Designing Repeatable Experiments on an Emulab Testbed

Andres Perez-Garcia, Christos Siaterlis, and Marcelo Masera

Institute for the Protection and Security of the Citizen
Joint Research Centre
Via E. Fermi 2749, 21027 Ispra (VA) Italy
{andres.perez-garcia,christos.siaterlis,marcelo.masera}@jrc.ec.europa.eu

Abstract. Emulation testbeds are increasingly used in an effort to promote repeatable experiments in the area of distributed systems and networking. In this paper we are studying how different design choices, e.g. use of specific tools, can affect the repeatability of experiments of an emulation testbed (e.g. based on the Emulab software).

Our study is based on multiple experiments that are checked for stability and consistency (e.g., repetition of the same experiment and measurement of the mean and standard deviation of our metrics). The results indicate that repeatability of quantitative results is possible, under a degree of expected statistical variation. The event scheduling mechanism of Emulab is proven to be accurate down to a sub-second granularity. On the other hand we demonstrate that there are significant differences between traffic generation tools in terms of consistent recreation of a predefined traffic pattern and therefore experiment repeatability.

The main contribution of this study is that based on experimental results we provide scientific proofs that Emulab as a platform can be used for scientifically rigorous experiments for networking research. New users of Emulab can benefit from this study by understanding that Emulab's scheduling mechanism, it's built-in packet generators and Iperf can sufficiently support repeatable experiments while TCPreplay cannot and therefore an alternative tool, i.e. TCPivo should be used.

Keywords: emulation, network test-bed, repeatability, traffic generators.

1 Introduction

Emulation testbeds are increasingly used in an effort to address the lack of scientific rigor [1] and realism as well as to promote repeatable experiments in the area of distributed systems and networking [2]. The study of complex systems or system of systems, e.g., the Internet, could be carried out by experimenting with real systems, software simulators or hardware emulators. Experimentation with real production systems suffers from the inability to control the experiment environment in order to reproduce results. Furthermore if the study intends

I. Tomkos et al. (Eds.): BROADNETS 2010, LNICST 66, pp. 28–39, 2012.

to test the resilience or security of a system, concerns about potential side-effects (faults and disruptions) to mission critical services rise. On the other hand the development of a dedicated experimentation infrastructure with real components is often economically prohibitive. Software based simulation would then appear as the best solution but due to the diversity and complexity of protocols, systems and architectures of the Internet hardware-based emulation is considered a flexible and powerful approach [3]. Indeed, emulation approaches and specifically those based on the Emulab software are becoming very popular [4]. Emulab is a network testbed, able to recreate a wide range of experimentation environments in which researchers can develop, debug and evaluate a complex system [5]. Emulation is particularly useful for security and resilience analysis [6],[7], because in order to study resilience a researcher has to expose the system-under-test to high load and extreme conditions.

In this paper we present a study of different parameters that might influence the repeatability of experiments on top of an Emulab-based testbed. We show that experimental results can be systematically reproduced (even in absolute numbers -quantitative- given a fixed hardware configuration). Furthermore we study the event scheduling system of Emulab as an important mechanism for repeating experiments as well as different traffic generating tools. To the best of our knowledge, previous studies compare different emulation/simulation approaches [8],[9] rather than systematically studying the repeatability of experiments by comparing different runs of the same experiment. In this paper, we take advantage of the automation functionality of Emulab in order to run multiple experiments (hundreds) and draw our conclusions after checking the experimental results for stability and consistency. Our contribution does not only lie on the presented experimental results but also in the transformation of our experience in terms of caveats, significant configuration parameters and limitations into a set of guidelines that researchers using Emulab could use as a reference. This would lower the barrier for new researchers trying to use Emulab and promote scientific rigorous experimentation.

The paper is structured as follows. We begin in Section 2 with a description of an Emulab-based testbed and its characteristics. Then we proceed in Section 3 with our study and experimental results. In Section 3.1 we present the experimental setup that is used in our experiments and in the following subsections we address how the repeatability of experiments can be influenced by the hardware allocation policy, Emulab's event scheduling mechanism and the use of various traffic generators. We conclude in Section 4 and summarize our findings.

2 The Emulab Platform and Its Features

One of the most promising approaches for experimentation with large and complex systems, e.g. those found in an industrial Supervisory Control and Data Acquisition (SCADA) network [10], is the use of emulation testbeds. Pure software simulation is often too simplistic to recreate complex environments and the use of an ad-hoc testbed is not recommended because it is very time-consuming

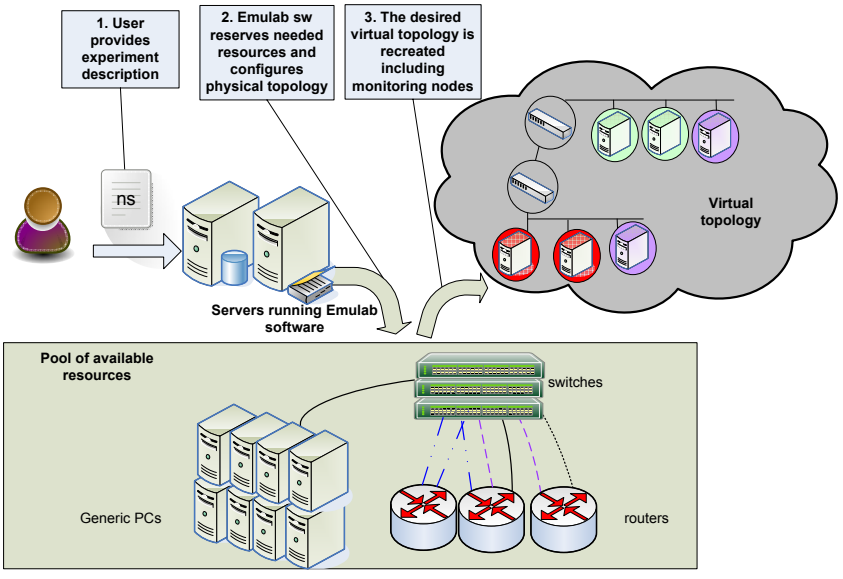


Fig. 1. Main steps for recreating a virtual network configuration within an Emulab-based testbed

and error-prone to setup, maintain and change. A trend, that is constantly becoming more popular, is the use of emulation testbeds like Emulab [5]. We have developed in our laboratory a testbed using the Emulab architecture and software, that allows us to automatically and dynamically map physical components (e.g. servers, switches) to a virtual topology. In other words the Emulab software configures the physical topology in way that it emulates the virtual topology as transparently as possible. This way we gain significant advantages in terms of repeatability, scalability, controllability and automation of our experiments.

Our emulation testbed consists mainly of two servers running the Emulab software and a pool of physical resources (e.g. generic PCs and network devices) that are free to be used as experimental nodes. The following steps (Figure 1) describe the re-creation of a virtual network configuration within our testbed:

1. First we need to create a detailed description of the virtual network configuration using an extension of the NS language [11] (the experiment script).
2. In our description we enumerate similar components as different instances of the same component type. This way pre-defined templates of different components (e.g a Linux server template) can be easily reused and automatically deployed and configured.
3. Whenever we want to run an experiment we instantiate it by using the Emulab software. The Emulab server automatically reserves and allocates the physical resources that are needed from the pool of available components.

This procedure is called *swap-in*, in contrast to the termination of the experiment which is called *swap-out*.
4. Furthermore the software configures network switches in order to recreate the virtual topology by connecting experimental nodes using multiple VLANs.
5. Finally, before the testbed is released for experimentation, the software configures packet capturing of predefined links for monitoring purposes.

At this point it is important to note that in step 4, the Emulab software uses two different strategies for network link emulation (e.g., delay, packet loss and bandwidth) according to the predefined instructions given in the experiment script. First, the *delay-node-shaping strategy* uses extra PCs to emulate network links. These PCs, called delay nodes, run Dummynet to simulate link level characteristics [12]. Second, the *end-node-shaping strategy* does not use extra resources and therefore runs Dummynet inside the end user nodes. In this paper we don't use the end-node-shaping strategy as it can lead to unstable and unrealistic results [13].

To achieve repeatable experiments on an Emulab testbed (i.e. the ability to repeat an experiment and obtain the same or statistically consistent results) a controlled environment is needed. In this paper we study how different tools and mechanisms can influence the repeatability of experiments and specifically:

1. *The hardware allocation strategy*, that matches physical resources, i.e., PC's and network links, to the virtual topology (at step 3). Emulab can use three different strategies:
   - the *fixed-hardware-allocation strategy* where all experimental nodes are matched with a specific hardware (e.g. a user node is always instantiated by PC10);
   - the *fixed-class-allocation strategy* where all experimental nodes are matched with a hardware of a specific class (e.g. a user node is always instantiated by a P4x2GHz);
   - the *free-hardware-allocation strategy* where experimental nodes are freely matched with any available hardware. We should note here that fixing the allocation of delay nodes is not possible.
2. *The event generation system*, that allows the researcher to schedule events. These events are an integral part of any experiment scenario. To reproduce a previously stored experiment scenario the researcher should be able to setup the experimental platform in the initial state and trigger all necessary events in the right order and time of occurrence.
3. *The traffic generating tools* and their ability to consistently reproduce the same background traffic environment. We consider two classes of tools: a) synthetic traffic generators (Iperf and emulab-buildin tools) b) tools that replay real traffic captures (Tcpreplay [14] and Tcpivo [15]).

3 Experimental Results

Based on a series of experiments, we study Emulab as a platform to conduct rigorous experiments in terms of repeatability. First we present our experimental

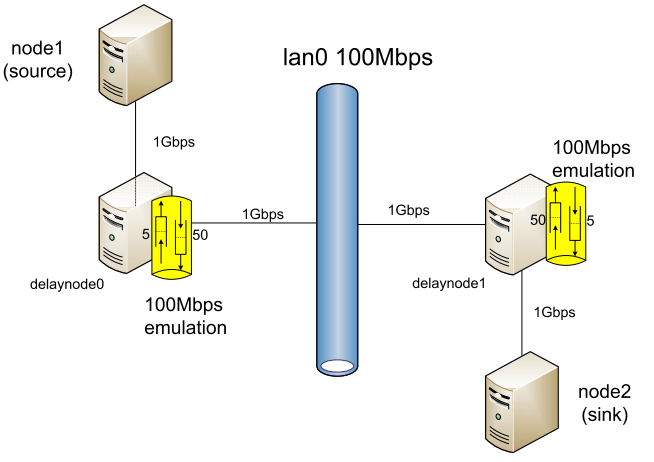


Fig. 2. The experimental setup that was used in our experiments

setup and then in the sections that follow, we present results demonstrating how different tools and mechanisms can influence repeatability.

3.1 Experimental Setup

We experiment with a topology that consists of a 100Mbps LAN with two user nodes. As we use the delay-node-shaping strategy, Emulab instantiates delay-nodes running Dummynet in order to model the network (Figure 2). Dummynet is configured with two pipes, inbound and outbound, to shape traffic entering and leaving a user node. In our experiments, inbound and outbound pipes have a queue size of 5 and 50 slots respectively. Physically, all interfaces in our testbed are configured at 1Gbps and it is left up to Dummynet to shape the traffic at the speed of our virtual topology, i.e. 100Mbps.

We have used two types of nodes in order to study the repeatability with different hardware. Experimental nodes are either Dell PC's with AMD 2GHz Athlon processor and 2GB RAM, or Fujitsu PC's with Intel PIV processor and 1GB RAM. As for the operating system, we have used both FreeBSD and Linux Fedora Core as user nodes, depending on the application, while only FreeBSD in delay-nodes.

The following tools have been used to launch the experiments and to collect and analyze experimental data:

- Iperf [16] is a tool to generate UDP traffic between a source node and a sink node. It also includes a build-in measurement functionality that provides statistics such as sustained bandwidth and packet loss, that we use to assess the network's performance.
- The CBR traffic generator that comes with Emulab works similar to Iperf. It does not provide statistics, but it is easier to use and schedule in the NS script.

- TCPReplay [14] uses a previously captured traffic file in libpcap format and replays it back onto the network, usually to test switches, routers and firewalls. It is a powerful tool that allows to classify traffic as client or server and rewrite Layer 2, 3 and 4 headers.
- TCPivo [15] is another, less known, free and open-source tool that supports high-speed packet replay from a trace file.

In order to run our experiments, we have made use of Emulab's potential to automatically launch scripts that configure and run the different experiments and store statistics in a repository for further analysis. This has allowed us to launch thousands of experiments in a short period of time without human interaction.

We have set up three sets of experiments to study how repeatability can be influenced by a) the hardware allocation strategy, b) the event generation system, and c) the use of different traffic generators. In the following sections we investigate these factors one by one.

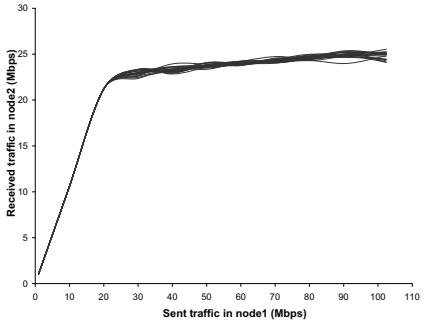
3.2 Hardware Allocation Strategies

We have already demonstrated in [13] that the quantitative results of an Emulab-based experiment are hardware-dependent. On the other hand, the hardware allocation might change from one experiment to another due to (un)availability of resources, randomness in Emulab's swap-in algorithm or other testbed policies. For this reason, we have performed a set of experiments in order to study the influence of hardware allocation in the repeatability of an experiment. We measure the network performance, i.e. the traffic received by the sink node versus the traffic sent by the source node, along repetitive experiments of three distinct experiment sets corresponding to the three hardware-allocation-strategies:

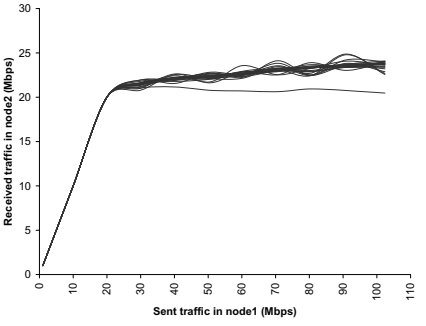
1. the *fixed-hardware-allocation strategy*, where each experimental node is fixed to a specific PC;
2. the *fixed-class-allocation strategy* where experimental nodes are chosen from the class of Dell PCs;
3. the *free-hardware-allocation strategy* where experimental nodes are freely chosen from the two hardware classes, i.e., Dell and Fujitsu PCs.

In all experiments, we measured the sustained network performance using Iperf's built-in measurement functionality. We generated UDP traffic from node1 to node2 with 512 bytes of payload and bandwidth ranging from 0Mbps up to 100Mbps. In the first experiment set we did not swap in and out in order to preserve the exact hardware allocation (not even a change in delay nodes), while in the other two experiment sets we swapped in and out for each experiment, leaving Emulab to freely choose the hardware allocation according to the predefined strategy.

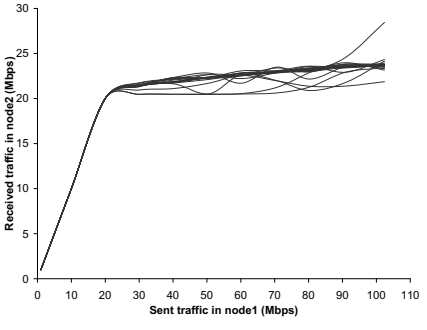
For each experiment set we run the same experiment 20 times and the results are depicted in Figures 3, while the statistics, average μ, standard deviation σ and coefficient of variation ($CV = \frac{\sigma}{\mu}$), are shown in Figure 3(d). From the figures



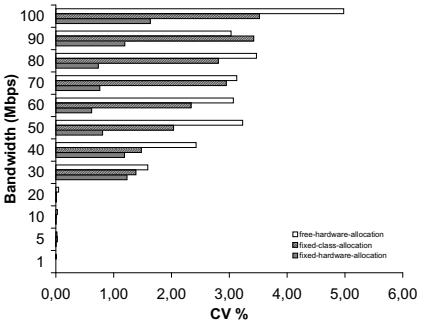
(a) Different runs using fixed-hardware-allocation strategy.



(b) Different runs using fixed-class-allocation strategy.



(c) Different runs using free-hardware-allocation strategy.



(d) Coefficient of variation.

Fig. 3. Repeatability of network's performance in three different strategies of hardware allocation

we can see that the 20 experiments in each case provide the same performance when traffic is under 20Mbps, i.e. when there are no packet losses. However, when Dummynet is not able to process all the packets and there are drops in the queues, we see that the hardware allocation introduces a higher variability in the results as we go from a fixed to a free allocation strategy.

In fact, if we look at the CV, which in general gets worse as the bandwidth grows, the best results in terms of repeatability are with the fixed-hardware-allocation strategy and the worst results with the free-hardware-allocation strategy. Another important observation is that even in the worst case the CV is under 5%, i.e., the maximum CV for the three allocation strategies is 1.63%, 3.52% an 4.98% respectively. This means that in experiments of moderate network load (where hardware dependence is not critical) even a free allocation strategy can result repeatable results.

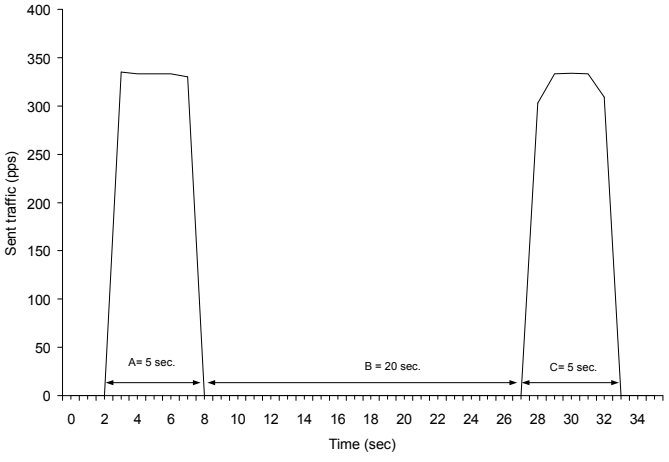


Fig. 4. CBR traffic scheduled in the NS script to be sent in node3

3.3 The Event Geeration System

In this section we study the accuracy of the event generation system in Emulab. In order to do that, we have run the same experiment 20 times (swapping in and out). In the experiment NS script we have scheduled four events to generate 2 pulses of CBR traffic from node1 to node2. Each pulse (events A and C) has a duration of 5 seconds, and the time between them (event B) is 20 seconds (Figure 4). The information registered by Emulab about event generation in each experiment is precise and consistent with the configuration.

We have captured the traffic with Tcpdump in delaynode0 and we have measured the time between the first and last packet of each pulse for the 20 experiments. Table 1 shows the statistics of the duration of events A, B and C along the 20 experiment runs. The standard deviation is always below 63 milliseconds, which should be precise enough for most experiments that schedule events in seconds or tens of seconds. In terms of CV, we see that the accuracy is better with longer periods.

This variation can be explained by looking at what happens after the events are scheduled by the system and before the traffic is captured. The events imply starting or stopping an application (CBR traffic generator) in a remote node, so there is a communication between the Emulab system and node1. Then, the traffic arriving to the delaynode0 has to pass through network cards, switch

Table 1. The average, standard deviation and coefficient of variation of the duration of events A, B, C

Event	Avg	Stdev	CV %
A	4.94	0.03	0.55%
B	20.08	0.05	0.25%
C	4.86	0.06	1.30%

and cables before it is captured. All these processes along with CPU scheduling inaccuracy result some time shifting. The conclusion is though that Emulab's event generation system is accurate and consistent.

3.4 Traffic Generators

In this part of the study, we have analyzed the repeatability of different traffic generators, namely Iperf, CBR, Tcpreplay and Tcpivo, by running each of them 10 times with the same configuration. For Iperf and CBR we generated pulses of 30 seconds with UDP packets of 512bytes of payload from node1 to node2 (synthetic traffic), while for Tcpreplay and Tcpivo, we reproduced a real trace of 30 seconds with TCP packets of random length (taken from the DATCAT repository [17]).

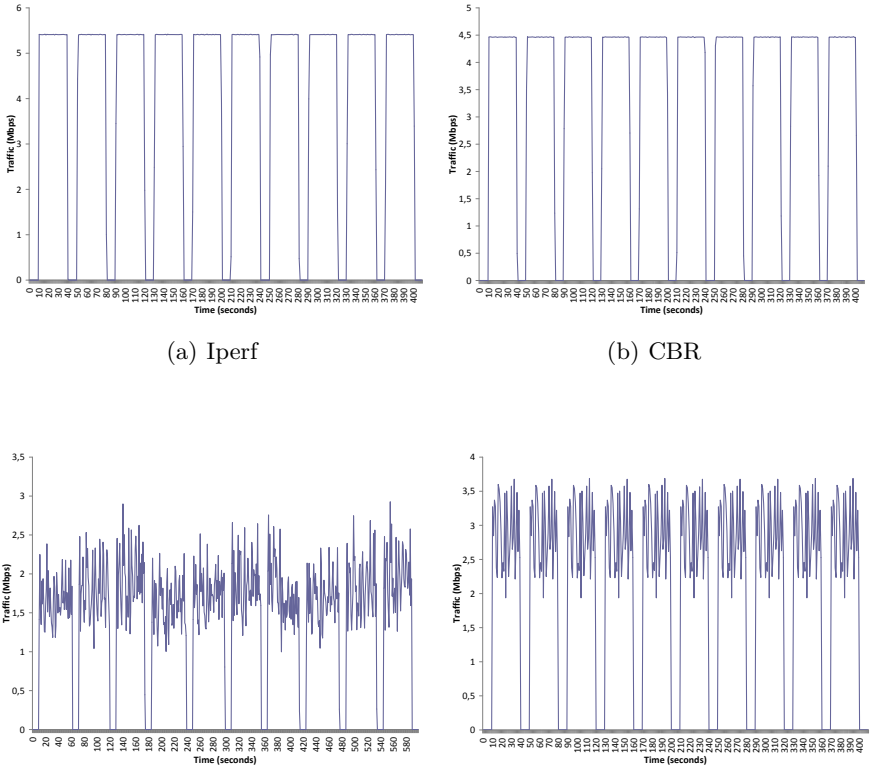
Figure 5 shows the traffic measured with Tcpdump in node2 for each of the traffic generators. At first sight, we see that tcpreplay is not able to reproduce a single trace with the same characteristics: each reproduction is different from the other. On the other hand the rest of the tools seem to generate traffic in a repeatable way. In fact, if we look at Table 2, the duration of traffic is practically the same for all the tools but Tcpreplay, where the CV is higher than 6% and the standard deviation is $\sigma = 3.27$ seconds. Furthermore we have subtracted the generated traffic signal that was produced by Tcpreplay and Tcpivo as measured in node2, i.e., $gen(t)$, from the original reference signal of the trace file we used as input to both tools i.e., $ref(t)$ and depicted the difference $ref(t) - gen(t)$ in Figure 6. We see that the differences are in the order of few Kbps in the case of Tcpivo, but it reaches up to 2Mbps with Tcpreplay. A small difference was expected due to the buffering mechanism in the network, but Tcpreplay wasn't able to provide repeatable results.

Table 2. Repeatability of traffic generators (traffic duration)

Tool	Avg	Stdev	CV %
Tcpreplay	48.45	3.27	6.74%
TCPivo	30.00	0.00	0.00%
CBR	30.07	0.02	0.05%
Iperf	30.00	0.00	0.00%

4 Conclusion

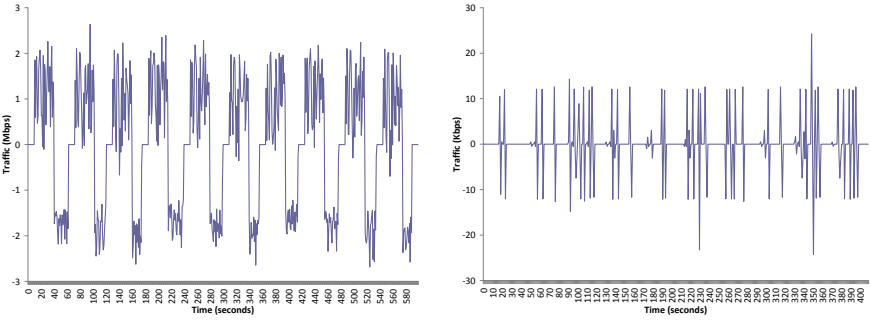
The study of complex systems and networks is a hard process. The limitations of software simulators and theoretic modeling as well as the required cost and effort to setup and maintain ad-hoc testbeds of real systems, make the use of emulation testbeds a promising approach. Emulation testbeds like Emulab are increasingly used in networking research in an effort to raise the level of scientific rigorousness and specifically by conducting repeatable experiments. In this paper we investigate how different mechanisms and tools of an Emulab testbed can influence the repeatability of experimental results.



(a) Iperf (b) CBR

(c) TCPReplay (d) TCPivo

Fig. 5. Traffic generated by different tools and measured in the sink node



(a) $ref(t) - gen(t)$ for TCPReplay (b) $ref(t) - gen(t)$ for TCPivo

Fig. 6. Subtraction between traffic signals of the reference trace file $ref(t)$ and the generated traffic $gen(t)$ by TCPReplay and TCPivo

Our contribution can be summarized in the following points. We confirm that repeatability of quantitative experimental results on an Emulab testbed is possible, under a degree of expected statistical variation. If the *fixed-hardware-allocation strategy* is used the coefficient of variation (CV) of the results lies on average under 1%. Furthermore the event scheduling mechanism of Emulab is proven to be accurate down to a sub-second granularity, ensuring thus a reliable and accurate reproduction of an experiment script. Finally we demonstrate that there are significant differences between traffic generation tools in terms of consistent recreation of a predefined traffic pattern and therefore repeatability. For synthetic traffic Emulab's built-in packet generator as well as Iperf were proven as adequate. As for replaying real traffic traces, our results show that only the less known TCPivo tool can guarantee repeatability whereas the popular TCPreplay tool fails to do so. In general this work could be seen as an effort, part of a general trend of the networking research community, towards the execution of repeatable experiments, i.e., to results that can be reproduced and validated by other researchers.

References

1. Pawlikowski, K., Joshua Jeong, H.d., Ruth Lee, J.s.: On credibility of simulation studies of telecommunication networks. IEEE Communications Magazine 40, 132–139 (2002)
2. Benzel, T., Braden, R., Kim, D., Neuman, C., Joseph, A., Sklower, K., Ostrenga, R., Schwab, S.: Design, deployment, and use of the deter testbed. In: DETER: Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007, p. 1. USENIX Association, Berkeley (2007)
3. Neville, S.W., Li, K.F.: The rational for developing larger-scale 1000+ machine emulation-based research test beds. In: International Conference on Advanced Information Networking and Applications Workshops, pp. 1092–1099 (2009)
4. Emulab Bibliography, `http://www.emulab.net/expubs.php/`
5. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. In: Proc. of the Fifth Symposium on Operating Systems Design and Implementation, pp. 255–270. USENIX Association, Boston (2002)
6. DETER. cyber-DEfense Technology Experimental Research laboratory Testbed, `http://www.isi.edu/deter/`
7. Mirkovic, J., Hussain, A., Fahmy, S., Reiher, P.L., Thomas, R.K.: Accurately measuring denial of service in simulation and testbed experiments. IEEE Trans. Dependable Sec. Comput. 6(2), 81–95 (2009)
8. Anderson, D.S., Hibler, M., Stoller, L., Stack, T., Lepreau, J.: Automatic online validation of network configuration in the emulab network testbed. In: ICAC 2006: Proceedings of the 2006 IEEE International Conference on Autonomic Computing, pp. 134–142. IEEE Computer Society, Washington, DC (2006)
9. Chertov, R., Fahmy, S., Shroff, N.B.: Fidelity of network simulation and emulation: A case study of tcp-targeted denial of service attacks. ACM Trans. Model. Comput. Simul. 19(1), 1–29 (2008)

10. Guglielmi, M., Fovino, I.N., Garcia, A.P., Siaterlis, C.: A preliminary study of a wireless process control network using emulation testbed. In: Proc. of the 2nd International Conference on Mobile Lightweight Wireless Systems. ICST, Barcelona (2010)
11. ISI, Network simulator ns-2, http://www.isi.edu/nsnam/ns/
12. Rizzo, L.: Dummynet: a simple approach to the evaluation of network protocols. SIGCOMM Comput. Commun. Rev. 27(1), 31–41 (1997)
13. Andres Perez Garcia, M.M., Siaterlis, C.: Testing the fidelity of an emulab testbed. In: Proc. of the 2nd workshop on Sharing Field Data and Experiment Measurements on Resilience of Distributed Computing Systems, Genova, Italy (June 2010)
14. Turner, A.: Tcpreplay tool, http://tcpreplay.synfin.net/trac/
15. Feng, W.-C., Goel, A., Bezzaz, A., Feng, W.-C., Walpole, J.: Tcpivo: a high-performance packet replay engine. In: MoMeTools 2003: Proceedings of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research, pp. 57–64. ACM, New York (2003)
16. NLANR/DAST, Iperf: The TCP/UDP bandwidth measurement tool, http://sourceforge.net/projects/iperf/
17. Cho, K.: WIDE-TRANSIT 150 Megabit Ethernet Trace 2008-03-18 (Anonymized) (collection), http://imdc.datcat.org/collection/1-05L8-9=WIDE-TRANSIT+150+Megabit+Ethernet+Trace+2008-03-18+%28Anonymized%29