An experimental characterization of the internal generation cycle of an open-source software traffic generator

Leopoldo Angrisani[1], Alessio Botta[1], Gianfranco Miele[2], Michele Vadursi[3]
[1] University of Naples Federico II, via Claudio 21, 80125 Napoli, Italy - Email: a.botta@unina.it
[2] University of Cassino and Southern Lazio, Via G. Di Biasio 43, 03043 Cassino (FR), Italy - Email: g.miele@unicas.it
[3] University of Naples "Parthenope", Centro Direz. Is. C4, 80143 Napoli, Italy - Email: vadursi@uniparthenope.it

Abstract **–The paper is the first step towards the goal of evaluating the measurement uncertainty of the inter-departure times (IDT) provided by software traffic generator. The paper is focused on the experimental characterization of the *internal* generation cycle of a well-known, open source generator, namely D-ITG, for different systems, and under the best possible conditions, i.e. with the minimum system loads. The resulting performance may be seen as the ideal limit the generator can tend to. The extended abstract presents the rationale for the activity, the underlying methodology and some initial tests that highlight the relevance of the clock resolution in the accuracy of IDT.**

Keywords – traffic generator, uncertainty, inter-departure times.

I. INTRODUCTION

Network traffic generators are widely employed in computer network performance testing, simulation and for research purposes. They are able to inject packets, following a particular traffic pattern, into a network in order to test its performance or to investigate if it can reliably support a particular application in controlled environments. This goal can be reached because they are able to emulate and/or replicate the traffic generated by common network applications or traffic with specific statistical characteristics [1].

As a consequence, results obtained by tests that involve network traffic generators, are strictly dependent by the ability of the generators to accurately emulate and/or replicate the desired traffic shape or statistical pattern [2], [3].

Traffic generators are implemented over both hardware and software platforms. The former are especially designed by instrument manufacturers and implemented on dedicated high performance hardware. As a consequence they are typically more precise and reach very good performance, but expensive. In particular, those solutions are pre-configured in order to carry out a certain type of tests. On the contrary, the latter are cheaper, often open-source and/or free of charge and more flexible, but it is expected that they have lower performance in terms of accuracy and precision [4], [5].

In spite of these characteristics that seem to endorse the use of hardware-based traffic generators, the use of software-based traffic generator is widespread in research and in network performance testing. There are several reasons that justify this choice and most of them are strictly connected to their flexibility. As an example, they can be easily installed in several nodes in order to emulate a network with distributed traffic sources, or they can be updated for specific purposes adding, for example, new traffic patterns.

Certified information about the imposed values of the characteristics of the traffic generated by software-based traffic generator, such as bit rate, inter-departure time (IDT), packet rate and so on, is an extremely important need. They would be provided as the manufacturers of hardware-based traffic generators already supply with their products.

Unfortunately to certify them is a very difficult task because their metrological properties (i.e., accuracy of the traffic generation process) depend on the commercial off-the-shelf (COTS) hardware used, the operating system (OS) adopted, and the status of the host used for traffic generation [6]. Therefore without that information the reference is uncertain and consequently obtained results could be useless.

In literature this problem is investigated by considering several approaches [6]-[9], but it is not fully analyzed according to the guide for the expression of uncertainty in measurements (GUM) [10].

In this scenario, stemming from the previous experiences of the authors in network measurements [11]-[13] and in measurement accuracy evaluation of network quality of service parameters [14]-[16], aims of this paper are to analyze the factors that could influence the IDT accuracy of a software-based traffic generators and to characterize them from a metrological point of view. To this aim a well-known software-based traffic generator, Distributed Internet Traffic Generator (D-ITG) [4] has been taken into account for the experiments.

In section II brief notes on D-ITG are reported. The methodology adopted for the characterization is described in section III along with the presentation of experimental results. Conclusions are given in section IV.

II. BRIEF NOTES ON D-ITG

D-ITG [4] is a well-known tool that is able to generate IPv4 and IPv6 traffic, as well as traffic at network, transport, and application layer. D-ITG uses stochastic processes to emulate the Inter Departure Time (IDT) and Packet Size (PS) of real applications, supporting several statistical distributions for IDT and PS random variables (exponential, uniform, cauchy, normal, pareto, etc.). This approach is actually followed by a large set of traffic generators [6]. Among the two random variables, the IDT is the most sensitive to poor accuracy, being tightly dependent on the way the host running the traffic generator manages the time (process scheduling, time function resolution, etc.). In fact, a simplified version of the generation loop of D-ITG is reported in Fig. 1. As shown, the generation loop (which is repeated for every packet generated) contains different memory accesses (mem), system calls (sys), computations (cpu), and I/O requests. In particular, for every packet, D-ITG:

- fetches the current time from the Operating System (OS) using a gettimeofday() function;
- initializes some variables;
- fills the packet payload with the timestamp taken before and other information;
- pushes the packet into in the outgoing socket buffer;
- raises a signal on the serial port, if required;
- stores the log information for this packet, if required (this information is actually buffered in RAM for a number of packets, and dumped on disk periodically);
- draws the new IDT and PS using a random number generator;
- fetches again the current time from the OS to know how long it has passed since the beginning of the loop;
- waits for the remaining time before sending the new packet.

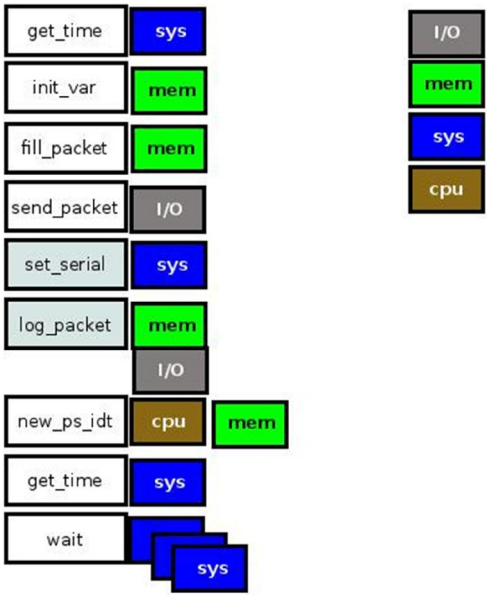

Fig.1. D-ITG simplified generation cycle.

These operations are necessary for packet generation, and this generation loop is very similar to that of other packet-level traffic generators.

III. CHARACTERIZATION METHODOLOGY AND EXPERIMENTAL RESULTS

Considering there are several sources of uncertainty in the internal generation loop of D-ITG, which are connected to a number of internal (i.e., strictly depending on the process of packet generation) and external (i.e., connected with the other ongoing processes managed by the OS) operating conditions, we decided to start by characterizing single operations, or small group of operations. The goal of the experimental activity is two-fold: understanding the contribution of each operation in the generation loop shown in Fig. 1 in the accumulation of delays that constitute the inter-departure time, along with its variability, and paving the way to the expression of the uncertainty associated to inter-departure times in the generation loop of D-ITG.

While the time needed for the completion of operations such as memory accesses, computations and I/O requests is basically deterministic and therefore has no impact on the uncertainty of inter-departure times, except for a possible systematic effect, the most critical issues, in terms of variability and predictability, are OS function calls. This is the reason why the initial experiments have been focused on those. In particular, in Fig. 1 three OS calls can be singled out: the gettimeofday() function, the set_serial() and the wait, which is basically a select() function. As the set_serial() performs an optional operation, the attention has been focused on the remaining two.

Theoretically speaking, one could be interested to evaluate the variability of the completion time of each of the two functions. However, from an operational point of view, it is difficult to characterize the functions separately. In fact, we need to timestamp packets as they go through the steps of the generation cycle, but timestamping implies further involving the OS with a new function call to the gettimeofday(), which brings on additional time contribution (and uncertainty) to the process. In other words, to characterize the select() function, we need to execute a select() and a gettimeofday(), anyway.

So, we start by analyzing the gettimeofday() and then move to the series of a gettimeofday() and a select() for different values of imposed waiting time. The approach consists in the iterated execution of the function(s) for a given number of times (ten thousand times in the first set of experiments), and is similar to the approach followed in [9]. The first-order difference of the timestamping results is then calculated, in order to achieve a vector containing the execution time of each iteration.

This way, even though we need to execute a gettimeofday() to timestamp the different executions of the select() function, we can nevertheless evaluate the relative weight of the select() function, by comparing the results of the two sets of executions.

Finally, the timestamping results of the series of the sole gettimeofday() and select() functions have been compared with those obtained when the internal generation loop of D-

ITG is executed. The goal is to experimentally verify which is the relative weight of the OS function calls in the generation loop. To achieve this, the loop in Fig. 1 (with the exclusion of the optional operations) has been executed in order to generate a CBR traffic stream with a packet rate that is consistent with the waiting time given as input to the select() function in the previous tests, and the inter-departure times have been evaluated as the difference of two successive gettimeofday() results.

The tests have been repeated for different imposed values of inter-departure times, from 200 μs to 0.1 s. Moreover, they have been performed on three different hosts, with the following hardware characteristics:

- CPU 4-core Intel(R) Xeon(R) CPU E5-1620 0 @ 3.60GHz;
- 8GB RAM;
- 1 TB SATA HD;
- Network card: Intel 82574L Gigabit Network

It is worth noting that the hosts as well as D-ITG have been configured in order to minimize the CPU resources consumption, i.e. with graphical interface inactivated, single user, local traffic generation (i.e., destination host = source host) and generation of minimum size packets.

In the first experiments, the sole gettimeofday() has been executed continuously, i.e. at the maximum rate granted by the hosts.

Table I and Table II account for the experimental results of accuracy tests performed on the generation loop, and the software traffic generator, respectively. Table I also includes the results of the execution on the sole gettimeofday(). Results are expressed in terms of average and experimental standard deviation of IDT, as well as relative difference between imposed and measured average IDT, namely Δ_{IDT}.

The first observation suggested by the results is that the relative weight of the delay introduced by the sole gettimeofday() is negligible with respect to average IDT as well as to the experimental standard deviation, at least for the packet rates considered.

The experimental results also show that for practically any packet rate, the measured IDT is greater than expected, for both experiments involving the generation loop and those involving the traffic generator. The difference Δ_{IDT} becomes relatively high for packet rates higher than 1000 pkt/s (IDT lower than 1 ms), and comparable values of Δ_{IDT} have been experienced for the same imposed packet rate.

Fig. 2 permits to compare the relative experimental standard deviation in the two set of experiments, upon the variation of the imposed packet rate. The figure shows that the

Table I. Experimental results of the accuracy test performed on the simplified generation loop composed by the OS calls *select*( ) and *gettimeofday*( ).

Imposed packet rate [pkt/s]	Imposed IDT [s]	μ_{IDT} [s]	Δ_{IDT} [%]	σ_{IDT} [s]
10	0.1	0.100125	0.13	0.000017
100	0.01	0.010066	0.66	0.000015
250	0.004	0.004061	1.52	0.000012
500	0.002	0.002061	3.03	0.000013
1000	0.001	0.001059	5.94	0.000014
2000	0.0005	0.0005631	12.61	0.0000082
3333	0.0003	0.000360	19.86	0.000010
5000	0.0002	0.0002596	29.78	0.0000059
gettimeofday		0.000000065		0.000000080

Table II. Experimental results of the accuracy test performed on the considered software traffic generator.

Imposed packet rate [pkt/s]	Imposed IDT [s]	μ_{IDT} [s]	Δ_{IDT} [%]	σ_{IDT} [s]
10	0.1	0.091000	-9.00	0.000019
100	0.01	0.0100674	0.67	0.0000043
250	0.004	0.0040658	1.65	0.0000031
500	0.002	0.002065	3.27	0.000014
1000	0.001	0.001065	6.51	0.000044
2000	0.0005	0.00057	13.05	0.00011
3333	0.0003	0.00036	20.60	0.00037
5000	0.0002	0.00026	30.95	0.00047

experimental standard deviation is of the same order of magnitude for packet rates up to 1000 pkt/s (IDT higher than 1 ms). On the contrary, for higher packet rates, the variability of the IDTs observed for the software traffic generator

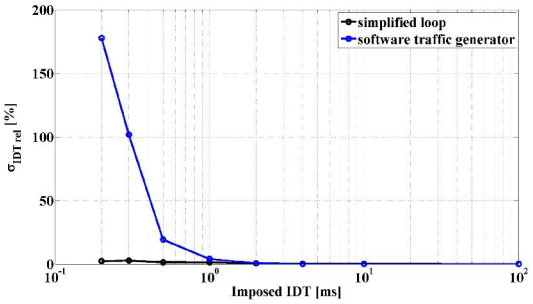

Fig. 2. Relative standard deviation of the obtained measurement results versus the imposed IDT.

becomes significantly higher, as demonstrated by the experimental standard deviation, which is one or even two orders of magnitude higher than that exhibited by the IDT of the generation loop. It looks like the set of operations performed by the traffic generator start to have a non-negligible role in the reduction of repeatability of IDTs when the imposed IDT goes under 1 ms.

The experiments have also highlighted the following phenomenon that is certainly worth being further investigated in successive tests. Very high peaks can be observed in IDT values, which are separated from each other of some tens of seconds. As it can be seen in Fig. 3, which gives details of the measured IDT values for two different packet rates, such peaks are even three orders of magnitude higher than the average IDT values. The fact that this phenomenon occurs also when the simplified loop is executed, reinforces the idea that it is due to the OS. More tests are being executed at the time when this paper was written in order to better understand and characterize this phenomenon, which certainly has an effect on the IDT uncertainty.

IV. CONCLUSIONS

The paper is the first step towards the goal of evaluating the measurement uncertainty of the inter-departure times (IDT) provided by software traffic generator. It has presented a methodology and experimental results aimed to characterize

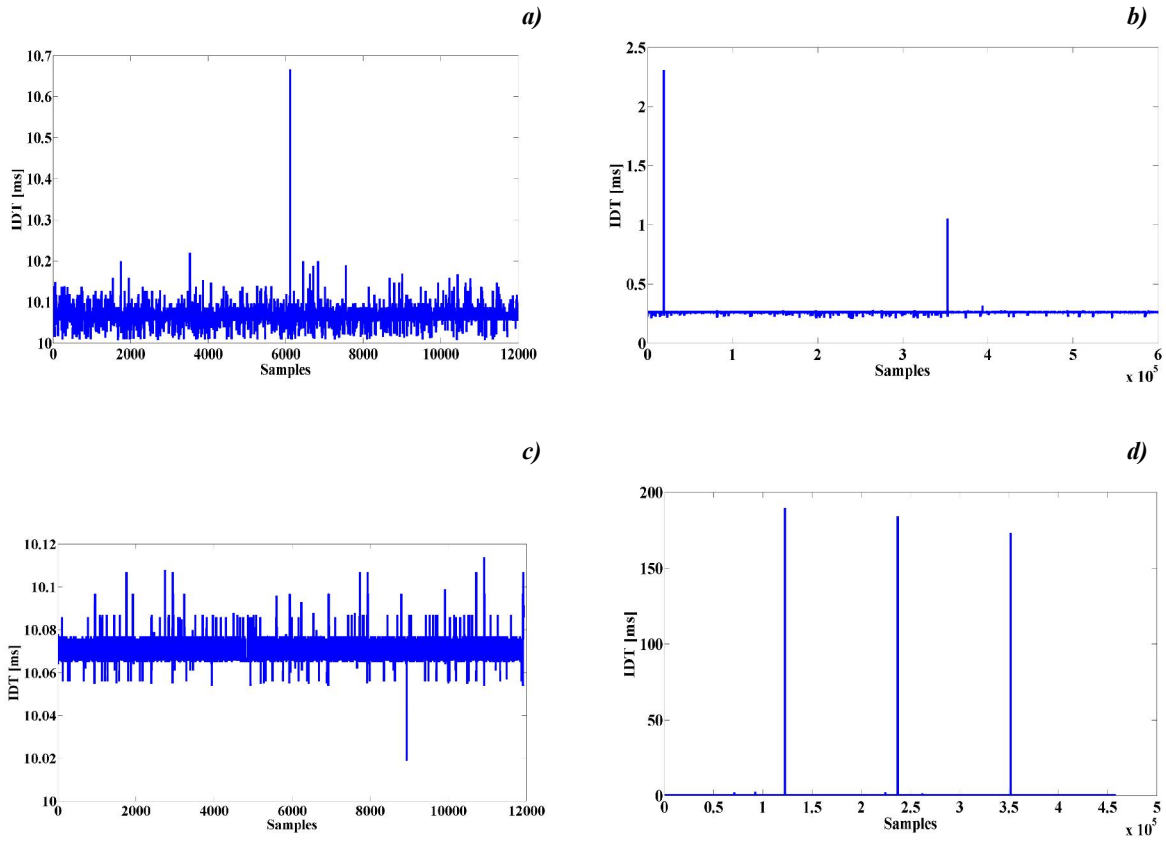

Fig. 3. Measured IDT for a),b) simplified loop, and c),d) software traffic generator.
Imposed IDT value was 10 ms for subplots on the left and 0.2 ms for subplots on the right.

the *internal* generation cycle of a well-known, open source generator, namely D-ITG, for different systems, and under the best possible conditions, i.e. with the minimum system loads.

The results have shown that for lower packet rates, the experimental standard deviation experienced for the execution of a simplified loop *select()-gettimeofday()* and for the execution of the internal loop of the software traffic generator used for the tests, are of the same order of magnitude. On the contrary, for packet rates greater than 1000 pkt/s (IDT lower than 1 ms), the IDTs measured in the tests with the software traffic generator exhibit a much higher experimental standard deviation, compared to those measured when the simplified loop is executed. This suggests that the other functionalities and operations performed by the traffic generator are responsible for a larger variability of IDTs, when the latter are lower than 1 ms. Of course, a much wider experimental campaign is needed to assess this behavior.

As regards the difference between imposed and average measured IDT, no significant difference is observed in the two cases.

Ongoing research activities are focused on extending the set of test cases at different (higher) packet rates, and analyzing the results, in order to infer the roles of the different sources of measurement uncertainty and ultimately evaluate the uncertainty of the IDT of the traffic generator, in compliance with the GUM [10]. The same approach is intended to be used for the characterization and uncertainty evaluation of other traffic generators.

REFERENCES

[1] L. Angrisani, C. Narduzzi, "Testing communication and computer networks an overview," *IEEE Instr. and Meas. Magazine*, pp. 12 24, Oct. 2008.

[2] F. Dressler, "*Policy-Based Traffic Generation for IP-BasedNetworks*," IEEE INFOCOM, Apr. 2006.

[3] K. V. Vishwanath and A. Vahdat, "Realistic and Responsive Network Traffic Generation," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 111–22, Oct. 2006.

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

[5] A. Santos, S. Fernandes, R. Antonello, G. Szabo, P. Lopes, D. Sadok, "*High-Performance Traffic Workload Architecture for Testing DPI Systems,*" 2011 IEEE Global Telecommunications Conference (GLOBECOM 2011), , vol., no., pp.1,5, 5-9 Dec. 2011.

[6] A. Botta, A. Dainotti, A. Pescapè, "Do You Trust Your Software-based Traffic Generator?", *IEEE Communications Magazine*, vol.48, no. 9, pp.158-165, Sept. 2010.

[7] M. Paredes-Farrera, M. Fleury, M. Ghanbari, "*Precision and accuracy of network traffic generators for packet-by-packet traffic analysis*," 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities TRIDENTCOM 2006, pp. 32-37.

[8] P. Arlos. *On the Quality of Computer Network Measurements*. PhD thesis, Blekinge Institute of Technology, Karlskrona, Sweden, 05:2005, Oct. 2005.

[9] K. Wac, P. Arlos, M. Fiedler, S. Chevul, L. Isaksson, R. Bults, "*Accuracy Evaluation of Application-level Performance*

Measurements," Proc. of Next Generation Internet Networks, pp. 1-5, 2007.

[10] JCGM, "*Guide 100-Evaluation of measurement data – guide to theexpression of uncertainty in measurement*," 2008.

[11] L. Angrisani, S. D'Antonio, M. Esposito, M. Vadursi, "Techniques for Available Bandwidth Measurement in IP Networks: a Performance Comparison," *Computer Networks*, Vol.50, Issue 3, 22 Feb. 2006, pp.332-349.

[12] L. Angrisani, A. Botta, A. Pescapè, M. Vadursi, "Measuring Wireless Links Capacity," *IEEE Intern. Symp. on Wireless Pervasive Computing ISWPC 2006,* 16-18 Jan 2006, Phuket (Thailand), pp.1-5.

[13] L. Angrisani, A. Pescapè, M. Vadursi, G. Ventre, "Performance measurement of IEEE 802.11b-based networks affected by narrowband interference through cross-layer measurements," *IET Communications*, vol. 2, No.1, pp. 82-91, Jan. 2008.

[14] L. Angrisani, D. Capriglione, L. Ferrigno, G. Miele, "Internet protocol packet delay variation measurements in communication networks: how to evaluate measurement uncertainty?," *Measurements*, vol. 46, no. 7, pp 2099–2109, Aug. 2013.

[15] L. Angrisani, D. Capriglione, L. Ferrigno, G. Miele, "A Methodological approach for estimating protocol analyzer instrumental measurement uncertainty in packet jitter evaluation," *IEEE Trans. on. Instr. and Meas.*, vol. 61, no. 5, pp. 1405-1416, May 2012.

[16] L. Angrisani, D. Capriglione, L. Ferrigno, G. Miele, "*Type A uncertainty in jitter measurements in communication networks*," in proc. of 2011 IEEE Instrumentation and Measurement Technology Conference (I2MTC), p. 1-5, May 2011.