GapReplay: A High-Accuracy Packet Replayer

Shuanghong Yu*, Fenghua Li*†, Han Zhang*†, Kunling He*, and Xingkun Yao‡
*Institute for Network Sciences and Cyberspace, BNRist, Tsinghua University, Beijing, China
†Zhongguancun Laboratory, Beijing, China
‡Information Technology Center, Tsinghua University, Beijing, China
{ysh20, hkl21}@mails.tsinghua.edu.cn, lifh@cernet.edu.cn, {zhhan, yaoxk}@tsinghua.edu.cn

ICC 2023 - IEEE International Conference on Communications | 978-1-5386-7462-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICC45041.2023.10279258

***Abstract*—Network traffic has become increasingly complicated and diverse with the continuing development of the Internet. It is challenging for the traffic generators to construct a large variety of traffic while maintaining the similarity to the real traffic. Given that real traffic from the production network is handy to retrieve, replaying traces from the production network can maximize the authenticity of traffic, contributing to experiments to evaluate the effectiveness of algorithms. Recently, many pieces of research in the networking field, especially algorithms based on machine learning (ML), are extremely sensitive to statistical characteristics of the traffic, such as ML-based traffic classification, traffic detection, traffic prediction, etc. However, there are only a few available statistics for the growing encrypted traffic, such as packet size and inter-arrival time (IAT). To replay traffic accurately, in this paper, we propose the GapReplay, a packet replayer that can remain identical with the original nanosecond-precision pcap trace in packet contents and achieve high accuracy in timestamps. GapReplay reaches line rate while transmitting packets by appending and extending packets, which will be dropped and truncated by a programmable device, to guarantee satisfying accuracy. We implement our mechanism on top of DPDK and P4. The evaluation results demonstrate that GapReplay can achieve a nanosecond-level accuracy, much better than the state-of-the-art such as MoonGen and tcpreplay, where the best of them is at least 1000 times less accurate than our mechanism.**

I. Introduction

With the continuous evolution of the Internet, there are more and more applications running on it, giving birth to a variety of transmission protocols and network devices. These protocols and devices are required to be tested and verified completely before deployment. Thus, many testbeds have emerged all over the world in recent decades, providing environment and resources needed by network experiments for researchers, such as PlanetLab [1], GENI [2], OneLab [3], CloudLab [4] and Fabric [5], etc. In the experiment, besides generating traffic by the researchers themselves, real traffic from the production network can be replayed reproducibly, validating the effect in the production environment and compare with other alternatives. In this paper, we propose and develop a high-accuracy traffic replay tool to provide an as real environment for experiments as possible and make these experiments reproducible.

Among these networking experiments, some are greatly affected by the accuracy of the replayed traffic. For example, the effect of the deep packet inspection (DPI) mechanism is diminishing due to the growing share of encrypted traffic (up to 80% by 2019 [6]). Instead, statistics such as inter-arrival times and packet sizes are useful when adopting ML-based methods, especially for the encrypted traffic. Various traffic classification algorithms [7]–[9], traffic detection mechanisms [10]–[12], traffic prediction methods [9], [13], intrusion detection systems (IDSes) [14]–[16] and intrusion prevention systems (IPSes) [17] heavily rely on these statistical characteristics.

These experiments are very sensitive to the timestamps, more specifically, inter-arrival times of replayed traffic. However, the accuracy of the existing replay tools is not so satisfying. For example, tcpreplay [18] is a widely used tool to replay pre-captured pcap files, which utilizes the kernel network stack. Due to the expensive overhead of switching from user space to kernel space, the inter-arrival times of replayed traffic sent by tcpreplay are far away from the original packets'. MoonGen [19] is a scriptable traffic generator built on top of DPDK [20], which contains example scripts for pcap replaying. However, only pcap files with microsecond precision can be parsed by MoonGen, making it inaccurate especially for high-speed scenarios, because it is very likely that there are two packets with the same microsecond timestamp in a high-speed pcap file.

To achieve high accuracy while replaying a pcap file, we design and implement a traffic replay tool based on DPDK and programmable hardware. From our point of view, the gap between two adjacent packets is the root cause of the error because it is basically impossible to control the gap accurately while replaying. However, controlling gaps between two adjacent packets is mainly adopted by the existing traffic replay tools. Therefore, we proposed eliminating all the gaps by making all the packets we send back-to-back. To enhance the accuracy of the input, pcap files featuring nanosecond resolution are mandatory to describe the inter-arrival times of packets more accurately. Additionally, to eliminate all gaps and ensure that all packets are back-to-back, large gaps are filled by a number of jumbo frames with an invalid ethertype. Besides the large gaps, there are small gaps that cannot hold any packets, even the smallest 60-byte frame. For those, the gap is covered by extending the former packet. Finally, the line-rate traffic will be processed by a programmable device where the appended and extended packets will be dropped and trimmed, respectively. We chose the programmable hardware to operate these packets because we need to guarantee that the latencies for all packets must be identical to avoid errors. In order to develop the prototype, a Tofino-based P4 switch is chosen as the programmable hardware. The evaluation results show that our traffic replay tool can limit the error between

Authorized licensed use limited to: Tsinghua University. Downloaded on October 15,2026 at 14:32:40 UTC from IEEE Xplore. Restrictions apply.

replayed traffic and original traffic within nanoseconds, while the error caused by other state-of-the-art replay tools is at least hundreds of microseconds.

The remainder of the paper is structured as follows. Section II surveys the related works, including tools for both traffic replaying and traffic generating. Section III presents the overall framework, which is composed of the gap replayer and the gap handler. In Section IV, some implementation details are stated. In Section V, the accuracy of our proposed mechanism is evaluated and compared with some other available traffic replay tools. Finally, we conclude the paper in In Section VI.

II. Related Works

A. Traffic Replayer

TCPivo [21] aims to replay traffic with high accuracy. To achieve that, several methods are adopted in TCPivo. First, it prefetches the pcap trace from the file system to reduce the IO latency. Second, while parsing packets from the pcap trace, TCPivo supports replacing the payload with self-generated, null-padded data to accelerate. Last, some kernel patches are provided by TCPivo to address the scheduling issue of the kernel, improving the accuracy for replaying.

Tcpreplay [18] is a collection of free open-source utilities for editing and replaying pre-captured network traces, including tcpreplay, tcprewrite, tcpreplay-edit, tcpprep, tcpbridge and tcpcapinfo. It supports replaying packets at arbitrary speeds. Packets preload is supported by tcpreplay, too.

Besides the above two, there are stateful replay tools that mimic the behavior of particular protocols. TCPOpera [22] emulates a TCP protocol stack and operates packets at flow level to eliminate all the ghost packets. DETER [23] is a deterministic replay tool for TCP protocol, which can be used to diagnose various TCP performance problems by recording and replaying individual TCP connections. DETER also adopts a kernel patch to reduce the delay ACK timeout.

B. Traffic Generator

D-ITG (Distributed Internet Traffic Generator) [24] is a platform to generate highly customized traffic with multiple patterns and protocols. It can also be used to perform measurements such as one-way-delay (OWD), round-trip-time (RTT), jitter and throughput, etc.

Swing [25] is a closed-loop, network-responsive traffic generator which learns the interactions of applications, users and network behaviors. Corresponding traffic can be generated using a simple structural model, and it's evaluated that the aggregate characteristics can match the original traffic.

MoonGen [19] is a scriptable traffic generator based on DPDK and LuaJIT. Besides generating traffic with complex patterns, it can be used to measure latencies with sub-microsecond accuracy by using the hardware timestamping capabilities of modern commodity NICs. Additionally, due to its scriptable capability, it can also be used to replay traffic by executing a pcap-replaying script.

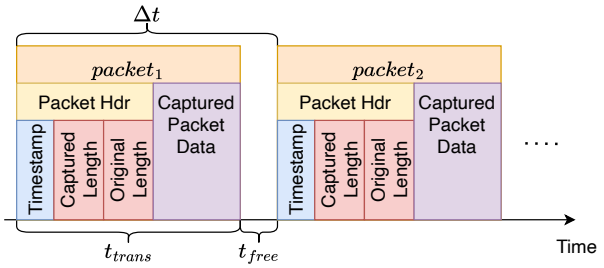


Fig. 1. Packets in a pcap file

III. Design of GapReplay

It is challenging for the existing traffic replay tools to control the IATs accurately by waiting for a specific duration with system calls like `sleep()`. Although MoonGen supports filling gaps between packets with packets whose CRC checksums are invalid, it is designed for traffic generation, cannot parse pcap files with nanosecond precision, and is unable to handle gaps shorter than 96 bytes, which is called the small gap in our framework. Therefore, in this paper, to accurately replay all the packets in a pcap file on wire according to the original sequence and inter-arrival times, we design and implement a high-accuracy packet replayer, GapReplay.

A. Factors Affecting Replay

The packets in the pcap file can be formulated in Fig. 1. Each packet has its own length, and there are inter-arrival times between every two adjacent packets. So these two are factors we should take into account to replay traffic accurately.

1) Packet Sizes: By simply copying packets from the pcap file, packet sizes are easy to be consistent with the original lengths.

2) Inter-arrival Times: Besides the packet sizes, there are differences Δt between the timestamps of two adjacent packets. Part of the difference t_{trans} is used to transmit the former packet, and the free time of the link t_{free} occupies the left. To replay packets accurately, we have to guarantee that all the IATs Δt of replayed traffic should be as close to the original ones in the pcap file as possible.

However, the packet is not immediately on wire after we send it by executing the system call. Instead, the overhead introduced by context switches between user/kernel space and calling the driver to put packets in the NIC's transmission queues can result in huge errors.

Moreover, even if the above errors can be avoided with user-space network frameworks such as DPDK, PF_RING ZC [26] and netmap [27], other errors are still likely to emerge. For instance, it is challenging to wait for exactly Δt if we use functions like `sleep()` or even the high-precision timer RDTSC because overheads are still there for interrupts and function calls.

Therefore, we need to devise another method to reduce these errors, which will be presented below.

Authorized licensed use limited to: Tsinghua University. Downloaded on October 15,2026 at 14:32:40 UTC from IEEE Xplore. Restrictions apply.

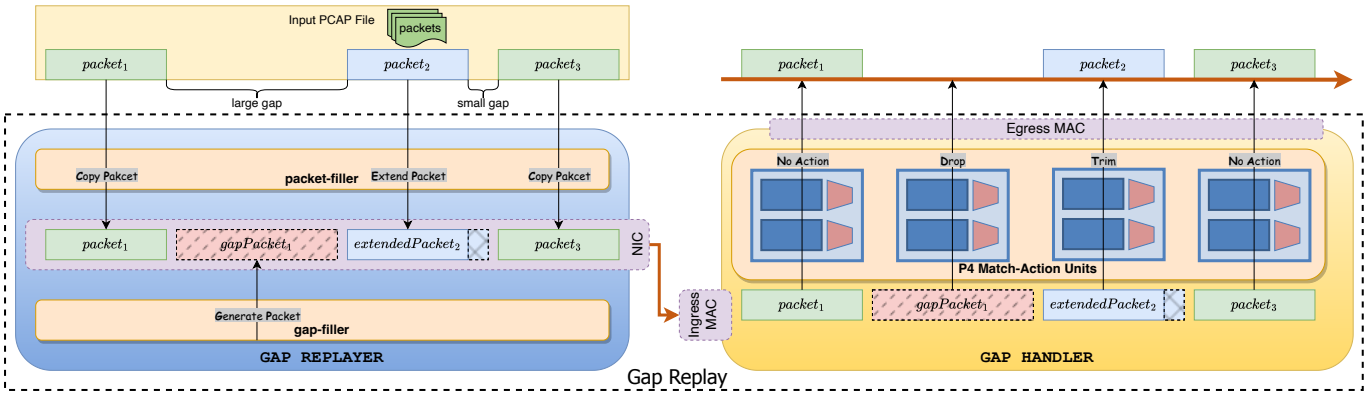


Fig. 2. Overview of the proposed framework

B. Overview

The overview of our mechanism can be demonstrated in Fig. 2. It has two components: a gap replayer and a gap handler.

The gap replayer consists of two fillers, i.e., a packet-filler and a gap-filler. The packet-filler parses packets from a pcap file and sends them on wire. Most of these packets remain unchanged like $packet_1$ and $packet_3$, which are called the normal packets, while some of them are slightly extended several bytes (up to 95 bytes), such as $packet_2$, and the extended part is marked with cross hatches after $extendedPacket_2$, these packets are defined as the extended packets. The gap-filler generates and transmits gap packets, which are jumbo frames with an invalid ethernet type 0x0801, illustrated as shadowed $gapPacket_1$.

After the gap replayer, the line-rate packets will be redirected to our programmable hardware, a P4 switch named gap handler in the framework. All the packets are divided into three scenarios, and the corresponding action is applied for each scenario as follows:

- **No action:** Normal packets are the same as the ones in the original pcap file, so there is no need to process them in the gap handler. Simply send them to the egress and forward them.
- **Drop:** Gap packets are generated and appended to wire by the gap filler and do not exist in the original pcap file. These packets should not be forwarded so the gap handler will discard them.
- **Trim:** Extended packets are similar to normal packets, but they are slightly extended. These packets should be restored to what they are in the pcap file, so they will be truncated a couple of bytes according to a specific field in itself, which will be explained in detail later.

Note that actions applied to packets are processed by hardware, so the overheads for all the packets are identical. That's why we chose programmable hardware to handle the gap packets and extended packets.

With the help of packet-filler and gap-filler, all the gaps in the original pcap file are filled, and the NIC of the gap replayer achieves line rate. Errors caused by inter-arrival times will never appear again because there are no gaps between packets anymore. The gap filling is the key to achieving line rate and avoiding errors, so the filling mechanism should be carefully designed. Gaps are divided into two categories, corresponding to the packet-filler and the gap-filler, respectively, which will be described in detail later.

C. Packet Gaps

The gaps between adjacent packets are classified into two categories according to the inter-arrival times, the link speed of the NIC and the sizes of the former packets. To simplify, we regard the available link capacity on wire that is not occupied by the packets as gap, which can be expressed as an equation

$$G = I \times N - L \ ,$$

where G is the length of the gap, I denotes the inter-arrival times, N denotes the link speed of the NIC and L denotes the length of the former packet.

Consider that the minimum frame size for Ethernet is 64 bytes [28], besides a seven-octet preamble, a one-octet start frame delimiter (SFD) and two interpacket gaps (IPGs) whose length is at least 12 bytes on each side of the frame, for any gap that is shorter than 96 bytes, it's impossible to insert any frame even if it is the smallest. Therefore, gaps shorter than 96 bytes are called small gaps, and the rest are divided into large gaps.

1) Large Gaps: The large gaps should be filled by a certain number of jumbo frames. The reason we use jumbo frames to fill gaps is to minimize the amount of them for simplicity. Note that the preambles, SFDs and IPGs should be considered while calculating the sizes and number of frames. The typical size of jumbo frames in our implementation is 8880 bytes, including the FCS. If the size of the last frame is calculated as shorter than 64 bytes, the last two frames will be averaged to avoid that the last one cannot be transmitted.

2) Small Gaps: As described above, no frame can be inserted into a small gap, and methods like `sleep()` cannot be applied either. The reason why the insertion operation cannot be performed is that the length of the wanted frame is invalid. However, the length of the former one is undoubtedly qualified, so the former packet can be extended to fill this small gap without any insertion.

Authorized licensed use limited to: Tsinghua University. Downloaded on October 15,2026 at 14:32:40 UTC from IEEE Xplore. Restrictions apply.

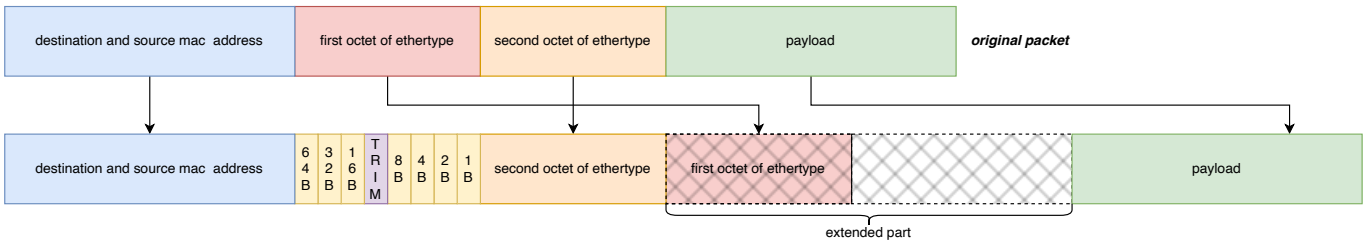


Fig. 3. Packet extension mechanism

TABLE I
NICs Available in Our Experiment

NIC	Bandwidth	Driver	Port	HWTSTAMP_FILTER_ALL
Intel I350	1Gbps	igb	TP	✓
Intel X550 T2	10Gbps	ixgbe	TP	✓
Intel 82599ES	10Gbps	ixgbe	FIBER	✗
Intel X722	10Gbps	i40e	FIBER	✗
Mellanox ConnectX-4 Lx (MCX4121-ACAT)	25Gbps	mlx5_core	FIBER	✓

IV. Implementation Details of GapReplay

In the 10Gbps scenario, there are no obstacles to achieving line rate on top of DPDK, and it is pretty easy for a P4 switch to discard all the gap packets with an invalid ethertype 0x0801. Therefore, in this section, we mainly describe the implementation of small gaps in detail.

In order to keep the length of the replayed packet consistent with the original packet's length, the extended packets must be truncated and restored. However, to the best of our knowledge, there are no commodity NICs or switches that have the capability to trim a dynamic length of a packet according to a specific field of itself. Programmable devices such as Field Programmable Gate Arrays (FPGA) and Programming Protocol-independent Packet Processors (P4) are thus adopted to accomplish the above operation.

Information should be encoded into packets to distinguish packets that need to be truncated and indicate the truncating length while processing traffic in the P4 switch. Because the minimum unit for transmission is a byte, i.e., 8bits, as long as a packet is going to be extended, it will be 8 bits longer than before, at least.

As described in Section III, the maximum length of small gaps is 95 bytes. Considering the IPG with the latter packet, which takes up at least 12 bytes, the maximum length for extending is 83 bytes. Except for the cost of storing the extended length, which takes 7 bits for numbers up to 83, one bit is left to indicate whether a packet is to be truncated.

As shown in Fig. 3, we chose the first octet of the ethertype field to place the information mentioned above. The fourth bit indicates whether the packet should be trimmed because ethertypes with the fourth bit set to 1 are the rarest and barely used according to the IEEE 802 Numbers from IANA [29]. The other 7 bits are gathered to represent the length to be truncated, that is, the 1st, 2nd, 3rd, 5th, 6th, 7th and 8th bits set to 1 will indicate the P4 switch to trim the packet by 64 bytes, 32 bytes, 16 bytes, 8 bytes, 4 bytes, 2 bytes and 1 byte respectively. The length to be trimmed ranges from 0 to 127 bytes, enough for the maximum value of 83 bytes. The extended part of the packet is marked with cross hatches, where the original first octet of ethertype is backed up at the beginning of it.

While processing the extended packets in P4, the original first octet of ethertype in the extended part is stored in metadata first, after which the function `pkt.advance()` [30] is called to ignore several bytes according to the 7 bits mentioned above. Once the trimming operation is done, the first octet of ethertype in metadata will be restored to its original position.

V. Evaluation

A. Experiment Setup

1) Hardware Timestamping: To compare the difference between the original traffic and the replayed traffic, the timestamps of replayed traffic we capture should be as accurate as possible. However, the accuracy of software timestamps provided by the Linux kernel cannot meet our requirements. In order to address that issue, hardware timestamps are embedded into pcap files while capturing replayed traffic, which needs additional capabilities for devices.

Hardware timestamp by NICs: NICs usually are capable of hardware timestamping, but most of them can only timestamp some specific packets, e.g., the PTP packets due to performance limitations. NICs with `HWTSTAMP_FILTER_ALL` rx_filter in driver are abled to timestamp all incoming packets [31], which can be used in our evaluation.

Hardware timestamp by P4 ASIC: Hardware timestamps are also available in P4 ASIC. Tofino provides the per-packet timestamp information. All packets entering the P4 pipeline have intrinsic metadata containing the hardware timestamp from the chip.

Authorized licensed use limited to: Tsinghua University. Downloaded on October 15,2026 at 14:32:40 UTC from IEEE Xplore. Restrictions apply.

Available NICs in our experiment are listed in Table I. All ports on our P4 switch are SFP+ or QSFP28, so the Intel I350 and X550-T2 cannot be used in our experiment. Besides, we found that the hardware timestamp of Mellanox ConnectX-4 Lx is not accurate all the time, which can influence our experiments greatly. Therefore, in order to timestamp packets using hardware ability while trimming and dropping packets with the P4 switch in the meantime, hardware timestamp from P4 ASIC is the only option in our experiment.

2) Data Source: The source pcap file is captured from the campus backbone of Tsinghua University in nanosecond resolution with more than 800000 packets. As for MoonGen, the source pcap file is in microsecond precision, derived from the nanosecond resolution source, because pcap files with the nanosecond magic number are unacceptable for MoonGen, as described in Section I.

It is important to examine if all the timestamps in the pcap file are valid, that is, all the IATs between two packets must be sufficient to transmit the former. Otherwise, there will be unavoidable errors among those packets while replaying.

3) Topology: In our implementation, the generated line-rate traffic is transmitted by an Intel 82599ES port to an SFP+ port of our P4 switch, where the packets are forwarded, discarded or truncated. Once the hardware processing is done, the P4 switch will redirect all the packets to one lane of a QSFP28 port, which is broken into four SFP28 ports. The port corresponding to the above lane is connected to our Mellanox CX4 NIC running at 25Gbps, ensuring sufficient bandwidth and capturing the replayed traffic.

4) Metrics: To evaluate the accuracy of our proposed traffic replay tool, two metrics are used in our validation experiments, as described below.

Cumulative Latency: Denote that the timestamp of the k-th packet in the source pcap file is t_k, and the timestamp of the k-th packet in the captured pcap file of replayed traffic is t'_k. Then the cumulative latency at the k-th packet l_k can be expressed as an equation

$$l_k = (t'_k - t'_0) - (t_k - t_0).$$

With the cumulative latency, the trend of delay can be observed obviously, indicating the accuracy of replayed packets.

Distribution of IATs' Deviation: The deviation of IATs at the k-th packet d_k can be formulated as

$$d_k = (t'_k - t'_{k-1}) - (t_k - t_{k-1}).$$

To evaluate the precision of replayed packets, we use this metric to furnish a clear view of the distribution, providing a better understanding of the deviations of all corresponding IATs.

B. Results

We compare our solution with another two replay tools, i.e., tcpreplay and MoonGen. Note that MoonGen only provides an example script for replaying based on their software rate control, so we implement another script based on their hardware rate control and use both of them in our evaluation.

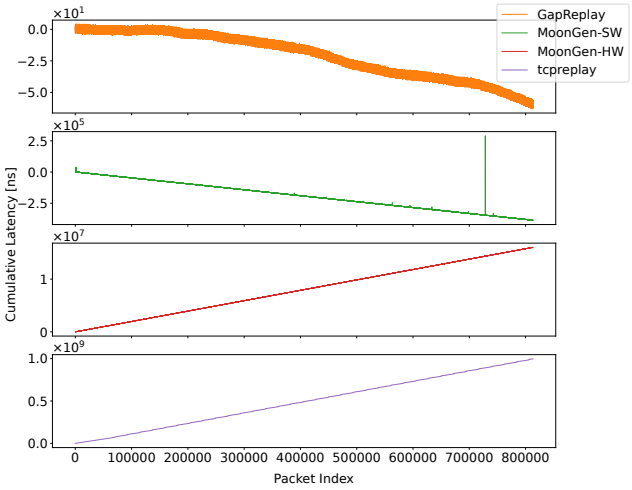
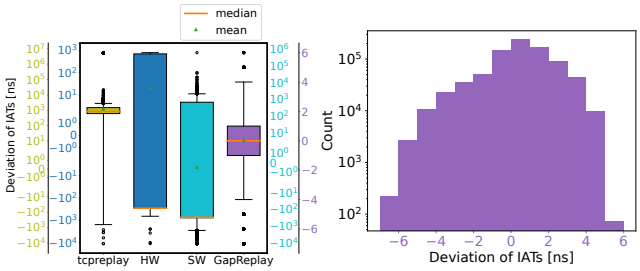


Fig. 4. Cumulative latency



(a) Comparison (b) GapReplay (detailed)

Fig. 5. Distributions of IATs' deviation

Fig. 4 illustrates the cumulative latency and the distributions of IATs' deviation are shown in Fig. 5, from where conclusions can be drawn as follows:

1) GapReplay Achieves the Best Accuracy and Precision: In Fig. 4, we can see that GapReplay achieves the best accuracy with the lowest cumulative latency, i.e., around 60ns. In contrast, the cumulative latency of MoonGen's software-based method, MoonGen's hardware-based method and tcpreplay is about 400µs, 16ms and 1s, respectively, much worse than the GapReplay.

As for the precision, the GapReplay's deviation of IATs ranges from -7ns to 6ns, which is also the best, and it is illustrated more specifically in Fig. 5(b). On the contrary, the scope is much broader and there are plenty of outliers for all the other three methods, as is illustrated in Fig. 5(a).

2) Software-Based Method Outperforms the Hardware-Based One In MoonGen: It is interesting to find that the experiment result of the software-based method is much better than the hardware-based method for MoonGen. After carefully reviewing the hardware-rate-control-based script we wrote, we guess this abnormal result is likely related to the poor resolution of the microsecond-level pcap file parsed by MoonGen.

3) Outliers in MoonGen and tcpreplay: From Fig. 5(a), we can notice that many outliers appear in the evaluation

Authorized licensed use limited to: Tsinghua University. Downloaded on October 15,2026 at 14:32:40 UTC from IEEE Xplore. Restrictions apply.

experiments except our proposed GapReplay, contributing inaccuracy to MoonGen and tcpreplay. Among them, the software-based MoonGen has the most concentrated outliers, which can be viewed as some bursts obviously in Fig. 4, and the tcpreplay owns the most significant single outlier, which is approximately 5.4ms.

VI. Conclusion

In this paper, we propose the GapReplay, a packet replayer with high accuracy and the ability to replay pcap traces reproducibly. The evaluation results prove that the packets replayed by GapReplay are much closer to the original ones when compared with other tools. In the future, we plan to extend our mechanism to more platforms such as NetFPGA [32] and other programmable hardware to improve the accuracy of the replayed traffic further, and to scale GapReplay up to 100Gbps scenarios to enhance the throughput.

Acknowledgement

This work was supported in part by the National Key R&D Program of China under Grant No.2020YFE0200500 and in part by the National Natural Science Foundation of China under Grants No.62002009. This work is part of Future Internet Technology Infrastructure (FITI).

References

[1] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, Jul. 2003.

[2] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "GENI: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, Mar. 2014.

[3] S. Fdida, T. Friedman, and T. Parmentelat, "OneLab: An open federated facility for experimentally driven future internet research," in *New Network Architectures*. Springer, 2010, pp. 141–152.

[4] D. Duplyakin, R. Ricci, A. Maricq, G. Wong, J. Duerig, E. Eide, L. Stoller, M. Hibler, D. Johnson, and K. Webb, "The Design and Operation of CloudLab," in *2019 USENIX annual technical conference (USENIX ATC 19)*, 2019, pp. 1–14.

[5] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K.-C. Wang, T. Lehman, and P. Ruth, "FABRIC: A National-Scale Programmable Experimental Network Infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47, 2019, conference Name: IEEE Internet Computing.

[6] M. Geller and P. Nair, "5G security innovation with Cisco," *Whitepaper Cisco Public*, pp. 1–29, 2018.

[7] N. Bayat, W. Jackson, and D. Liu, "Deep Learning for Network Traffic Classification," arXiv, Tech. Rep. arXiv:2106.12693, Jun. 2021, arXiv:2106.12693 [cs] type: article. [Online]. Available: http://arxiv.org/abs/2106.12693

[8] S. Roy, T. Shapira, and Y. Shavitt, "Fast and lean encrypted Internet traffic classification," *Computer Communications*, vol. 186, pp. 166–173, Mar. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422000408

[9] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "Mutated traffic detection and recovery: an adversarial generative deep learning approach," *Annals of Telecommunications*, Apr. 2022. [Online]. Available: https://doi.org/10.1007/s12243-022-00909-8

[10] O. Bader, A. Lichy, C. Hajaj, R. Dubin, and A. Dvir, "MalDIST: From Encrypted Traffic Classification to Malware Traffic Detection and Classification," in *2022 IEEE 19th Annual Consumer Communications Networking Conference (CCNC)*, 2022, pp. 527–533, iSSN: 2331-9860.

[11] O. Osanaiye, K.-K. R. Choo, and M. Dlodlo, "Change-point cloud DDoS detection using packet inter-arrival time," in *2016 8th Computer Science and Electronic Engineering (CEEC)*, Sep. 2016, pp. 204–209.

[12] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli, "Tunnel Hunter: Detecting application-layer tunnels with statistical fingerprinting," *Computer Networks*, vol. 53, no. 1, pp. 81–97, Jan. 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128608003071

[13] A. Pimpinella, A. E. C. Redondi, F. Loh, and M. Seufert, "Machine-Learning Based Prediction of Next HTTP Request Arrival Time in Adaptive Video Streaming," in *2021 17th International Conference on Network and Service Management (CNSM)*, 2021, pp. 558–564, iSSN: 2165-963X.

[14] M. Zolotukhin, P. Kotilainen, and T. Hämäläinen, "Intelligent IDS Chaining for Network Attack Mitigation in SDN," in *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*, 2021, pp. 786–791.

[15] Y. Ponomarchuk and D.-W. Seo, "Intrusion detection based on traffic analysis in wireless sensor networks," in *The 19th Annual Wireless and Optical Communications Conference (WOCC 2010)*, 2010, pp. 1–7, iSSN: 2379-1276.

[16] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, "Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset)," in *Selected Papers from the 12th International Networking Conference*, ser. Lecture Notes in Networks and Systems, B. Ghita and S. Shiaeles, Eds. Cham: Springer International Publishing, 2021, pp. 73–84.

[17] M.-J. Chen, C.-C. Wen, H.-C. Lin, and Y.-S. Chu, "ASIC design and implementation for VoIP intrusion prevention system," in *2016 International Conference on Applied System Innovation (ICASI)*, 2016, pp. 1–4.

[18] F. Klassen and AppNeta, "Tcpreplay: Pcap editing and replaying utilities," *tcpreplay. appneta. com*, 2013.

[19] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "MoonGen: A Scriptable High-Speed Packet Generator," in *Proceedings of the 2015 Internet Measurement Conference*. Tokyo Japan: ACM, Oct. 2015, pp. 275–287. [Online]. Available: https://dl.acm.org/doi/10.1145/2815675.2815692

[20] "Data Plane Development Kit." [Online]. Available: https://www.dpdk.org/

[21] W.-c. Feng, A. Goel, A. Bezzaz, W.-c. Feng, and J. Walpole, "TCPivo: a high-performance packet replay engine," in *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, ser. MoMeTools '03. New York, NY, USA: Association for Computing Machinery, Aug. 2003, pp. 57–64. [Online]. Available: https://doi.org/10.1145/944773.944783

[22] S.-S. Hong and S. F. Wu, "On interactive internet traffic replay," in *International Workshop on Recent Advances in Intrusion Detection*, 2005, pp. 247–264.

[23] Y. Li, R. Miao, M. Alizadeh, and M. Yu, "{DETER}: Deterministic {TCP} replay for performance diagnosis," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, 2019, pp. 437–452.

[24] S. Avallone, S. Guadagno, D. Emma, A. Pescape, and G. Ventre, "D-ITG distributed Internet traffic generator," in *First International Conference on the Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings.*, Sep. 2004, pp. 316–317.

[25] K. V. Vishwanath and A. Vahdat, "Swing: Realistic and Responsive Network Traffic Generation," *IEEE/ACM Transactions on Networking*, vol. 17, no. 3, pp. 712–725, Jun. 2009, conference Name: IEEE/ACM Transactions on Networking.

[26] "PF_ring ZC (Zero Copy)," Apr. 2014. [Online]. Available: https://www.ntop.org/products/packet-capture/pf_ring/pf_ring-zc-zero-copy/

[27] L. Rizzo, "netmap: a novel framework for fast packet I/O," in *21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 101–112.

[28] S. Bradner and J. McQuaid, "Benchmarking methodology for network interconnect devices," Internet Engineering Task Force, Tech. Rep. RFC 2544, 1999.

[29] "IEEE 802 Numbers." [Online]. Available: https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml

[30] "P4 Specifications." [Online]. Available: https://p4.org/specs/

[31] A. Grigorjew, P. Diederich, T. Hoßfeld, and W. Kellerer, "Affordable Measurement Setups for Networking Device Latency with Sub-Microsecond Accuracy," *Würzburg Workshop on Next-Generation Communication Networks (WueWoWas'22)*, p. 5, 2022.

[32] "NetFPGA." [Online]. Available: https://netfpga.org/

Authorized licensed use limited to: Tsinghua University. Downloaded on October 15,2026 at 14:32:40 UTC from IEEE Xplore. Restrictions apply.