA-length is 13 bits and the IPv4 suffix length is 8 bits. This leaves 5 bits representing the PSID length. So, the number of port sets is $2^5 = 32$, each consisting of $2^{11} = 2048$ ports. Therefore, the PSID 9 in hexadecimal (9 in decimal) has a port range that starts from port (9 * 2048 = 18432) and ends at port (10 * 2048 -1 = 20479). Thus, the source port 19722 in packet 1 is pseudo-randomly selected within the intended port range. This validates the function of the Tester in this regard. However, the destination port 46521 is pseudo-randomly selected from the wide range (1 – 49151) as recommended by RFC 4814 [33].

In contrast, VPP sets the PSID 11 in the destination IPv6 address of packet 4 (the MAP address **2001:db8:ce:a11:0:c000:250:11**) based on the destination port **35336** of the received packet 3. That is, the PSID 11 in hexadecimal (17 in decimal) has a port range that starts from port (17 * 2048 = 34816) and ends at port (18 * 2048 -1= 36863). Thus, the PSID 11 set by VPP in packet 4 is proper because the destination port 35336 in packet 3 is within its intended port range. This validates the function of VPP in this regard. However, the source port 41999 is pseudo-randomly selected from the wide range (1024 – 65535) as recommended by RFC 4814 [33].

The last four packets are background packets sent from the Tester in both directions. The Tester selects the IPv6 addresses as inserted in the configuration file, whereas the source port addresses are pseudo-randomly selected from the wide range (1024 – 65535), and the destination port addresses are pseudo-randomly selected from the wide range (1-49151) as recommended by RFC 4814 [33]. However, VPP had some problems forwarding the background packets as it expected them to encapsulate IPv4 packet contents; consequently, VPP discarded them accordingly.

VII. THE BENCHMARKING TESTS AND RESULTS

The primary test parameter settings common to the benchmarking experiments are listed in Table 2.

All test experiments were run 20 times to get more reliable results, and the median was considered the summarizing function. The 1st percentile, the 99th percentile, and the dispersion values were used to illustrate the stability of the results (i.e., how consistent or scattered they are). The dispersion value is calculated as in (2).

$$Dispersion = \frac{99^{th}percentile - 1^{st}percentile}{median} \times 100\% \quad (2)$$

The median values were represented as bars in the bar charts of the results, whereas the 1st percentile and 99th percentile values were represented as error bars.

Mapperf reports the throughput results as the number of frames per second per direction, in contrast to commercial network performance testers, which typically report the total number of frames per second. Hence, the Mapperf measurement results using bidirectional traffic should be doubled to know the equivalent results (i.e., the total number of forwarded frames per second).

A. THE PERFORMANCE OF MAPPERF IN THE ENCAPSULATION MODE

Before starting with the benchmarking experiments of the MAP-E BR implementation, the capacity of Mapperf in the encapsulation mode had to be identified to determine when the Tester could form a bottleneck during the test. In other words, the maximum frame rate that Mapperf can satisfy without any frame loss. For this purpose, two loopback tests

TABLE 2. Primary parameter settings of the benchmarking experiments.

Test parameter	Value
Experiment duration	Each run was 60 seconds long for all test types except for the latency test, which was 120 seconds.
Sleep duration	10 seconds between any two experiments, to prepare the test systems for the next experiment.
Traffic direction	Bidirectional for all test experiments, except for the throughput tests with unidirectional traffic.
IPv4 frame size	64 bytes (i.e., IPv6 frame size is 40 bytes more due to the additional IPv6 header), except for the throughput tests using different frame sizes.
Traffic type	100% foreground traffic. The background traffic was inapplicable because the MAP-E BR implementation attempts to manipulate each received IPv6 frame as encapsulated.
Number of active CPU cores at the DUT	For VPP, a single worker thread was deployed; so only one CPU core was used. This was core 2 at TS1 and core 4 at TS2. The exception is the throughput tests using a different number of worker threads.
CPU clock frequency at the DUT	For the scalability tests, the possible minimum clock frequency was also used at the DUT (1.2GHz at TS1 and 800MHz at TS2). For all other tests, the nominal clock frequency was used at the DUT.
Number of simulated CEs	1000 CEs, except for the throughput tests using a different number of CEs.

were run: one, denoted as LT1, is by connecting the two 10G network interfaces of an equivalent Tester server of TS1 via a direct cable, as shown in Fig. 9. The other, denoted as LT2, is by connecting the two 25G network interfaces of the Tester server of TS2 via the network switch using the same VLAN, as shown in Fig. 10.

In both loopback tests, Mapperf can work without a DUT. It can recognize the received frames, whether they are IPv4 test frames or IPv6 containing IPv4 (i.e., encapsulated) test frames, and then behave accordingly.

The results of this test can help specify the appropriate value of some testing parameters, such as the number of worker threads that VPP can utilize. Hence, this could help avoid a situation where the reported benchmarking results do not accurately reflect what the DUT can achieve for the given test parameters due to the limited capacity of Mapperf.

The results of LT1 and LT2 are listed in Table 3. The tests were run using 100% foreground bidirectional traffic with an IPv4 frame size of 64 bytes. It is essential to note that frame rates should be considered *per direction*. As they were achieved using bidirectional traffic, the total number of frames per second handled by Mapperf is double the values shown in Table 3.

The results showed that Mapperf can achieve a high throughput rate of approximately 6.54 Mfps in LT1 and 9.15 Mfps in LT2. It is noted that not the median, but the first percentile values were taken into consideration, and the authors recommend that the Tester can only be used up to somewhat (e.g., 5-10%) lower frame rates than the first percentiles to indeed prevent the Tester from being a bottleneck.

B. THROUGHPUT TEST RESULTS

Four types of throughput tests were conducted to evaluate the performance of the VPP MAP-E BR: using a different

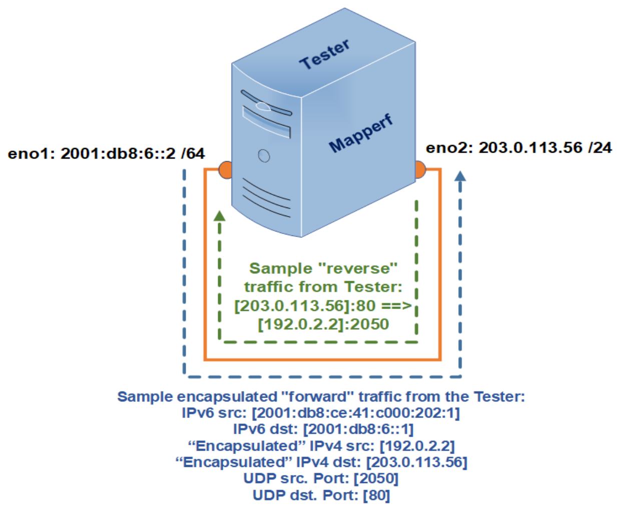

Tester
Mapperf
eno1: 2001:db8:6::2 /64
eno2: 203.0.113.56 /24
Sample "reverse" traffic from Tester: [203.0.113.56]:80 ==> [192.0.2.2]:2050
Sample encapsulated "forward" traffic from the Tester:
IPv6 src: [2001:db8:ce:41:c000:202:1]
IPv6 dst: [2001:db8:6::1]
"Encapsulated" IPv4 src: [192.0.2.2]
"Encapsulated" IPv4 dst: [203.0.113.56]
UDP src. Port: [2050]
UDP dst. Port: [80]

FIGURE 9. Loopback Test 1 (LT1).

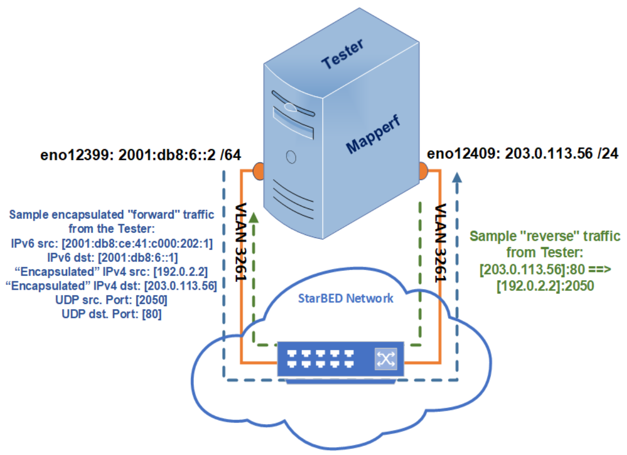

Tester
Mapperf
eno12399: 2001:db8:6::2 /64
eno12409: 203.0.113.56 /24
VLAN 3261
VLAN 3261
Sample encapsulated "forward" traffic from the Tester:
IPv6 src: [2001:db8:ce:41:c000:202:1]
IPv6 dst: [2001:db8:6::1]
"Encapsulated" IPv4 src: [192.0.2.2]
"Encapsulated" IPv4 dst: [203.0.113.56]
UDP src. Port: [2050]
UDP dst. Port: [80]
Sample "reverse" traffic from Tester: [203.0.113.56]:80 ==> [192.0.2.2]:2050
StarBED Network

FIGURE 10. Loopback Test 2 (LT2).

TABLE 3. Maximum frame rate achieved by Mapperf using the encapsulation mode in the two loopback tests.

Loopback Test	1st Percentile (fps)	99th Percentile (fps)	Median (fps)	Dispersion (%)
LT1	6,543,113	6,552,682	6,550,276	0.14608
LT2	9,150,810	9,201,886	9,200,498	0.55513

number of active CPU cores at the DUT, using different frame sizes, employing a different number of simulated CEs, and utilizing unidirectional traffic. The latter is optional in RFC 8219 [9]. The condition to pass these tests is to receive all sent frames without any frame loss.

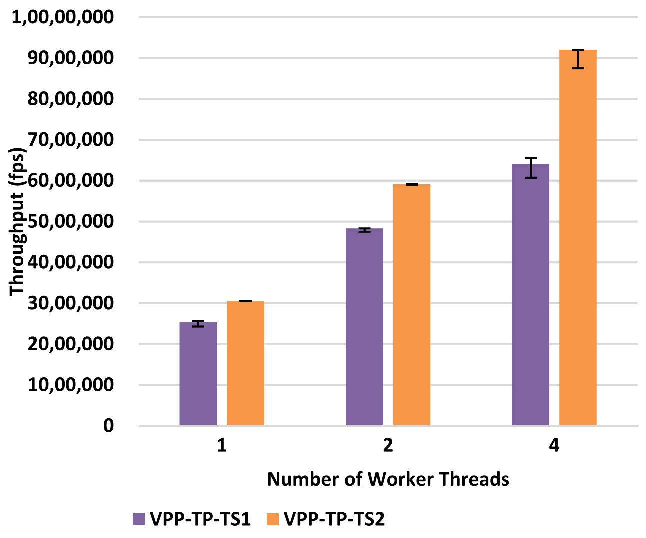
An additional test type using different proportions of foreground and background traffic could not be performed due to the behavior of VPP in the encapsulation mode, which expects every received frame to be encapsulated, whereas the background frames are not.

1) TESTS WITH A DIFFERENT NUMBER OF ACTIVE CPU CORES

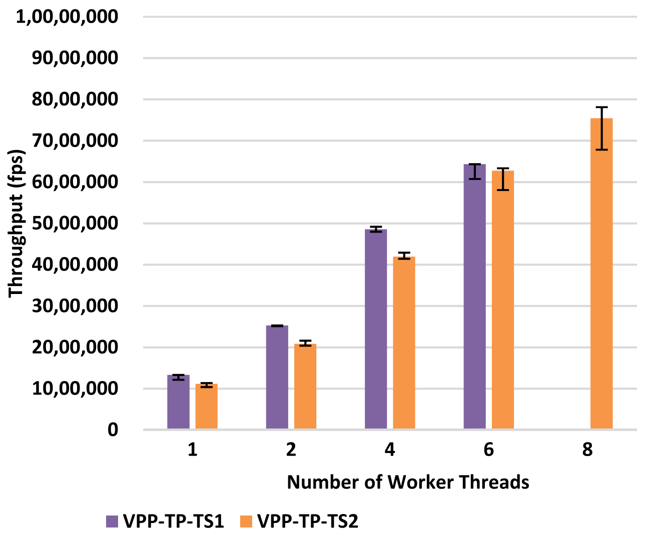
These tests help investigate the scalability of VPP in relation to the number of active CPU cores at the DUT. Fig. 11.a shows how the performance of VPP scales at both test systems using an increasing number of worker threads starting from 1 up to 4 worker threads (besides the main thread of VPP, which usually deploys core 0). Each worker thread of TS1 used a single active CPU core of a fixed frequency of 2.4 GHz, whereas each worker thread of TS2 used a single active CPU core of a fixed frequency of 2.2 GHz. All utilized CPU cores in both systems belong to the same NUMA node (NUMA node 0).

The results proved that the performance of VPP scaled up effectively each time the number of worker threads doubled, and high throughput rates were achieved even when using a low number of worker threads. It should be noted that the recorded median throughput rate with four worker threads at TS1 was 6,405,503fps, which was so close to the capacity of Mapperf at TS1 (6,54 Mfps). In contrast, the recorded median throughput rate with four worker threads at TS2 was 9,198,715fps, which exceeded the capacity of Mapperf at TS2 (9.15 Mfps); thus, the first one is unreliable. The second one does not accurately reflect the actual throughput rate that can be achieved by VPP, as it exceeds what Mapperf can handle at TS2. That is, Mapperf formed a bottleneck in this case. (Please refer to Section VI-A to determine when Mapperf can create a bottleneck during the test.)

To address this issue, the test was repeated with the CPU clock frequency of the DUTs set to their possible lowest values, 1.2 GHz and 0.8 GHz for TS1 and TS2, respectively. The results are shown in Fig. 11.b. The throughput rate for eight worker threads used at TS1 is not included because no more than seven worker threads could be utilized in NUMA node 0 (the main thread of VPP used core 0). Nevertheless, the throughput rate at TS1 when six worker threads were used is so close to the maximum capacity of the Tester. The figure shows nearly linear scalability from 1 to 4 workers for TS1 and from 1 to 6 workers for TS2. The performance of the final tests (with six workers in TS1 and eight workers in TS2) is lower than one could expect. In both cases, it can be attributed to the fact that the maximum performance of the Tester was closely approached, and thus, the results are not reliable. Especially in the case of TS2 using eight workers, the dispersion of the results is too high (more than 10% of the median), which can be explained by the fact that some of



(a) the CPU frequency was set to 2.4 GHz at TS1 and 2.2 GHz at TS2 (The results of the tests with 4 worker threads on both test systems are limited by the Tester performance.)



(b) the CPU frequency was set to 1.2 GHz at TS1 and 0.8 GHz at TS2 (The results with 6 workers on TS1 and with 8 workers on TS2 are limited by the Tester performance.)

FIGURE 11. Throughput of VPP using different number of worker threads at the DUT.

the tests at rates around 7Mfps failed due to the seemingly random loss of a low amount of test frames (e.g., 0.01%) according to the measurement log file.

2) TESTS WITH DIFFERENT FRAME SIZES

In these tests, different frame sizes were used to comply with the guidelines of section V-A1 of RFC 8219 [1], [9]. Fig. 12 shows the results of these tests for VPP deploying only one worker thread at both test systems. The frame size represents the size of the test frames carrying IPv6 datagrams. The IPv4 test frames are 40 bytes shorter (i.e., the IPv6 test frames have an additional 40 bytes for the IPv6 header).

The results are quite stable, as the error bars are relatively small for TS1, and they are barely visible for those of TS2. The throughput of VPP at both systems is almost constant for small frame sizes. However, a sharp decreasing tendency started from the frame size of 552 bytes at TS1 and 1064 bytes at TS2. This can be attributed to reaching the maximum transmission capacity of the 10 GbE network interfaces of TS1 and the 25 GbE network interfaces of TS2. That is, they formed a bottleneck during the test. Table 4 summarizes the approximate maximum frame rate that can be achieved by these network interfaces, expressed in frames per second (fps), and is calculated based on the formula in Appendix A.1 of RFC 5180 [8].

Furthermore, TS2 yielded better performance outcomes than TS1 due to the greater hardware capabilities. The small frame sizes reveal a throughput increase of approximately 18% at TS2 compared to TS1 due to the higher single-core performance of its CPU. However, starting from the frame size of 552 bytes, the difference was not determined by the CPU performance, but rather, the network interfaces of TS1 began to form a bottleneck in the benchmarking test and could not handle higher frame rates.

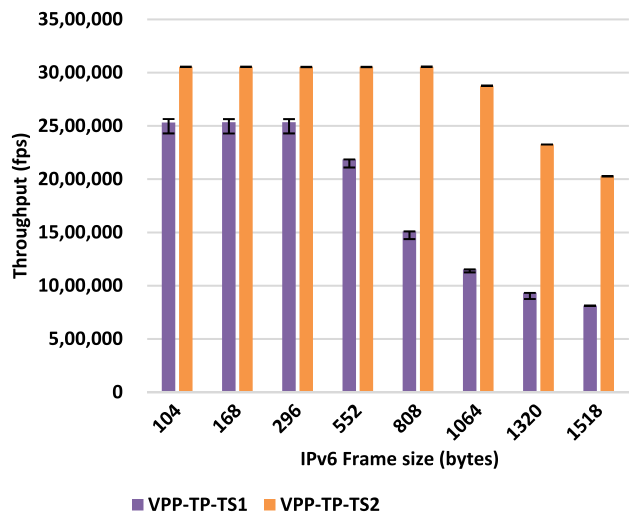


FIGURE 12. Throughput of VPP using different frame sizes (the IPv4 frame sizes are always 40 bytes less).

TABLE 4. Maximum frame rate for the 10GbE NIC of TS1 and 25GbE NIC of TS2.

IPv6 Frame Size (bytes)	TS1 Maximum Frame Rate (fps)	TS2 Maximum Frame Rate (fps)
104	10,080,645	25,201,612
168	6,648,936	16,622,340
296	3,955,696	9,889,240
552	2,185,314	5,463,286
808	1,509,661	3,774,154
1064	1,153,136	2,882,841
1320	932,835	2,332,089
1518	812,743	2,031,859

3) TESTS WITH A DIFFERENT NUMBER OF SERVED CES

In these tests, the Tester simulated a high number of CEs, starting from one million, and the number was doubled in each subsequent experiment. As shown in Fig. 13, the performance of VPP was not affected by the number of served CEs. This is justifiable as the BR followed specific MAP rules to translate the test frames regardless of the MAP address of the simulated CE, which also explains the high scalability of MAP-E technology.

As stated earlier, the throughput of VPP at TS2 is higher than that at TS1 due to the difference in the hardware capabilities of the two systems.

A small difference can be observed between the results shown in the first two columns of Fig. 12 (throughput measured with 104/64 bytes frames and using 1000 CEs) and in the first two columns of Fig. 13 (throughput measured also with 104/64 bytes frames but using 1000000 CEs). It can be explained by the fact that the difference between their test conditions is not only the three order of magnitude in the number of simulated CEs (1000 vs 1000000), but also the size of the PSID. In the first case, the PSID was set to 13,

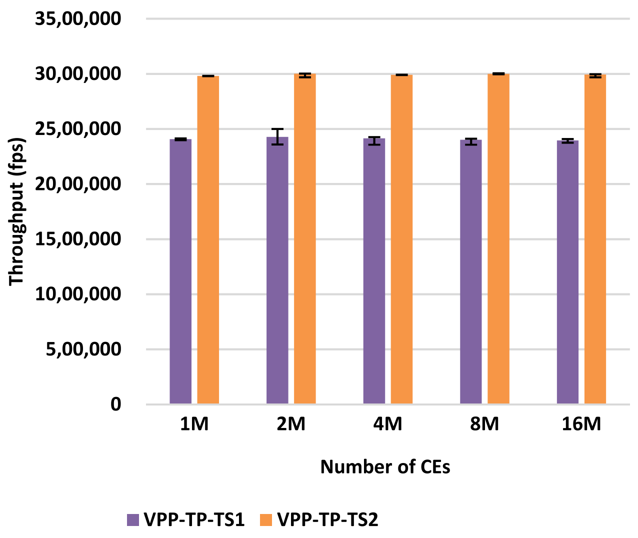


FIGURE 13. Throughput of VPP using different number of served CEs.

and only one thousand CEs will be pseudorandomly selected from the total number of CEs, which is 2^{13} =8192 alternatives. In contrast, in the second case, the PSID was set to 20, and approximately all the CE alternatives (one million) will be covered.

4) TESTS WITH UNIDIRECTIONAL/BIDIRECTIONAL TRAFFIC

In addition to using bidirectional traffic in all test experiments, these tests aimed to obtain fine-grained throughput values for each direction individually. As shown in Fig. 14, the reverse direction in both systems yields lower performance outcomes than the forward direction. This can be explained by the fact that the DUT required more work in the reverse direction due to the encapsulation overhead.

On the other hand, even though Ethernet transfers the test frames in full duplex, the other system resources, such as the CPU core executing the worker thread, the memory, and the PCI Express bus, were shared. Consequently, throughput significantly decreased when using bidirectional traffic because the same CPU core had to process the traffic flowing in both directions.

C. FLR TEST RESULTS

RFC 8219 [9] recommends starting the test with the maximum possible frame rate of the media and decreasing it by no more than 10% in each consecutive step. However, in practice, the achieved frame rate is much lower. Thus, the test experiments were conducted at typical frame rates to yield more meaningful outcomes.

The FLR results of VPP at both systems are shown in Fig. 15. The test started sending test frames at a frame rate of 2 million fps and increased by 200,000 fps with each consecutive experiment run, stopping after examining a frame rate of 4 million fps.

The results were rational, as it was expected that more frame loss could occur gradually when higher frame rates were utilized during the test. The difference between the FLR of both systems can be attributed to the difference in their throughput rates achieved during the earlier tests.

D. LATENCY TEST RESULTS

The latency test results for VPP at both systems are listed in Table 5. The frame rate used in the test was the median frame rate achieved during the throughput test, which employed the same parameter settings (e.g., IPv4 frame size of 64, bidirectional traffic, etc.). The number of tagged frames was set to 50,000. The tagged frames were selected according to the uniform time distribution after 60 seconds from the beginning of the test duration, as required by RFC 8219.

Interestingly, the results showed a slight decrease in the latency values at TS2 compared to TS1 for both directions. In contrast to TS1, TS2 deploys additional network appliances (e.g., the network switch) that could increase the delay when transferring the test frame. However, their impact was less effective compared to the significant gain that VPP obtained from the packet vector length at TS2. That is, VPP processes a vector of packets simultaneously. After checking the vector length at both systems, it was much shorter at TS2 than at TS1 (44 packets at TS2 vs 241 packets at TS1. Each packet in the vector must wait until all other packets arrive before they can be processed in parallel. Hence, the wait time at TS2 could be shorter than that at TS1. This accordingly affects the latency values, in general, on both systems.

E. PDV TEST RESULTS

The PDV test results for VPP at both systems are listed in Table 6. Similarly, the frame rate used in this test was the median frame rate reported by the throughput test, which used the same parameter settings. As computing the PDV

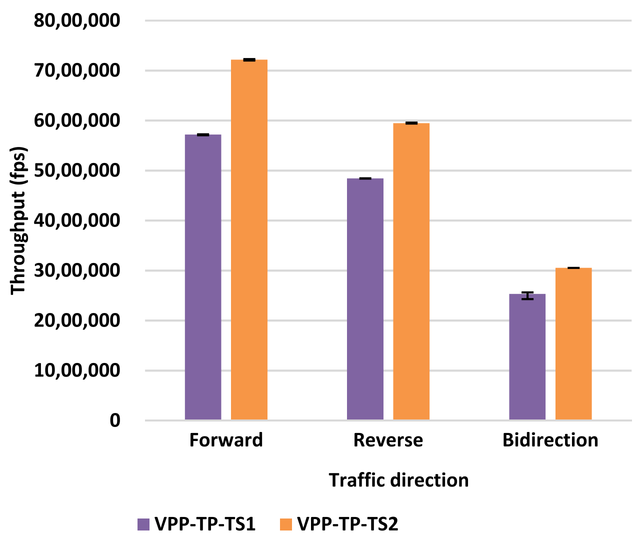


FIGURE 14. Throughput of VPP in the unidirectional tests (rates are to be interpreted as "per direction," the total number of packets forwarded is double the displayed numbers in the case of bidirectional traffic).

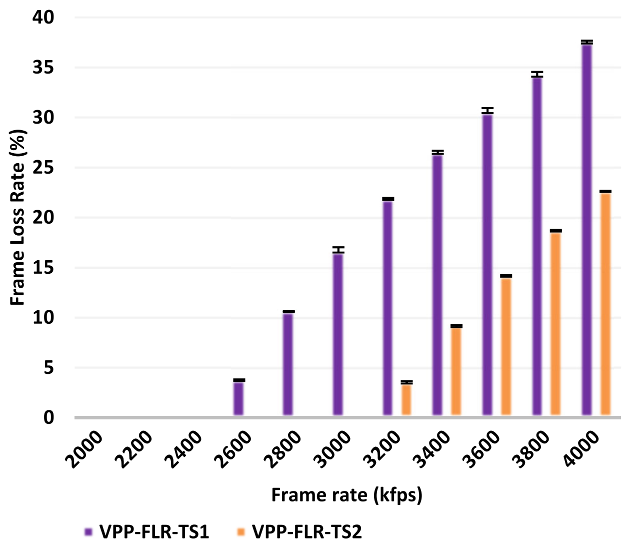


FIGURE 15. FLR of VPP.

TABLE 5. Latency of VPP using a frame rate of 2,531,239 fps at TS1 and 3,054,741 fps at TS2.

	VPP-LAT-TS1				VPP-LAT-TS2			
	Forward TL	Forward WCL	Reverse TL	Reverse WCL	Forward TL	Forward WCL	Reverse TL	Reverse WCL
1st Percentile (ms)	0.13833	0.31918	0.14256	0.33307	0.12452	0.23984	0.13395	0.24464
99th Percentile (ms)	0.14607	0.43853	0.14712	0.43472	0.14254	0.27860	0.14335	0.28945
Median (ms)	0.14315	0.35054	0.14534	0.36688	0.12995	0.26382	0.13572	0.27301
Dispersion (%)	5.40	34.04	3.13	27.7	13.86	14.68	6.92	16.41

TABLE 6. PDV of VPP using a frame rate of 2,531,239 fps at TS1 and 3,054,741 fps at TS2.

	VPP-PDV-TS1		VPP-PDV-TS2	
	Forward PDV	Reverse PDV	Forward PDV	Reverse PDV
1st Percentile (ms)	0.32114	0.34181	0.22200	0.22708
99th Percentile (ms)	0.38477	0.38480	0.26680	0.26335
Median (ms)	0.34743	0.36587	0.24289	0.25134
Dispersion (%)	18.31	11.75	18.44	14.43

originally relies on the one-way latency of every sent frame, the PDV test resulted in an outcome that relatively matches that of the latency test. In other words, the PDV at TS2 is significantly smaller compared to TS1 (for both directions) for the same justifications mentioned in the previous section.

VIII. FUTURE WORK

Although the new MAP-E capability of Mapperf was validated through a sufficient number of diverse benchmarking experiments that reflect various testing conditions, the authors believe that several additional test experiments should be conducted after enhancing the performance of the Tester. For example:

1) In some benchmarking measurements, the network interfaces (i.e., 10GbE at TS1 and 25GbE at TS2) prevented testing the actual packet forwarding performance of the given MAP-E BR implementation using larger frame sizes, as the interfaces formed a bottleneck because they were not able to handle higher rates than their maximum capacity, as described in Table 4. Hence, using more efficient network interfaces can help overcome this obstacle.
2) The loopback tests described in Section VI-A effectively demonstrated the performance of the Tester in benchmarking MAP-E BR devices. However, the code logic of the Tester could be optimized further (e.g., enhancing sending loops, improving modularity, etc.) to enable handling higher throughput rates and avoid situations when the Tester creates a bottleneck during the tests.
3) Another dimension of testing the scalability of the MAP-E BR implementations could be integrated into the Tester by adding an update that enables the manipulation of multiple MAP domains. The current version of Mapperf can test MAP-E BR implementations with a single MAP domain only.
4) As both MAP-T and MAP-E are stateless technologies that exhibit high levels of scalability, deploying multiple sender and receiver threads for each direction at the Tester could be promising to enhance the scalable capabilities of Mapperf without causing significant interference with highly efficient MAP-T/E implementations.

Finally, it can be said that Mapperf (with its two operational modes, encapsulation and translation) can also serve as a base model for developing other types of benchmarking testers that can measure the performance and scalability of other IPv6 transition technologies. For example, it could be relatively easy to extend it to benchmark the Lightweight 4over6 technology [4].

IX. CONCLUSION

This paper presented a comprehensive analysis of the scalability and performance of the BR device of MAP-E IPv6 transition technology. For this purpose, an encapsulation capability was developed and plugged into "Maptperf," an RFC 8219 [9] compliant tester initially designed and implemented by the authors to benchmark MAP-T BR devices [13]. Hence, the tester is renamed "Mapperf." The added MAP-E capability deploys the single DUT test setup of RFC 8219 [9] to accomplish its benchmarking measurements. On the other hand, a popular free software, VPP, was utilized as the MAP-E BR implementation at the DUT.

To prove the functionality and efficiency of the developed plugin and to analyze the scalability and performance of VPP, a high number of various testing experiments were conducted using two different test systems, TS1 and TS2. The combination of scalability experiments involved measurements using a different number of active CPU cores at the DUT and using a different number of simulated served CEs. In contrast, the combination of performance experiments involved measurements of the throughput using different frame sizes and using unidirectional traffic, as well as the FLR, latency, and PDV measurements.

The results showed that VPP scales almost linearly at both systems when the number of active CPU cores at the DUT is incremented. In contrast, its performance was not affected by doubling the number of simulated CEs, starting from one million up to 16-fold, which highlights its high scalability. On the other hand, the performance of VPP was relatively stable despite the increase in test frame sizes. However, the unidirectional tests proved lower performance rates for the reverse direction than for the forward direction due to the encapsulation overhead, and considerable performance degradation when bidirectional traffic was utilized. Moreover, the FLR results were expected as the higher frame rates used, the more significant frame loss could occur. The latency and PDV values at TS2 were lower than those at TS1 due to the shorter vector sizes deployed by VPP of TS2 for its manipulated test packets, despite using additional network appliances such as the network switch.

APPENDIX A
DESCRIPTION OF MAPPERF CONFIGURATION FILE PARAMETERS

See Table 7.

APPENDIX B
DESCRIPTION AND USAGE OF COMMAND LINE ARGUMENTS

See Table 8.

APPENDIX C
THE MAPPERF.CONF FILE USED IN THE SELF-TEST

```
# Mapperf.conf configuration file (for self-test)

  # Basic parameters
  Tester-L-IPv6 2001:db8:6::2
  Tester-R-IPv4 203.0.113.56
  Tester-R-IPv6 2001:db8:7::2 # for background frames
  Tester-L-MAC ec:f4:bb:dd:07:28 # ST eno1
  Tester-R-MAC ec:f4:bb:dd:07:2a # ST eno2
  DUT-L-MAC ec:f4:bb:dd:07:2a # ST eno2
  DUT-R-MAC ec:f4:bb:dd:07:28 # ST eno1

  # Port selection parameters
  # Some port range boundary values
  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
  bg-dport-min 1 # as RFC4814 recommends
  bg-dport-max 49151 # as RFC4814 recommends
  bg-sport-min 1024 # as RFC4814 recommends
  bg-sport-max 65535 # as RFC4814 recommends

  # How port numbers vary? 1:inc., 2:dec., 3:random
  FW-var-sport 3

  FW-var-dport 3
  RV-var-sport 3
  RV-var-dport 3

  # MAP rules parameters
  NUM-OF-CEs 1000 # Number of simulated CEs
  BMR-IPv6-Prefix 2001:db8:ce::
  BMR-IPv6-Prefix-Length 51
  BMR-IPv4-Prefix 192.0.2.0
  BMR-IPv4-prefix-Length 24
  BMR-EA-Len 13
  DMR-IPv6-Prefix 64:ff9b::
  DMR-IPv6-prefix-Length 64

  # Device hardware parameters
  CPU-FW-Send 2 # Forward Sender runs on this core
  CPU-FW-Receive 4 # Forward Receiver runs on this core
  CPU-RV-Send 6 # Reverse Sender runs on this core
  CPU-RV-Receive 8 # Reverse Receiver runs on this core
  Mem-Channels 2

  # Network traffic parameters
  FW 1 # Forward direction (0:inactive; 1:active)
  RV 1 # Reverse direction (0:inactive; 1:active)
  Promisc 0 # Promiscuous mode (0:inactive; 1:active)
```

APPENDIX D
THE MAPPERF.CONF FILE USED IN THE TESTER-DUT TEST

```
# Mapperf.conf configuration file (for tests with DUT)

  # Basic parameters
  Tester-L-IPv6 2001:db8:6::2
  Tester-R-IPv4 203.0.113.56
  Tester-R-IPv6 2001:db8:7::2 # for background traffic
  DUT-L-IPv6 2001:db8:6::1 # corresponding BR for
  MAP-E
  Tester-L-MAC ec:f4:bb:ef:98:a0 # Tester eno1
  Tester-R-MAC ec:f4:bb:ef:98:a2 # Tester eno2
  DUT-L-MAC ec:f4:bb:dc:a6:b8 # DUT eno1
  DUT-R-MAC ec:f4:bb:dc:a6:ba # DUT eno2

  # Port selection parameters
  # Some port range boundary values
  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
  bg-dport-min 1 # as RFC4814 recommends
  bg-dport-max 49151 # as RFC4814 recommends
  bg-sport-min 1024 # as RFC4814 recommends
  bg-sport-max 65535 # as RFC4814 recommends
  # How port numbers vary? 1:inc., 2:dec., 3:random

  FW-var-sport 3
  FW-var-dport 3
```

TABLE 7. Description of Mapperf configuration file parameters.

Config. File Parameters	Description
Basic Parameters	
Tester-L-IPv6	The IPv6 address of the left-side interface of the Tester. It should be in the same subnet as that of the DUT-L-IPv6. It is used only for the background traffic. For the MAP-T traffic, it is replaced by the MAP address of the simulated CE.
Tester-R-IPv4	The IPv4 address of the right-side interface of the Tester. It should be in the same subnet as that of the DUT-R-IPv4. This will simulate an IPv4 server.
Tester-R-IPv6	The IPv6 address that will be used by the Tester for forwarding the background (i.e., non-translated) traffic via the right-side interface.
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.
FW-dport-min	The lower limit value of the destination port range that can be used by the forward test packets of the foreground traffic.
FW-dport-max	The upper limit value of the destination port range that can be used by the forward test packets of the foreground traffic.
RV-sport-min	The lower limit value of the source port range that can be used by the reverse test packets of the foreground traffic.
RV-sport-max	The upper limit value of the source port range that can be used by the reverse test packets of the foreground traffic.
bg-sport-min	The lower limit value of the source port range that can be used by the background test packets.
bg-sport-max	The upper limit value of the source port range that can be used by the background test packets.
bg-dport-min	The lower limit value of the destination port range that can be used by the background test packets.
bg-dport-max	The upper limit value of the destination port range that can be used by the background test packets.
FW-var-sport	How the source port numbers vary within their range when sending forward test frames. Possible values are 1 for increase, 2 for decrease, or 3 for pseudo-randomly change.
FW-var-dport	How the destination port numbers vary within their range when sending forward test frames. Possible values are 1 for increase, 2 for decrease, or 3 for pseudo-randomly change.
RV-var-sport	How the source port numbers vary within their range when sending reverse test frames. Possible values are 1 for increase, 2 for decrease, or 3 for pseudo-randomly change.
RV-var-dport	How the destination port numbers vary within their range when sending reverse test frames. Possible values are 1 for increase, 2 for decrease, or 3 for pseudo-randomly change.
MAP Rules Parameters	
NUM-OF-CEs	The number of CEs to be simulated in the test.
BMR-IPv6-Prefix	The rule IPv6 prefix of the BMR that will be shared among all CEs of the same MAP domain.
BMR-IPv6-prefix-length	The number of bits in the MAP address that will be reserved for the BMR-IPv6-Prefix.
BMR-IPv4-Prefix	The public IPv4 prefix of the BMR that will be shared among all CEs of the same MAP domain.
BMR-IPv4-prefix-length	The number of bits of the BMR-IPv4-Prefix.
BMR-EA-length	The number of EA-bits (i.e., IPv4 suffix + PSID) in the MAP address.
DMR-IPv6-Prefix	The IPv6 prefix of the DMR that will be provisioned by the corresponding BR to the simulated CE.
DMR-IPv6-prefix-length	The number of bits in the DMR address (i.e., the IPv4-embedded IPv6 address) that will be reserved for the DMR-IPv6-Prefix.
Device Hardware Parameters	
CPU-FW-Send	The ID of the CPU core to be used by the forward sender thread.
CPU-FW-Receive	The ID of the CPU core to be used by the forward receiver thread.
CPU-RV-Send	The ID of the CPU core to be used by the reverse sender thread.
CPU-RV-Receive	The ID of the CPU core to be used by the reverse receiver thread.
Mem-Channels	The number of memory channels to be used.
Network Traffic Parameters	
FW	A switch flag to enable or disable testing in the forward direction. Possible values are 0 for inactive and 1 for active.
RV	A switch flag to enable or disable testing in the reverse direction. Possible values are 0 for inactive and 1 for active.
Promisc	A switch flag to enable or disable testing in the promiscuous mode. Possible values are 0 for inactive and 1 for active. It is only set when needed for the sake of fixing some testing problems (e.g., using incorrect MAC addresses or generating bad checksums).

TABLE 8. Description and usage of command line arguments.

Command Line Argument (unit)	Description	Usage
IPv6 frame size (bytes).	The size of the IPv6 test frames. It should be set according to section 5.1.1 of RFC 8219 [9]. When MAP-T BR is tested, the size of the IPv6 test frames will be set to its value, whereas when MAP-E BR is tested, their size will be set to 20 bytes longer. The size of the IPv4 frames will always be 20 bytes shorter, and the background test frames will always be of this size.	Mapperf-tp, Mapperf-lat, Mapperf-pdv
Frame rate (frames per second).	The rate at which the test frames will be transmitted.	Mapperf-tp, Mapperf-lat, Mapperf-pdv
Test duration (seconds).	The time duration of a single experiment. Section 24 of RFC 2544 [7] specifies the lower limit value to 60, while Mapperf sets the upper limit value to 3600.	Mapperf-tp, Mapperf-lat, Mapperf-pdv
Stream timeout (milliseconds).	How long should the Tester wait before stopping receiving test frames after sending them completely? This parameter could be compared to the 2000 milliseconds "after sending timeout" recommended by RFC 2544 [7] in its section 23 and complied with RFC 8219 [9] recommendations.	Mapperf-tp, Mapperf-lat, Mapperf-pdv
n & m (n/a)	Two relatively prime numbers specify the proportions of the foreground and background frames. (m out of n is the proportion of foreground traffic, n-m out of n is the proportion of the background traffic. For instance, n=10, m=9, means 90% of the traffic is foreground and 10% is background).	Mapperf-tp, Mapperf-lat, Mapperf-pdv
First tagged delay (seconds)	The delay from the beginning of the test until the sending of the first tagged frame. Section 7.2 of RFC 8219 [9] specifies the lower limit value to 60, whereas Mapperf sets the upper limit value to 3600.	Mapperf-lat
Number of tagged frames (n/a)	The number of tagged frames (i.e., frames with timestamps). Section 7.2 of RFC 8219 [9] requires at least 500, whereas Mapperf sets the upper limit value to 50,000.	Mapperf-lat
Frame timeout (milliseconds)	The frame will be considered "lost" if its delay is greater than the value of this parameter. The value of 0 means that no per-frame timeout is used.	Mapperf-pdv
Operation Mode (n/a)	The mode that the Tester will run. It could be 'e' or 'E' for encapsulation, or 't' or T' for translation.	Mapperf-tp, Mapperf-lat, Mapperf-pdv

```
RV-var-sport 3
RV-var-dport 3

# MAP rules parameters
NUM-OF-CEs 1000 # Number of simulated CEs in the test
BMR-IPv6-Prefix 2001:db8:ce::
BMR-IPv6-prefix-length 51
BMR-IPv4-Prefix 192.0.2.0
BMR-IPv4-prefix-length 24
BMR-EA-length 13
DMR-IPv6-Prefix 64:ff9b::
DMR-IPv6-prefix-length 64

# Device hardware parameters
CPU-FW-Send 2 # Forward Sender runs on this core
CPU-FW-Receive 4 # Forward Receiver runs on this core
CPU-RV-Send 6 # Reverse Sender runs on this core
CPU-RV-Receive 8 # Reverse Receiver runs on this core

Mem-Channels 2
# Network traffic parameters
FW 1 # Forward direction (0:inactive; 1:active)
RV 1 # Reverse direction (0:inactive; 1:active)
Promisc 0 # Promiscuous mode (0:inactive; 1:active)
```

APPENDIX E
THE PSEUDOCODE OF THE THROUGHPUT SHELL SCRIPT

set the configuration parameters (*test direction, maximum frame rate, frame size, test duration, timeout, percentage of foreground traffic, sleep duration, measurement error (e), and the number of test runs*) to their desired values

```
  set test_run to 0
  WHILE test_run < number_of_test_runs
    set lower bound rate L to 0
    set upper bound rate H to the maximum frame rate
  REPEAT
    set frame_rate to (H+L)/2
    call Mapperf-tp with proper config. Params.
    wait for test_duration+timeout
    IF (test_direction = FORWARD)
    THEN  IF (fw_received = fw_sent)
            THEN the test is "successful"
            OTHERWISE, the test is "failed"
    ELSEIF (test direction = REVERSE)
    THEN IF (rv_received = rv_sent)
            THEN the test is "successful"
            OTHERWISE, the test is "failed"
```

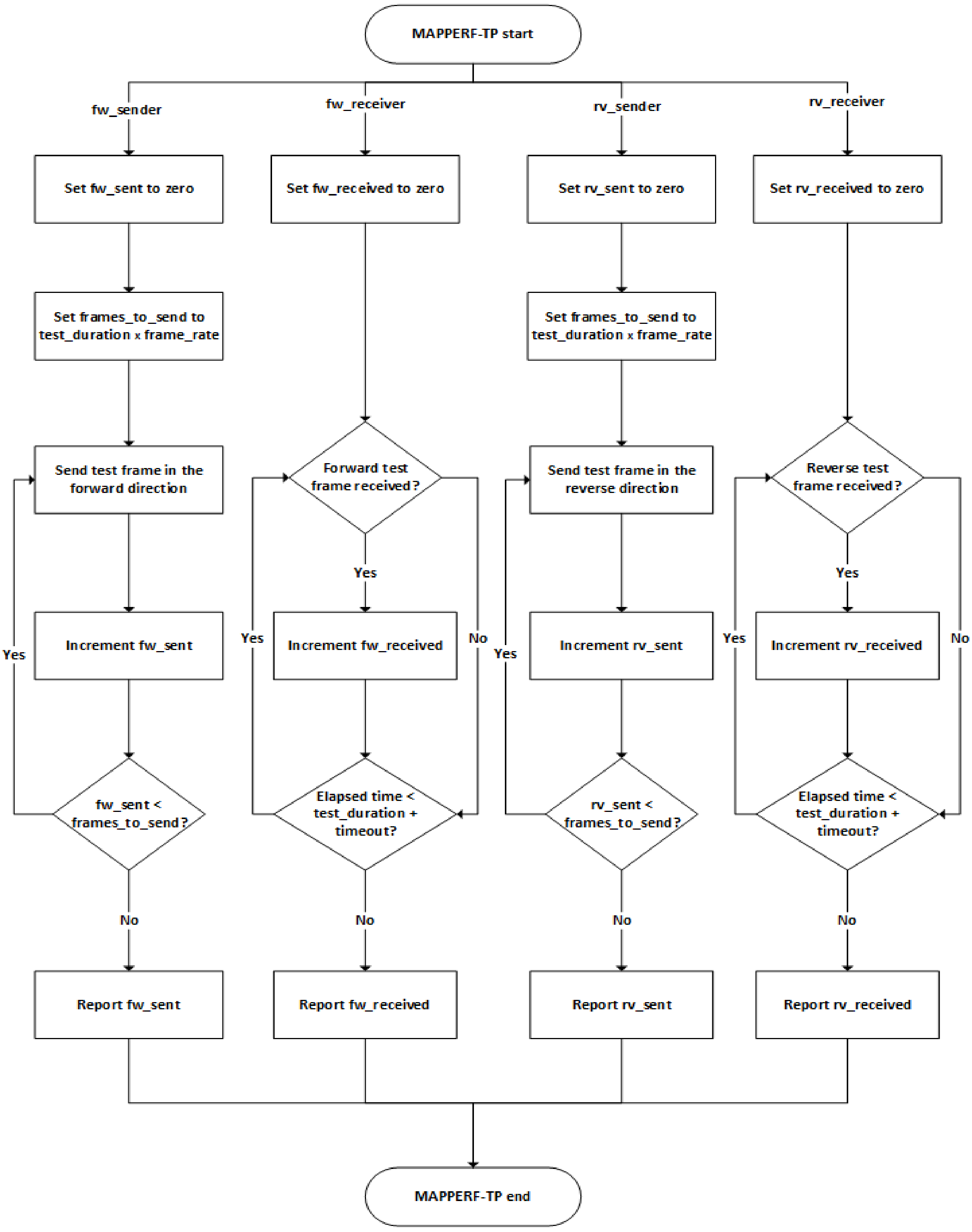

MAPPERF-TP start
fw_sender
fw_receiver
rv_sender
rv_receiver
Set fw_sent to zero
Set fw_received to zero
Set rv_sent to zero
Set rv_received to zero
Set frames_to_send to test_duration x frame_rate
Set frames_to_send to test_duration x frame_rate
Send test frame in the forward direction
Forward test frame received?
Send test frame in the reverse direction
Reverse test frame received?
Yes
Yes
No
Yes
Yes
No
Yes
Yes
Increment fw_sent
Increment fw_received
Increment rv_sent
Increment rv_received
fw_sent < frames_to_send?
Elapsed time < test_duration + timeout?
rv_sent < frames_to_send?
Elapsed time < test_duration + timeout?
No
No
No
No
Report fw_sent
Report fw_received
Report rv_sent
Report rv_received
MAPPERF-TP end

FIGURE 16. The Basic Flowchart of Mapperf-tp.

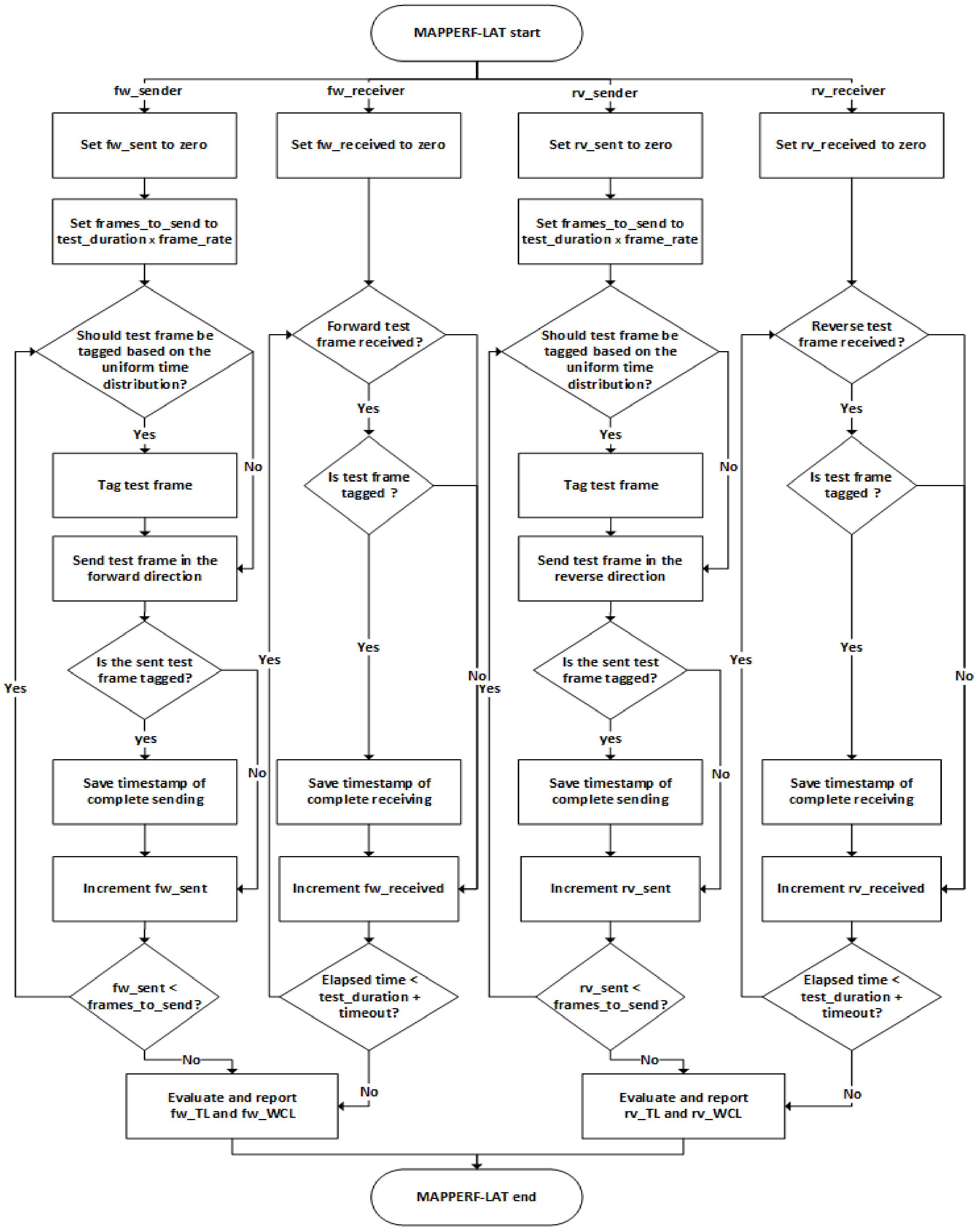

MAPPERF-LAT start
fw_sender
fw_receiver
rv_sender
rv_receiver
Set fw_sent to zero
Set fw_received to zero
Set rv_sent to zero
Set rv_received to zero
Set frames_to_send to test_duration x frame_rate
Set frames_to_send to test_duration x frame_rate
Should test frame be tagged based on the uniform time distribution?
Forward test frame received?
Should test frame be tagged based on the uniform time distribution?
Reverse test frame received?
Yes
No
Tag test frame
Is test frame tagged ?
Tag test frame
Is test frame tagged ?
Send test frame in the forward direction
Send test frame in the reverse direction
Is the sent test frame tagged?
Is the sent test frame tagged?
yes
Save timestamp of complete sending
Save timestamp of complete receiving
Save timestamp of complete sending
Save timestamp of complete receiving
Increment fw_sent
Increment fw_received
Increment rv_sent
Increment rv_received
fw_sent < frames_to_send?
Elapsed time < test_duration + timeout?
rv_sent < frames_to_send?
Elapsed time < test_duration + timeout?
Evaluate and report fw_TL and fw_WCL
Evaluate and report rv_TL and rv_WCL
MAPPERF-LAT end

FIGURE 17. The Basic Flowchart of Mapperf-lat.

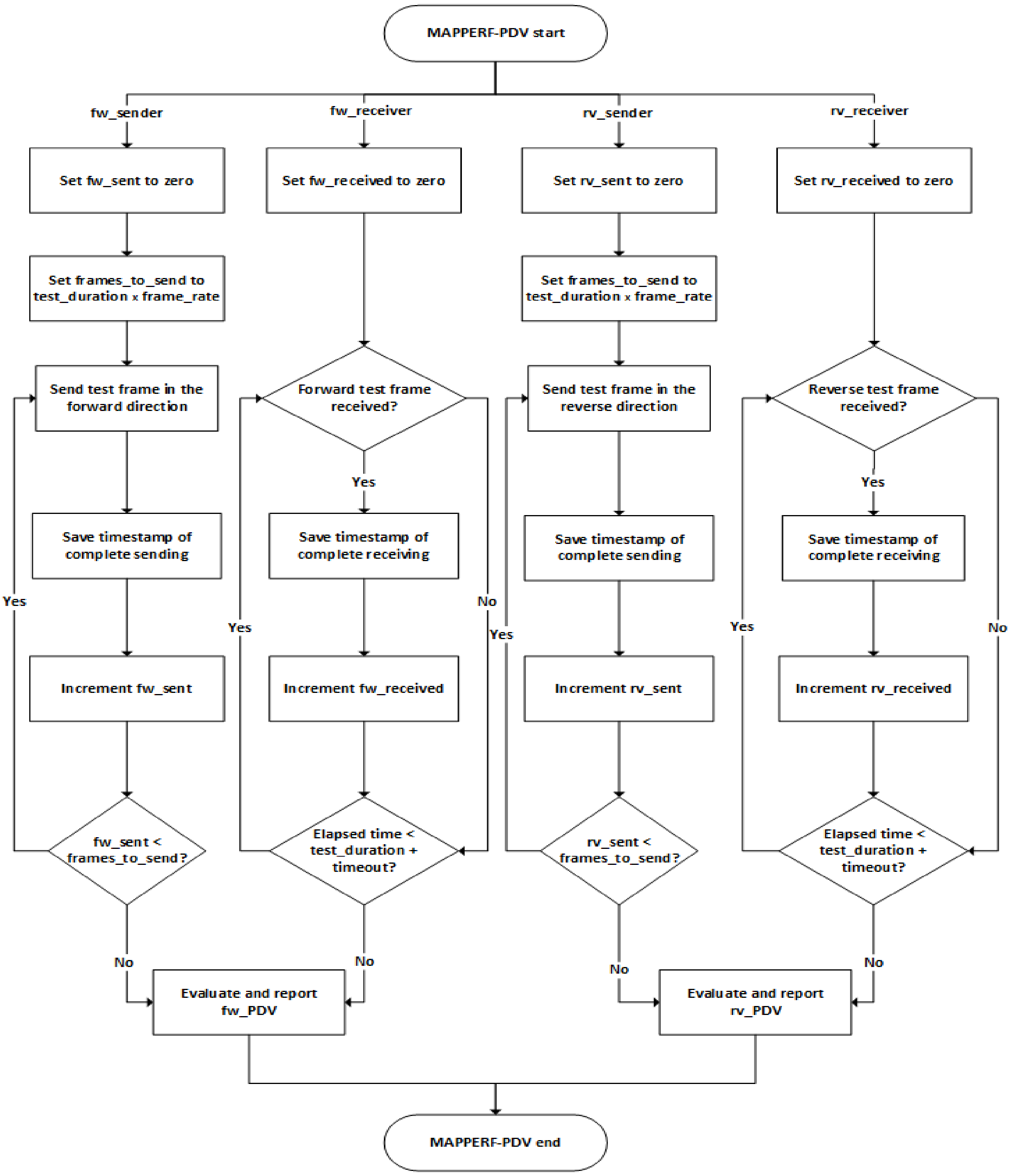

MAPPERF-PDV start
fw_sender
fw_receiver
rv_sender
rv_receiver
Set fw_sent to zero
Set fw_received to zero
Set rv_sent to zero
Set rv_received to zero
Set frames_to_send to test_duration x frame_rate
Set frames_to_send to test_duration x frame_rate
Send test frame in the forward direction
Forward test frame received?
Send test frame in the reverse direction
Reverse test frame received?
Yes
Yes
Save timestamp of complete sending
Save timestamp of complete receiving
Save timestamp of complete sending
Save timestamp of complete receiving
Yes
Yes
No
Yes
Yes
No
Increment fw_sent
Increment fw_received
Increment rv_sent
Increment rv_received
fw_sent < frames_to_send?
Elapsed time < test_duration + timeout?
rv_sent < frames_to_send?
Elapsed time < test_duration + timeout?
No
No
No
No
Evaluate and report fw_PDV
Evaluate and report rv_PDV
MAPPERF-PDV end

FIGURE 18. The Basic Flowchart of Mapperf-pdv.

```
ELSEIF (test direction = BIDIRECTIONAL)
THEN IF (fw_received = fw_sent AND
          rv_received = rv_sent)
          THEN the test is "successful"
          OTHERWISE, the test is "failed"
ENDIF
IF the test is successful
THEN set L to frame_rate
OTHERWISE set H to frame_rate
sleep for a sleep duration
```

```
    UNTIL H-L <=e
    record frame_rate into the results file
    increment test_run
ENDWHILE
```

APPENDIX F
THE PSEUDOCODE OF THE FLR SHELL SCRIPT

set the configuration parameters (*test direction, start_rate, end_rate, rate_step, frame size, test duration, timeout, percentage of foreground traffic, sleep time, and the number of test runs*) to their desired values

```
set test_run to 0
WHILE test_run < number_of_test_runs
set frame_rate to start_rate
REPEAT
    call Mapperf-tp with proper config. params.
    wait for test_duration+timeout
    IF (test_direction = FORWARD)
    THEN set sent to fw_sent
        set received to fw_received
    ELSEIF (test direction = REVERSE)
    THEN set sent to rv_sent
        set received to rv_received
    ELSEIF (test direction = BIDIRECTIONAL)
    THEN set sent to fw_sent+rv_sent
    set received to fw_received+rv_received
    ENDIF
    set FLR to (sent-received)/sent∗100
        record FLR into the results file
        increment frame_rate by rate_step
        sleep for a sleep duration
    UNTIL frame_rate > end_rate
    increment test_run
ENDWHILE
```

APPENDIX G
THE PSEUDOCODE OF THE LATENCY SHELL SCRIPT

set the configuration parameters (*test direction, Throughput rate, frame size, test duration, timeout, percentage of foreground traffic, preamble delay, number of tagged frames, sleep time, and the number of test runs*) to their desired values

```
set test_run to 0
WHILE test_run < number_of_test_runs
    call Mapperf-lat with proper config. params.
    wait for test_duration+timeout
    IF (test_direction = FORWARD)
    THEN record fw_TL and fw_WCL into the results
        file
    ELSEIF (test direction = REVERSE)
    THEN record rv_TL and rv_WCL into the results file
    ELSEIF (test direction = BIDIRECTIONAL)
    THEN record fw_TL, fw_WCL, rv_TL, and rv_WCL
      into the results file
    ENDIF
    sleep for a sleep duration
    increment test_run
ENDWHILE
```

APPENDIX H
THE PSEUDOCODE OF THE PDV SHELL SCRIPT

set the configuration parameters (*test direction, Throughput rate, frame size, test duration, timeout, percentage of foreground traffic, sleep time, and the number of test runs*) to their desired values

```
set test_run to 0
WHILE test_run < number_of_test_runs
    call Mapperf-pdv with proper config. params.
    wait for test_duration+timeout
    IF (test_direction = FORWARD)
    THEN record fw_PDV into the results file
    ELSEIF (test direction = REVERSE)
    THEN record rv_PDV into the results file
    ELSEIF (test direction = BIDIRECTIONAL)
    THEN record fw_PDV and rv_PDV into the results
        file
    ENDIF
    sleep for a sleep duration
    increment test_run
ENDWHILE
```

APPENDIX I
THE BASIC FLOWCHART OF MAPPERF-TP

See Fig. 16.

APPENDIX J
THE BASIC FLOWCHART OF MAPPERF-LAT

See Fig. 17.

APPENDIX K
THE BASIC FLOWCHART OF MAPPERF-PDV

See Fig. 18.

ACKNOWLEDGMENT

The resources of NICT StarBED, Nomi, Ishikawa, Japan, were used remotely for the development of Mapperf and for carrying out benchmarking measurements.

The authors would like to thank Shuuhei Takimoto for the possibility of using StarBED. They also thank Bertalan Kovács for reading and commenting on the manuscript.

REFERENCES

[1] G. Lencse, J. P. Martinez, L. Howard, R. Patterson, and I. Farrer, *Pros and Cons of IPv6 Transition Technologies for IPv4-as-a-Service (IPv4aaS)*, document IETF RFC 9313, Oct. 2022, doi: 10.17487/RFC9313.

[2] M. Mawatari, M. Kawashima, and C. Byrne, *464XLAT: Combination of Stateful and Stateless Translation*, document IETF RFC 6877, Apr. 2013, doi: 10.17487/RFC6877.

[3] A. Durand, R. Droms, J. Woodyatt, and Y. Lee, *Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion*, document IETF RFC 6333, 2011, doi: 10.17487/RFC6333.

[4] Y. Cui, Q. Sun, M. Boucadair, T. Tsou, Y. Lee, and I. Farrer, *Lightweight 4Over6: An Extension to the Dual-Stack Lite Architecture*, document IETF RFC 7596, 2015, doi: 10.17487/RFC7596.

[5] E. O. Troan, W. Dec, X. Li, C. Bao, S. Matsushima, T. Murakami, and E. T. Taylor, "Mapping of address and port with encapsulation (MAP-E)," document IETF RFC 7597, 2015, doi: 10.17487/RFC7597.
[6] X. Li, C. Bao, E. W. Dec, O. Troan, S. Matsushima, and T. Murakami, *Mapping of Address and Port Using Translation (MAP-T)*, document IETF RFC 7599, Jul. 2015, doi: 10.17487/RFC7599.
[7] S. Bradner and J. McQuaid, *Benchmarking Methodology for Network Interconnect Devices*, document IETF RFC 2544, Mar. 1999, doi: 10.17487/RFC2544.
[8] C. Popoviciu, A. Hamza, G. Van de Velde, and D. Dugatkin, *IPv6 Benchmarking Methodology for Network Interconnect Devices*, document IETF RFC 5180, May 2008, doi: 10.17487/RFC5180.
[9] M. Georgescu, L. Pislaru, and G. Lencse, *Benchmarking Methodology for IPv6 Transition Technologies*, document IETF RFC 8219, Aug. 2017, doi: 10.17487/RFC8219.
[10] G. Lencse and Á. Bazsó, "Benchmarking methodology for IPv4aaS technologies: Comparison of the scalability of the jool implementation of 464XLAT and MAP-T," *Comput. Commun.*, vol. 219, pp. 243–258, Apr. 2024, doi: 10.1016/j.comcom.2024.03.007.
[11] FD.io. *VPP*. Accessed: May 22, 2025. [Online]. Available: https://wiki.fd.io/view/VPP
[12] Jool. *MAP-T Summary*. Accessed: May 22, 2025. [Online]. Available: https://www.jool.mx/en/map-t.html
[13] A. Al-hamadani and G. Lencse, "Maptperf: An RFC 8219 compliant tester for benchmarking MAP-T border relay routers," *Comput. Netw.*, vol. 257, Feb. 2025, Art. no. 111012, doi: 10.1016/j.comnet.2024.111012.
[14] G. Lencse and N. Nagy, "Towards the scalability comparison of the jool implementation of the 464XLAT and of the MAP-T IPv4aaS technologies," *Int. J. Commun. Syst.*, vol. 35, no. 18, Dec. 2022, Art. no. e5354, doi: 10.1002/dac.5354.
[15] P. Srisuresh and M. Holdrege, *IP Network Address Translator (NAT) Terminology and Considerations*, document IETF RFC 2663, 1999, doi: 10.17487/RFC2663.
[16] M. Bagnulo, P. Matthews, and I. V. Beijnum, *Stateful NAT64: Network Address and Protocol Translation From IPv6 Clients To IPv4 Servers*, document IETF RFC 6146, Apr. 2011, doi: 10.17487/RFC6146.
[17] X. Li, C. Bao, M. Chen, H. Zhang, and J. Wu, *The China Education and Research Network (CERNET) IVI Translation Design and Deployment for the IPv4/IPv6 Coexistence and Transition*, document IETF RFC 6219, May 2011, doi: 10.17487/RFC6219.
[18] M. Georgescu, H. Hazeyama, Y. Kadobayashi, and S. Yamaguchi, "Empirical analysis of IPv6 transition technologies using the IPv6 network evaluation testbed," *EAI Endorsed Trans. Ind. Netw. Intell. Syst.*, vol. 2, no. 2, p. e1, Feb. 2015, doi: 10.4108/inis.2.2.e1.
[19] E. J. Bound, *IPv6 Enterprise Network Scenarios*, document IETF RFC 4057, 2005, doi: 10.17487/RFC4057.
[20] M. Asama. *Map Supported Vyatta*. Accessed: May 22, 2025. [Online]. Available: https://www.ginzado.ne.jp/~m-asama/vyatta/map
[21] A. Botta, A. Dainotti, and A. Pescapé, "A tool for the generation of realistic network workload for emerging networking scenarios," *Comput. Netw.*, vol. 56, no. 15, pp. 3531–3547, Oct. 2012, doi: 10.1016/j.comnet.2012.02.019.
[22] M. Georgescu, H. Hazeyama, T. Okuda, Y. Kadobayashi, and S. Yamaguchi, "Benchmarking the load scalability of IPv6 transition technologies: A black-box analysis," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2015, pp. 329–334, doi: 10.1109/ISCC.2015.7405536.
[23] S. Avallone, S. Guadagno, D. Emma, A. Pescape, and G. Ventre, "D-ITG distributed Internet traffic generator," in *Proc. 1st Int. Conf. Quant. Eval. Syst.*, 2004, pp. 316–317, doi: 10.1109/qest.2004.1348045.
[24] Jool. *464XLAT Summary*. Accessed: May 22, 2025. [Online]. Available: https://www.jool.mx/en/464xlat.html
[25] G. Lencse, "Design and implementation of a software tester for benchmarking stateless NAT64 gateways," *IEICE Trans. Commun.*, vol. 104, no. 2, pp. 128–140, Feb. 2021, doi: 10.1587/transcom.2019ebn0010.
[26] C. Bao, X. Li, F. Baker, T. Anderson, and F. Gont, *IP/ICMP Translation Algorithm*, document IETF RFC 7915, Jun. 2016, doi: 10.17487/RFC7915.
[27] 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.*, vol. 192, pp. 75–88, Aug. 2022, doi: 10.1016/j.comcom.2022.05.028.
[28] DPDK. *DPDK Documentation*. Accessed: May 22, 2025. [Online]. Available: https://core.dpdk.org/doc
[29] A. Al-Hamadani. *Mapperf: An RFC 8219 Compliant MAP-T/E BR Tester Written in C++ Using DPDK*. Accessed: May 22, 2025. [Online]. Available: https://github.com/alhamadani-ahmed/Mapperf
[30] R. Durstenfeld, "Algorithm 235: Random permutation," *Commun. ACM*, vol. 7, no. 7, p. 420, Jul. 1964, doi: 10.1145/364520.364540.
[31] O. D. Arbeláez. *How Competitive Are C++ Standard Random Number Generators*. Accessed: May 22, 2025. [Online]. Available: https://medium.com/@odarbelaeze/how-competitive-are-c-standard-random-number-generators-f3de98d973f0
[32] FD.io. *Wiki Main Page*. Accessed: May 22, 2025. [Online]. Available: https://wiki.fd.io/view/Main_Page
[33] D. Newman and T. Player, *Hash and Stuffing: Overlooked Factors in Network Device Benchmarking*, document IETF RFC 4814, Mar. 2007, doi: 10.17487/RFC4814.

AHMED AL-HAMADANI received the degree from Florida Institute of Technology (FIT), USA, in 2013, and the M.Sc. degree in computer science. He is currently pursuing the degree with the Department of Networked Systems and Services. Since then, he has been a Lecturer with the Department of Computer Engineering, University of Mosul, Iraq. His field of research is benchmarking and performance analysis of IPv6 Transition Technologies. He has been awarded the Stipendium Hungaricum Scholarship to pursue his Ph.D. degree in informatics from Budapest University of Technology and Economics (BME), Hungary.

GÁBOR LENCSE received the M.Sc. and Ph.D. degrees in computer science from Budapest University of Technology and Economics, Budapest, Hungary, in 1994 and 2001, respectively.

He has been with the Department of Telecommunications, Széchenyi István University, Győr, Hungary, since 1997. Currently, he is a Professor. He has also been a part-time Senior Research Fellow with the Department of Networked Systems and Services, Budapest University of Technology and Economics, since 2005. He is the co-author of RFC 8219 and RFC 9313. His research interests include the performance and security analysis of IPv6 transition technologies.