A First Look at Accurate Network Traffic Generation in Virtual Environments

Giuseppe Aceto[1], Ciro Guida[1,2], Antonio Montieri[1], Valerio Persico[1], Antonio Pescapè[1]
(1) University of Napoli Federico II (Italy), name.surname@unina.it
(2) University of Bergamo (Italy)

***Abstract*—The generation of synthetic network traffic is necessary to several fundamental networking activities, ranging from device testing to path monitoring, with implications on security and management. While literature focused on high-rate traffic generation, for many use cases *accurate* traffic generation is of importance instead. These scenarios have expanded with Network Function Virtualization, Software Defined Networking, and Cloud applications, which introduce further causes for alterations of generated traffic. Such causes are described and experimentally evaluated in this work, where the generation accuracy of D-ITG, an open-source software generator, is investigated in a virtualized environment. A definition of accuracy in terms of Mean Absolute Percentage Error of the sequences of Payload Lengths (PLs) and Inter-Departure Times (IDTs) is exploited to this end. The tool is found accurate for all PLs and for IDTs greater than one millisecond, and after the correction of a systematic error, also from $100\,\mu$s.**

***Index Terms*—network traffic generation; network management; virtual environments; Software Defined Networking; Network Function Virtualization.**

I. Introduction

The complex and ever-evolving nature of traffic traversing the Internet, and the parallel rapid evolution of networking infrastructure and services, both prompted for the need of advanced and automated tools for evaluating network paths, devices, and applications. A fundamental activity related to such evaluations is the capability of generating network traffic with desired characteristics: Traffic Generation (TG). Such activity is necessary e.g. to test security appliances such as firewalls, intrusion and anomaly detection systems, but also for testing the new network automation management infrastructure offered by programmable network paradigms such as Software Defined Networking (SDN) and Network Function Virtualization (NFV). While much research has been performed in the past regarding *high performance traffic generation*, focused on generating high rates of traffic, little attention has been devoted to *accurate traffic generation*, whose purpose is to reproduce the timing and dimensions of packets with maximum control on the error. Such accuracy generally benefits the repeatability of the experiments, but has gained more importance due to the adoption of data-hungry advanced deep learning methods showing great success in identifying, classifying, and predicting (encrypted) network traffic. Indeed, such methods need sizeable datasets to learn a model of traffic, and *dataset augmentation* approaches are being researched to satisfy these needs [1]. In these scenarios, by generating traffic affected by unchecked alterations may either fail to trigger the identification and classification capabilities of a trained model (if used for testing), or could constitute an ineffective source for augmenting the dataset (if used for training). While *hardware-based network traffic generators* can in principle be configured to provide the highest accuracy, they are severely restricted in usage scenarios due to their limited deployment possibilities and high cost; on the contrary, *software-based* generators not only can be easily deployed in high number of instances, but also can be easily "moved" to every end-system and middlebox on the network. Moreover, software-based network traffic generators can be deployed in virtualized environment, better emulating the actual modern application deployment scenario, including cloud services and NFV. On the other hand, the execution in virtualized environments and on general-purpose operating systems can impact in many ways the accuracy of both timing and dimension (aggregation) of packets. Potential causes of discrepancies include: processing overhead in the networking stack; timing "noise" due to other processes; optimization features such as interrupt coalescence and offloading to the network interface card of segmentation and reassembly. Therefore an assessment of actual generation capabilities for a virtualized network traffic generator (and possible remedies) is in order.

Given the lack of scientific literature regarding the accurate software-based network traffic generation [2], in this work we propose an assessment of TG accuracy, taking into account all above considerations regarding TG via (virtualized) software solutions. To validate the proposed methodology, we present an assessment of the accuracy in terms of *Payload Length* and *Inter-Departure Time* generated by an open-source traffic generator tool called `Distributed Internet Traffic Generator (D-ITG)` [3]. This software has been chosen for its great flexibility, allowing to explore *accurate traffic generation* (and in future works also mixes of different kinds of traffic generation).

To summarize we present the following key contributions:

- we define generation accuracy (in terms of inter-departure time and transport-layer payload length sequences), and propose a methodology to assess, regardless of the generation tool, how accurate the traffic generation is;
- we provide an experimental evaluation of `D-ITG` accuracy in generating network traffic according to user-given requirements in a virtualized environment;
- we apply and evaluate the correction of systematic error

that adversely affect the required inter-departure time sequences;
- we perform *all the analyses* with reference to a dataset of human-generated real mobile app traffic, to quantify the impact of error in a real-world scenario.

The rest of this paper is organized as follows. Section II describes background and related work in TG. Section III details the methodology proposed for validating TG accuracy and the setup used for the experimentation. Section IV presents the evaluation results and provides guidance for the generation of real traffic traces. Section V concludes the paper.

II. Background and Related Work

While synthetic TG is an activity widely performed for testing and active network measurements, different aspects are of interest according to the use case. For testing purposes, the main desired characteristic is the capacity of generating high throughput, able to saturate the path under test [2]. For other purposes (e.g., active measurements, training of machine learning algorithms), the ability to accurately emulate or even faithfully replicate the smallest-detail traffic properties (in terms of payload length and inter-departure time) is much more significant, but is investigated only in ad-hoc cases, not under the general umbrella of synthetic TG. High-performance traffic generators can be implemented via dedicated hardware, but besides being relatively very expensive [4], they lack the deployment flexibility allowed by software, that is the more needed the more SDN and NFV become widely adopted. Besides deployment flexibility, also the ease of configuration and update play in favor of software-based traffic generators [5], that therefore provide attractive solutions in nearly all application scenarios. Software-based traffic generators can be categorized in three classes described hereafter.

- *Replay engines*—work based on pre-recorded packet traces and are able to replicate (a portion of) such traffic. Examples of such generators are [3], [6], [7], [8], [9].
- *Model-based*—allow to generate network traffic based on theoretical statistical distribution or distributions learned from data using machine learning and deep learning techniques (e.g., [7], [10], [11], [12]).
- *Application-based* generators are related to specific type of network traffic and network conditions at different levels of the protocol stack. Examples of this kind of tools are [13], [14], [15], [16].

`D-ITG` is able to act as both *replay engine* and *model-based* generator. More in detail, `D-ITG` supports: *(i)* packets generated in a *stateful* way that allows to better replicate the behavior of real network applications; *(ii)* a *flexible generation* in terms of user-configurable parameters, in addition to a TG by pre-configured well-known statistical distribution; *(iii)* the possibility to replicate a *synchronized bidirectional traffic* from Packet CAPture (PCAP) pre-recorded traces.

This last feature is an essential factor in the evaluation we propose, since it allows to inject traffic into the network with characteristics as close as possible as the required ones.

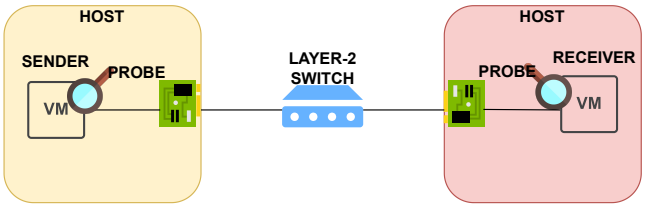


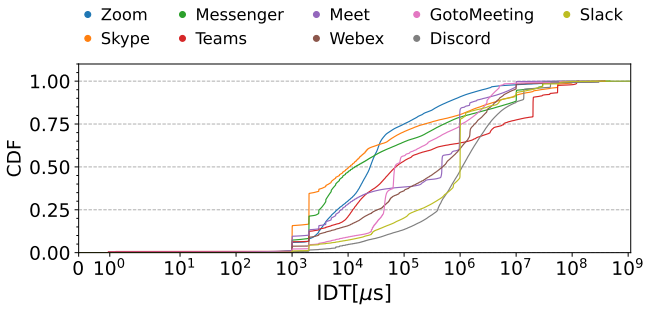
Figure 1: Experimental setup. Sender and receiver software components are executed on different Virtual Machines (VM in the figure), in turn running on distinct physical machines interconnected via a layer-2 switch; `tcpdump` is executed on the guest.

For this analysis, we have considered other publicly available software generation tools (i.e. [6], [7], [9], [10], [16]) but, despite each offering its peculiar features, none of them focused on *accurate traffic generation* or allowed the necessary flexibility to explore it.

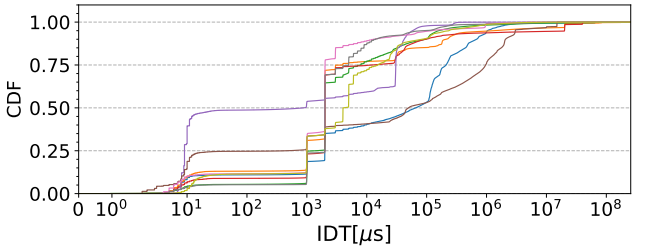
While the deployment in virtualized environment caters to modern ICT infrastructure needs, it also constitutes an additional challenge to the accuracy of software traffic generators. Indeed, the presence of further software layers for sending and receiving packets can affect timing accuracy. Regarding this aspect, Whiteaker et al. [17] quantify the virtualization impact, showing that a solution Xen-based adds about **a hundred microseconds** on round-trip time; similar values ($40 - 80\,\mu s$) have been found investigating a high-performing SDN setup [18]. Wang and Ng [19] instead focused on intra-datacenter delay considering the Amazon EC2 infrastructure-as-a-service, recording delays higher than $10\,ms$ in some cases. Real-world network traffic traces have been collected and exploited in a vast literature: we considered the recent publicly available MIRAGE dataset [20], for extracting the inter-departure times and payload lengths relevant for actual scenarios. All these measurements provide a reference frame for the attainable accuracy and traffic characteristics of interest: in such a frame, in this work we measure the generation accuracy of a software-based traffic generator, `D-ITG` [5] in a virtualized environment, discussing the implied factors and possible solutions.

III. Methodology

In this section we define how we evaluate the generation accuracy of a software-based TG tool in a virtualized environment. First, we describe the experimental setup we refer to, also providing the implementation details for the experimental campaigns we have conducted. Then, we discuss the parameters of interest we focus on to quantify the TG accuracy, as well as the evaluation metrics we consider. Finally, we present how we have designed our experimental campaigns, based on the observation of real traffic.

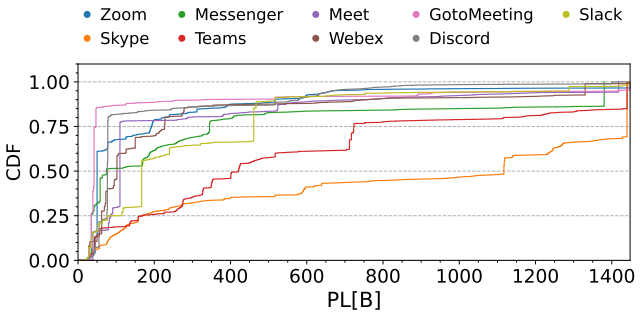


(a) IDT Upstream.

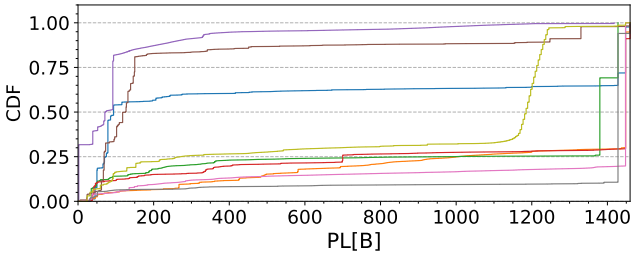


(b) IDT Downstream.

Figure 2: Cumulative distribution function of IDTs for upstream (a) and downstream (b) flows on real traffic.



(a) PL Upstream.



(b) PL Downstream.

Figure 3: Cumulative distribution function of PLs for upstream (a) and downstream (b) flows or real traffic.

A. *Experimental Setup*

The experimental architecture we refer to for our analysis is reported in Fig. 1. All the experiments are conducted leveraging `D-ITG` version 2.9.1. The two parts of the distributed traffic generator (sender and receiver, namely) run on two different virtual machines (guests), which are in execution on two distinct physical machines (hosts). The two physical machines are interconnected via a layer-2 switch. This setup fits with our scenario of interest, where the software components run on dedicated—though virtual—machines. Specifically, the simple network path interconnecting the machines is in line with the goals of our experimentation, which intentionally focuses on the accuracy of the traffic generated by the client. Interference by other processes or devices is meant to be avoided, as its assessment is outside of the scope of our investigation.

To keep the measurements as close as possible to the generating process, we put our probes in the same virtualized environment. Although this could cause interference among the generation, capture, and logging activities, we minimize such an impact by leveraging standard process scheduling utilities (i.e. `nice`/`ionice` and `ramdisk` for logging) to have both a reliable and close-to-generator reading and a general-purpose setup analogous to a real-world virtualized client-server scenario. Network traffic is captured via `tcpdump`.

In the following, we report the implementation details to guarantee reproducibility. The proposed setup is general enough to be applied to evaluate any network-traffic software generation tool. Both virtual machines are provided with 4 GB RAM and 2 vCPU and run Linux operating system. OpenStack (with KVM hypervisor) is used to deploy the virtual machines. The interconnecting link has 1 Gbps capacity.

Hereinafter, we present the results on the traffic as observed on the guest at sender side. We verified that no significant discrepancy is noticeable when comparing the same generation observed at the sender and at the receiver side: analyses and results do not suffer from significant noise added by the virtualization technology, the host operating systems, or the switch, and are thus representative of the investigated phenomenon (generation accuracy).

B. *Evaluating Generation Accuracy*

In this work, we focus on the ability of a virtualized software-based TG tool to accurately reproduce network traffic. This ability can be defined via two distinct but interrelated capabilities: *(i)* generating packets with requested Inter-Departure Times (IDTs); *(ii)* generating packets of requested transport-layer Payload Lengths (PLs). The assumption is that the traffic generator is requested to generate a sequence of packets, with the sequences of IDTs and PLs being given. The accuracy is evaluated comparing the requested sequences with the generated ones.

Generating Requested IDTs. A number of causes for inaccurate IDTs have been identified in the literature which can be divided in internal and external factors [5]: the former are dependent on tasks performed by the traffic generator itself (e.g., for logging information about outgoing packets), while the latter depend upon other tasks (e.g., related to the operating system or other concurrent processes). Internal factors can be tackled with a careful design and implementation of the

software, while external interference can be mitigated via the isolation of the generating process.

To quantitatively assess IDT generation accuracy, we compare the sequences of IDTs requested and generated, and rely on Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE), as the two metrics are complementary in their utility to interpret the practical implications about how accurate we are at generating a specific IDT sequence and how we can make corrections to possible sources of inaccuracy.

Generating Requested PLs. When using the TCP transport-layer protocol, there is no guarantee (by design) that the data committed in a single sending (a *message*) will fit in a single network-layer packet. In fact, the process of TCP segmentation will dynamically adapt to network conditions. As a result, TCP can bundle up many messages in a single segment, or split a message into many segments. In addition, other mechanisms (aimed at improve the network stack efficiency in terms of throughput or latency achievable) may also generate interference. For instance, *TSO/GSO* [21] allows the network stack to submit large packets directly to the hardware which is in charge of splitting them into smaller appropriate-sized packets. Furthermore, *Interrupt Coalescing* [22] allows to hold back send events either until a certain amount of work is pending, or a timeout timer triggers. This technique reduces interrupt overhead, while incurring latency penalties.

When the mentioned approaches are in place, they interfere with the accuracy in the generation of PL sequences, also impacting IDT accurate reproduction. In the following, when inspecting for accurate PL generation, we refer to *coalesced packets* when multiple requested packets are aggregated in a single one, regardless of the cause. For this reason, in our analysis we check whether the generated PL sequence exactly matches the requested one: when this condition does not hold, we exclude the coalesced packets from the evaluation of IDT sequence accuracy. In this way we can assess the two errors (on size and timing) separately and in a reliable way.

C. Design of the Experimentation

Generation accuracy is known to depend upon the characteristics of the requested traffic, with shorter IDTs and smaller PLs being harder to be generated accurately. Therefore, in our experimental campaigns, we instruct `D-ITG` to generate traffic with different IDT and PL values.

In fact, real traffic may have different characteristics in terms of volume, packet rate, and bit rate, depending upon the application that produces it. In this study, we consider as a scenario of specific interest the traffic generated by mobile applications. More specifically, we refer to the MIRAGE project [20] considering a real-world reference dataset—containing the traffic generated by 9 apps (namely, *Zoom, Skype, Messenger, Teams, Meet, Webex, GotoMeeting, Discord*, and *Slack*)—and inspect the distribution of PL and IDT as observed in this traffic. Figures 2 and 3 report the empirical distribution for IDT and PL, considering upstream and downstream direction separately. Despite the distributions remarkably vary according to the specific application (as expected), the analysis highlights how in real traffic the observed IDTs range from few microsecond to tens seconds, whereas PLs span from few bytes to around 1500 B. Notably, significant discrepancies are also evident, depending on traffic direction.

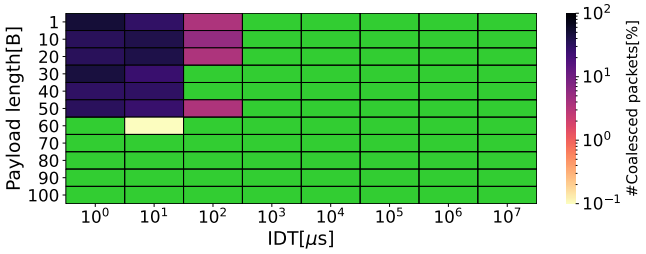


Figure 4: Average ratio of coalesced packets for different IDT and PL combinations. Green cells depicts combinations of IDTs and PLs in which no coalescence occurs.

Accordingly, in our experimentation we have considered PL values in the range 1–1398 B (in accordance with the 1450 B maximum transfer unit in our setup, to avoid undesired fragmentation) and IDT values in the range 1 µs-10 s, in order to evaluate the traffic generation accuracy when aiming at mimicking the traffic generated by mobile apps. For each combination of IDT and PL, we generate 20 flows of 100 packets each.

IV. Experimental Evaluation

Based on the methodology discussed in Sec. III, this section provides the evaluation of `D-ITG` in generating packet sequences with requested IDTs (sec. IV-A) and PLs (Sec. IV-B). Based on the knowledge acquired, we also introduce a calibration mechanism in order to tackle the systematic error assessed (Sec. IV-C).

A. PL Sequences Evaluation

To assess the accuracy of PLs generated, Fig. 4 depicts the percentage of coalesced packets for each combination of IDT and PL requested. Firstly, we underline that the experiments with PL greater than 100 B showed *a total absence of coalescence* and thus Fig. 4 focuses solely on the PL range 1 B-100 B w.r.t. the whole range of requested IDT (spanning from 1 µs to 1 sec). Each cell of heatmap reports the percentage of requested packets that suffered from coalescence, averaged over all the repetitions of the same experiment. Going into details, we can notice that the phenomenon is particularly evident for IDTs < 100 µs and PLs in the range 1–50 B with the highest peak of coalescence ($\approx 90\%$) observed when D-ITG is requested to generate 1B packets. On the other hand, the higher the IDT/PL values, the lower the ratio of coalesced packets: for PLs > 60 B and IDTs > 100 µs the phenomenon disappears (cells colored in green). More specifically, for IDT $= 100$ µs, the coalescence is observed only for PLs ≤ 50 B, but rarely ($\approx 10\%$). Such analysis proves that for IDTs < 100 µs, the sequence of PLs actually injected into the network may not reflect those requested by the user. Since this phenomenon

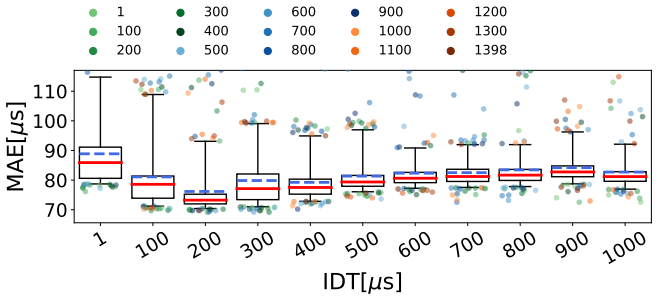
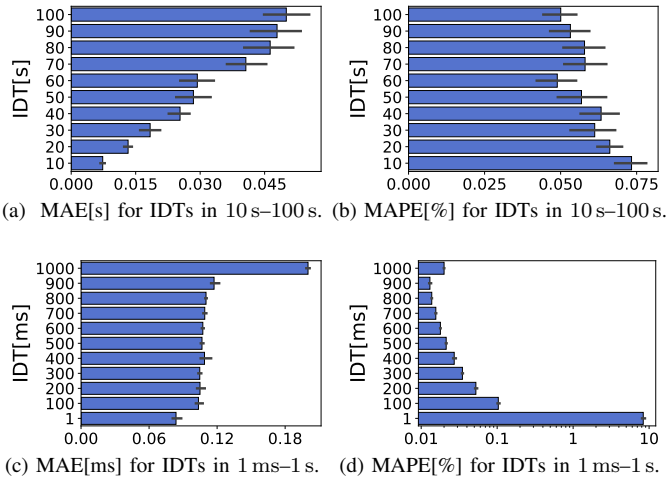


Figure 5: Distribution of PL-independent MAE when considering the 1 µs–200 µs range of IDTs. Dots represent outliers, with colors indicating the related PL values. Whiskers report the 5^{th} and 95^{th} percentiles. Solid red and dashed blue lines depict the median and the mean MAE, respectively.



(a) MAE[s] for IDTs in 10 s–100 s. (b) MAPE[%] for IDTs in 10 s–100 s.

(c) MAE[ms] for IDTs in 1 ms–1 s. (d) MAPE[%] for IDTs in 1 ms–1 s.

Figure 6: MAE and MAPE for different IDTs. PL of 900 B.

affects the evaluation of generated IDTs, in the following analyses, we filter out coalesced packets.

B. IDT Sequences Evaluation

In order to investigate the dependence of IDT generation accuracy upon PL (even in absence of the coalescence phenomenon), we have performed an extensive experimental campaign, covering the range of values of interest for both parameters. Specifically, we have considered PL values from 1 to 1398 B and IDTs from 1 µs to 100 s. The results of the whole campaign are not shown for brevity but summarized hereafter. Figure 5 depicts the total MAE in generating IDTs $\in [1\,\mu s, 1\,ms]$. The results show that the interquartile ranges are distributed around $[70\,\mu s, 90\,ms]$, regardless of IDT. In addition, the outliers do not show any PL dependency, as no clusters are visible.

Accordingly, in the following analyses we adopt a fixed PL of 900 B, corresponding to the median (instead of the mean, for better robustness to outliers) of the PLs observed in the real-world reference dataset (see Fig. 3).

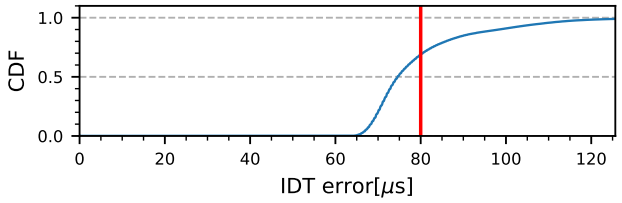


Figure 7: Cumulative distribution of the difference between generated and requested IDTs. Reported values refer to the 1^{st}–99^{th} percentiles range. The vertical red line represents the mean error (approximated to the nearest requested IDT value). PLs are in the range 1 B–1398 B and IDTs are in the range 1 µs–200 µs.

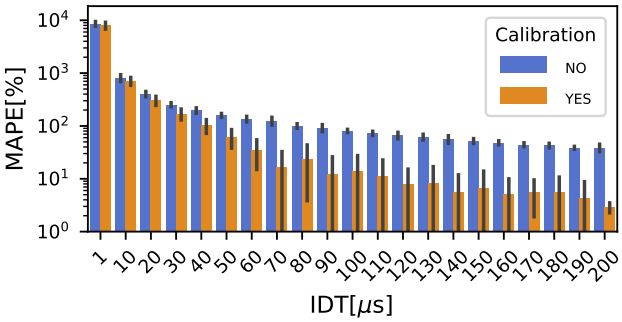


Figure 8: Effects of the calibration mechanism on generation accuracy for IDTs in the range 1 µs–200 µs.

To investigate the accuracy of generation w.r.t. timing, hereinafter we present an analysis at different time scales, in terms of MAE and MAPE between requested and generated IDTs. Figure 6 depicts the performance at the seconds (IDTs $\in [10\,s, 100\,s]$) and milliseconds (IDTs $\in [1\,ms, 1\,s]$) time scales[1], respectively. Also, from the results reported in Fig. 4, it is evident that for IDTs > 1 ms there are no coalescence occurrences; hence, the accuracy in IDT generation is not influenced by the specific PL considered.

Figure 6 shows that `D-ITG` offers a good accuracy in generating the requested IDTs. In Fig. 6a there is an evident trend for MAE—going from ≈ 0.01 s for 10 s IDT to ≈ 0.05 s for 100 s IDT—corresponding to a very small MAPE (Fig. 6b) always lower than 0.075%. A similar situation can be noted at smaller time-scales. Figure 6c reveals that in absolute terms there is a consistent error of hundreds of microseconds for all required IDTs (MAE ≈ 0.1 ms). In relative terms, Fig. 6d shows a decreasing error as the required IDT increases, with a MAPE of $\approx 8\%$ when an IDT of 0.1 ms is requested and less than 1% (and commonly $< 0.1\%$) for higher IDTs.

[1]The $[10\,s, 100\,s]$ range is of special interest since it constitutes at least the 20% of the traffic from the reference real-world dataset (see Fig. 2).

C. IDT Calibration Analysis

Based on these considerations, in the following, we focus on IDTs $\in [1\,\mu s, 200\,\mu s]$ in which a greater variability of errors between requested and generated IDTs can be observed. With the aim of highlighting possible systematic errors, in Fig. 7 we show the cumulative distribution (aggregated by PL) of the timing errors, that are expressed as the difference between generated and requested IDTs. We can notice that 99% of errors are between $65\,\mu s$ and $120\,\mu s$. Also, the evident slope of the distribution suggests that the timing error is due to a systematic source. This error value can be estimated by calculating the average of the errors committed in generating all the required IDTs and approximating this average value to the nearest required IDT. In this way, we obtain a calibration value of $80\,\mu s$. Based on this result, we propose a simple *calibration mechanism* that corrects the systematic error affecting the IDT to be generated before requesting it to D-ITG. In more details, we apply this calibration mechanism to the requested IDTs, so as to mitigate such a consistent error: when the requested IDT $> 80\,\mu s$, the requested generation time is anticipated by $80\,\mu s$; conversely, when the requested IDT $\leq 80\,\mu s$, we instruct `D-ITG` to generate an IDT of $0\,\mu s$ (i.e. back-to-back packets are requested). The effects of this calibration are reported in Fig. 8 in terms of MAPE between requested and generated IDTs by comparing the error with and without the calibration in the range $[1\,\mu s, 200\,\mu s]$. We note that for requested IDTs $< 40\,\mu s$, the effect of calibration is negligible, while for higher IDT values its beneficial effect is increasingly evident, achieving more than an order of magnitude reduction in MAPE when IDTs $> 100\,\mu s$ are requested.

V. Conclusions and Future Perspectives

In this paper we have addressed the issue of *accurate traffic generation* and provided a general methodology for the evaluation of generation accuracy in terms of both IDT and PL, which can be applied to any software generation tool, also in virtualized environments. We provided an assessment of generation accuracy using `D-ITG`, which had never been assessed under the conditions we set, as it was designed to maintain average generation rates. We have verified that `D-ITG` shows great accuracy (MAPE $\leq 0.1\%$) for any PL and for IDTs $>1\,ms$, while a systematic source of error of $\approx 80\,\mu s$ impacts IDTs$\leq 1\,ms$. By applying a simple calibration we reduce the MAPE of one order of magnitude for IDTs down to $100\,\mu s$, effectively reaching the physical limit found in literature for traffic traversal in virtual switches. In a future work, we intend to build on this analysis to propose `D-ITG` as a data augmentation tool for training models on a mix of real and synthetic traffic captures, in order to improve the performance of the same models in the case of class-imbalance problems, and to reduce risks of privacy compromise when real data are used in model training. Other more complex scenarios, involving a mix of traffic flows, can be explored as well.

References

[1] T. Ovasapyan, V. Danilov, and D. Moskvin, "Application of synthetic data generation methods to the detection of network attacks on internet of things devices," *Automatic Control and Computer Sciences*, vol. 55, no. 8, pp. 991–998, 2021.

[2] O. A. Adeleke, N. Bastin, and D. Gurkan, "Network traffic generation: A survey and methodology," *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–23, 2022.

[3] A. Botta, A. Dainotti, and A. Pescapé, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012.

[4] P. Emmerich, S. Gallenmüller, G. Antichi, A. W. Moore, and G. Carle, "Mind the gap-a comparison of software packet generators," in *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. IEEE, 2017, pp. 191–203.

[5] A. Botta, A. Dainotti, and A. Pescapé, "Do you trust your software-based traffic generator?" *IEEE Communications Magazine*, vol. 48, no. 9, pp. 158–165, 2010.

[6] 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, 2009.

[7] D. Turull, P. Sjödin, and R. Olsson, "Pktgen: Measuring performance on high speed networks," *Computer Communications*, vol. 82, pp. 39–48, 2016.

[8] B. R. Patil, M. Moharir, P. K. Mohanty, G. Shobha, and S. Sajeev, "Ostinato-a powerful traffic generator," in *2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*. IEEE, 2017, pp. 1–5.

[9] "GitHub - appneta/tcpreplay: Pcap editing and replay tools for *NIX and Windows," https://github.com/appneta/tcpreplay, accessed: 2021-05-21.

[10] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "MoonGen: A Scriptable High-Speed Packet Generator," *Proceedings of the 2015 ACM Conference on Internet Measurement Conference - IMC '15*, 2015.

[11] M. Tuberquia-David, F. Vela-Vargas, H. López-Chávez, and C. Hernández, "A multifractal wavelet model for the generation of long-range dependency traffic traces with adjustable parameters," *Expert Systems with Applications*, vol. 62, pp. 373–384, 2016.

[12] S. Xu, M. Marwah, M. Arlitt, and N. Ramakrishnan, "Stan: Synthetic network traffic generation with generative neural models," in *International Workshop on Deployable Machine Learning for Security Defense*. Springer, 2021, pp. 3–29.

[13] P. Yan-hui, W. Tao, and Z. Jian-Peng, "A distributed synthetic network traffic generation system," in *2010 First International Conference on Pervasive Computing, Signal Processing and Applications*. IEEE, 2010, pp. 1061–1064.

[14] A. Hafsaoui, N. Nikaein, and L. Wang, "Openairinterface traffic generator (otg): A realistic traffic generation tool for emerging application scenarios," in *2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE, 2012, pp. 492–494.

[15] Y. Zhou, Z. Xi, D. Zhang, Y. Wang, J. Wang, M. Xu, and J. Wu, "Hypertester: high-performance network testing driven by programmable switches," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, 2019, pp. 30–43.

[16] "TRex," https://trex-tgn.cisco.com/, accessed: 2021-11-11.

[17] J. Whiteaker, F. Schneider, and R. Teixeira, "Explaining packet delays under virtualization," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, pp. 38–44, 2011.

[18] J. Hwang, K. K. Ramakrishnan, and T. Wood, "Netvm: High performance and flexible networking using virtualization on commodity platforms," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 34–47, 2015.

[19] G. Wang and T. E. Ng, "The impact of virtualization on network performance of amazon ec2 data center," in *2010 Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–9.

[20] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "Mirage: Mobile-app traffic capture and ground-truth creation," in *2019 4th International Conference on Computing, Communications and Security (ICCCS)*. IEEE, 2019, pp. 1–8.

[21] "Linux networking documentation - segmentation offloads," https://www.kernel.org/doc/html/latest/networking/segmentation-offloads.html, accessed: 2021-10-25.

[22] V. Tran, J. Tourrilhes, K. Ramakrishnan, and P. Sharma, "Accurate available bandwidth measurement with packet batching mitigation for high speed networks," in *2021 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2021, pp. 1–6.