Towards a Flexible and Extensible Framework for Realistic Traffic Generation on Emerging Networking Scenarios

Anderson dos Santos Paschoalon , Christian Esteve Rothenberg

Departamento de Engenharia de Computação e Automação Industrial (DCA)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (Unicamp)
Caixa Postal 6101, 13083-970 – Campinas, SP, Brasil

{pchoalon ,chesteve}@dca.fee.unicamp.br

Abstract – New emerging technologies have a larger unpredictability, compared to legacy equipment. They require a larger set of meaningful tests on many different scenarios. But, in the open source world is hard to find a single tool able to provide realism, speed, easy usage and flexibility at the same time. Most of the tools are monolithic and devoted to specific purposes. This work presents a flexible and extensible framework which aims to decouple synthetic traffic modelling from its traffic generator engine. Through a new abstraction layer, it would become possible to use modern and throughput optimized tools to create realistic traffic, in an automated way. This enables a platform agnostic configuration and reproduction of complex scenarios via analytical models. Also we use *pcap* files and live-capture to create "Compact Trace Descriptors".

Keywords – realistic, framework, traffic generation, modelling, burstiness, fractal, flow level, packet level, Hurst exponent, wavelet, pcap, emulation, stochastic, inter-departure, packet size, Swing, D-ITG, Harpoon, Swing, SourcesOnOff, LegoTG, DPDK

1. Introduction

Emerging technologies such as SDN and NFV are great promises. If succeeding at large-scale, they should change the development and operation of computer networks. But, enabling technologies such as virtualization still pose challenges on performance, reliability, and security [6]. Thus, guarantee the Service Layer Agreements on emerging scenarios is now a harder question. Applications may have a huge performance degradation processing small packets [12]. As conclude by many investigations, realistic and burstiness traffic impacts on bandwidth measurement accuracy [2]. Also, realistic workload generators are essential security research [4]. Thus, there is a demand for tests able to address realism at high throughput rates.

The open-source community offers a huge variety of workload generators and benchmarking tools [4] [9]. Each tool uses different methods on traffic generation, focusing on a certain aspects. Some traffic generator tools provide support emulation of single application workloads. But this is not enough to describe an actual Service Provider(ISP) load or even a LAN scenario. Other tools work as packet replay engines, such as TCPreplay and TCPivo. Although in that way is possible to produce a realistic workload at high rates, it comes with some issues. First, the storage space required becomes huge for long-term and high-speed traffic capture traces. Also, obtaining good traffic traces sometimes is hard, due privacy issues and fewer good sources. Many tools aim the support of a larger set of protocols and high-performance such Seagull and Ostinato. Many are also able to control inter-departure time and packet size using stochastic models, like D-ITG [4] and MoonGen. They can provide a good control of the traffic, and high rates. But, in this case, selecting a good configuration is by itself a research project, since how to use each parameter to simulate a specific scenario is a hard question [7]. It is a manual process and demands implementation of scripts or programs leveraging human (and scarce) expertise on network traffic patterns and experimental evaluation. Some tools like Swing and Harpoon, try to use the best of both worlds. Both use capture traces to set intern parameters, enabling an easier configuration. Also, Swing uses complex multi-levels which are able to provide a high degree of realism [13]. But they have their issues as well. Harpoon does not configure parameters at packet level [11] and is not supported by newer Linux kernels. Swing [13] aims to generate realistic background traffic, not offering high throughput [13] [2]. As is possible to see, this a result of the fact that its traffic generation engine is coupled to its modeling framework. You can't opt to use a newer/faster packet generator. The only way of replacing the traffic engine is changing and recompiling the original code. And this is a hard task.

This project aims to create a framework able solve many of presented issues. It must be able to "learning" patterns and characteristics of real network traffic traces. Then, using packet generators and accelerators it should reproduce a network traffic with similar characteristics. Based on observation of live captures or PCAP files, the software must choose the bests parametrized stochastic functions (from a list) to fit the data. These parametrized stochastic functions along with collected header features (such as protocols and addresses) will be record in a machine-readable file (such as XML or JSON) we baptise as a Compact Trace Descriptor (CTD). This data must serve as input for an API of a traffic generator. So it will be possible to control packet parameters, and flow's behaviours, through different APIs. Also, and speed-up may be achieved, though the use of DPDk's KNI interfaces[1]. So, the main goal is to offer an easier configuration, realism, at a higher speed than the available platforms today. Also, it will add programmability and abstraction to the traffic generation, since the user may edit or create a custom traffic descriptor in a platform agnostic way. The the intermediate layer of the figure 1 summarize, goal of the project in an illustrative way.

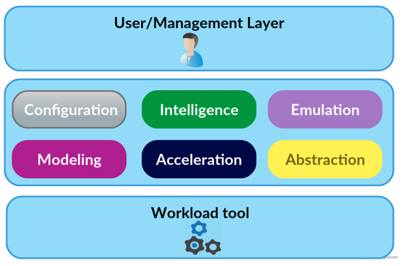

Figure 1. Proposal representation in a layer diagram. It automates features such as configuration, modelling, and parametrization, intelligence, emulation, and abstraction through an additional layer.

2. Literature review and related work

A common taxonomy used on traffic generator tools based on the layer of operation. It divides them into four categories [3]:

Application-level/Special-scenarios traffic generators: they emulate applications behaviours, through stochastic and responsive models. Eg.: Surge, GenSym.

Flow-level traffic generators: they emulate features of the flow level, such as file transference and bursts. But do not model applications or packet behaviour.Eg.: Harpoon.

Packet-level traffic generators: they model inter-departure time and packet size through stochastic distributions. They focus on performance testing. Eg: D-ITG [4], Ostinato, Seagull, TG.

Multi-level traffic generators: These traffic generators models each mentioned layer, describing user and network behaviour. They generate an accurate background traffic, but usually, have bottlenecks on bandwidth. Eg.: Swing [13].

Varet et al. [1] creates an application in C, called SourcesOnOff. It models the activity interval of packet trains using probabilistic distributions. To choose the best stochastic models, the authors captured many traffic traces using TCPdump. Then, they figure out what distribution (Weibull, Pareto, Exponential, Gaussian, etc.) fits better the original traffic traces, using the Bayesian Information Criterion. For this task, they choose the function with the smaller BIC. The smaller this value is, the better the function fits the data.

Bartlett et al. [2] implements a modular framework for composing custom traffic generation. Its goal is making easy the combine of different traffic generators and modulators in different test-beds. It automatizes the process of installation, execution, resource allocation and synchronization using a centralized orchestrator and a software repository. It already has support to many tools, and to add support to new tools is necessary to add and edit two files, called TGblock, and ExFile.

3. System Architecture

We developed and architecture that solves the listed issues. It has five components: a Sniffer, an SQLite database, a Trace Analyzer, a Flow Generator, and a Network Traffic Generator as a subsystem.

The Sniffer is responsible collecting data from the network traffic. It extracts data from the packets and stores them in the database. This information can be protocols, packet size, inter-arrival time, flows, and so on. Also, after finishing a capture, this component is the responsible for providing data visualization. It may work over a PCAP file, or over an Ethernet interface. The prototype version of this component uses tsahrk to capture packets and Shell/Octave scripts. To improve performance and avoid bottlenecks, next implementation

[1] http://dpdk.org/doc/guides/sample_app_ug/kernel_nic_interface.html

will use libtins librarie for packet processing. The criteria to classify the traffic into flows is the same of SDN switches: internet protocol, source/destination addresses, transport protocol, and source/destination transport ports. The framework uses an SQLite database.

The Trace Analyzer is the core of the project. It is the tool responsible for characterizing the trace. Using the stored information, breaks the trace into flows, and parametrize each of them. The parameters are header fields and stochastic functions/coefficients for each flow. The component models the behaviour of the trace on flow level and packet level. At the packet level, is possible to model the packet-size and the inter-departure time, during packet bursts (ON times). At the flow level, is possible to control bursts periods, session length, and the number of bytes delivered. We will use likelihood criterions to choose the best probabilistic function and parameters. Options are the smaller error, Akaike information criterion, and Bayesian information criterion [1]. It will sort the parametrized functions in a priority list. After the parametrization, the Trace Analyzer records these features in a machine-readable file (XML, JSON) called "Compact trace descriptor".

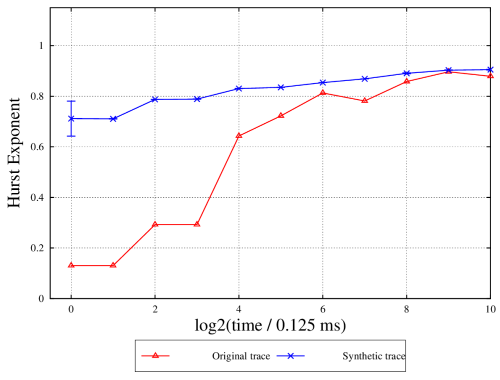

Figure 2. Hurst exponent value of original and synthetic traces

The Flow Generator pick these abstract parameters and feed an Ethernet workload generator tool. It crafts each flow in an independent way, in a different thread. The presented prototype just uses the D-ITG API as workload tool. But it can use any packet-level traffic generator with API or CLI. This component handles the flow level models and parametrizes the packet-level tool underneath. Since each packet-level tool supports a different set of stochastic functions, the Flow Generator should pick the first compatible model from the priority list. But prototype presented here still uses simple models on packet and flow crafting, supporting just constant distributions. But the next release should support at packet-level heavy-tailed [1] and Poison functions for the inter-departure times, and bimodal distributions [5] [10] for the packet size. At the flow level, two different alternatives can be used. Model file transference and session, such as in Harpoon [11]; or use an envelope process, as suggested by Melo et al [8].

4. Partial results

To as proof of concept, we propose a set of tests. We choose them, based on tests used to ensure realism, on related many works [1] [13] [2]. They aim to ensure realism and similarity. Realism tests measure if a synthetic traffic has expected features of an Ethernet capture. Similarity tests measures if the generated traffic represents specific characteristics of the original one. Here, due the limited space, we will present just two results. The first, which test realism, is the Hurst exponent evaluation. It is able to test the self-similarity of the generated traffic. To be self-similar, a process must have a Hurst exponent between 0.5 and 1 [7]. Also, usual values of Ethernet traffic lay between 0.8 and 0.9 [7].

Thus a realistic Ethernet traffic must have a Hurst exponent close to the last interval. The second test is Wavelet Multiresolution Energy Analysis. It is able to capture characteristics of the traffic at different time-scales. For example, it enables visualization of a periodic tendency(decrease) or a self-similar tendency(increase) at a certain time scale. Also, at each point, it represents the mean energy of that signal at that time scale. So, similar Ethernet traffics must have slopes at close time-scales. Also, they must have close energy scales. More close are the curves, more similar are the traces.

The evaluated prototype support just constant functions. It selects the inter-departure time equal to the mean. The packet size is set as the most frequent value. The flow's start time and duration are the same from the observed traffic. We capture the original traffic trace on the laboratory LAN. The results are at figures 2 and 3. On both analysis, the generation of the synthetic trace was repeated 30 times. The keys which serves as input to D-ITG were randomly selected. At is possible to see that the on both cases the Hurst exponent converge to the same value, close to 0.9. But, on the wavelet multiresolution analysis, both curves still different behaviours.

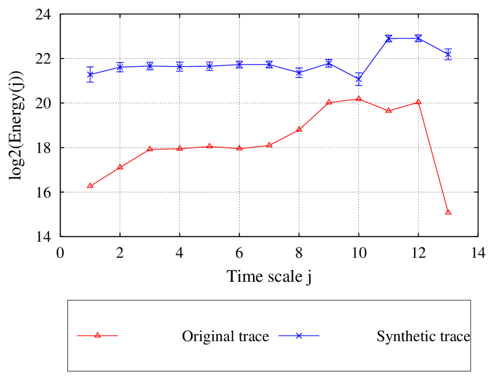

Figure 3. Wavelet Multiresolution Energy Analysis of the original and synthetic traces

5. Conclusion and future work

The framework prototype was already able to generate a realistic (self-similar) Ethernet traffic. But, as expected, is still unable to represent well the particular features of the original traffic trace. This is a result of the, still, poor stochastic modelling. The next job will be implementing a significative modelling of the original traffic trace, through the specified methodology. We expect more significant results, them. Also, we will expand the framework to others workload platforms and compare the results. Packet acceleration could speed-up the performance, which may enable the reproduction of high-throughput traces. This can be implemented using DPDK. Finally, the results should be compared to Swing, on realism, similarity, and performance. This will give a measurement of how good is the framework, compared with others alternatives, and its strong and weak points.

References

[1] Nicolas Larrieu Antoine Varet. Realistic network traffic profile generation: Theory and practice. *Computer and Information Science*, 7(2), 2014.

[2] G. Bartlett and J. Mirkovic. Expressing different traffic models using the legotg framework. In *2015 IEEE 35th International Conference on Distributed Computing Systems Workshops*, pages 56–63, June 2015.

[3] A. Botta, A. Dainotti, and A. Pescape. Do you trust your software-based traffic generator? *IEEE Communications Magazine*, 48(9):158–165, Sept 2010.

[4] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. A tool for the generation of realistic network workload for emerging networking scenarios. *Computer Networks*, 56(15):3531 – 3547, 2012.

[5] Ewerton Castro, Ajey Kumar, Marcelo S. Alencar, and Iguatemi E.Fonseca. A packet distribution traffic model for computer networks. In *Proceedings of the International Telecommunications Symposium – ITS2010*, September 2010.

[6] Bo Han, V. Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *Communications Magazine, IEEE*, 53(2):90–97, Feb 2015.

[7] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, Feb 1994.

[8] Cesar A.V. Melo and Nelson L.S. da Fonseca. Envelope process and computation of the equivalent bandwidth of multifractal flows. *Computer Networks*, 48(3):351 – 375, 2005. Long Range Dependent Traffic.

[9] S. Molnár, P. Megyesi, and G. Szabó. How to validate traffic generators? In *2013 IEEE International Conference on Communications Workshops (ICC)*, pages 1340–1344, June 2013.

[10] L. O. Ostrowsky, N. L. S. da Fonseca, and C. A. V. Melo. A traffic model for udp flows. In *2007 IEEE International Conference on Communications*, pages 217–222, June 2007.

[11] Joel Sommers, Hyungsuk Kim, and Paul Barford. Harpoon: A flow-level traffic generator for router and network tests. *SIGMETRICS Perform. Eval. Rev.*, 32(1):392–392, June 2004.

[12] S. Srivastava, S. Anmulwar, A. M. Sapkal, T. Batra, A. K. Gupta, and V. Kumar. Comparative study of various traffic generator tools. In *Engineering and Computational Sciences (RAECS), 2014 Recent Advances in*, pages 1–6, March 2014.

[13] K. V. Vishwanath and A. Vahdat. Swing: Realistic and responsive network traffic generation. *IEEE/ACM Transactions on Networking*, 17(3):712–725, June 2009.