Network Testing Using a Novel Framework for Traffic Modeling and Generation*

Oluwamayowa Ade Adeleke
Computer Science
University of Houston
Houston, USA
oaadelek@central.uh.edu

Nicholas Bastin
Engineering Technology
University of Houston
Houston, USA
nick.bastin@gmail.com

Deniz Gurkan
Engineering Technology
University of Houston
Houston, USA
dgurkan@uh.edu

***Abstract*—Network traffic modeling plays an important role in the generation of realistic network traffic in test environments. Especially in cases where researchers carry out experiments with real production-like traffic, as seen in specific home, enterprise, campus, LAN, or WAN networks. We present our ongoing work on a new framework that enables the methodical creation of application-agnostic traffic models from given network traces of a known network topology. The framework then uses these models to generate realistic traffic on a given network topology. We share a preliminary evaluation of the framework based on repeatable experiments where we model a typical web application traffic and then regenerate the traffic based on the model in a test network on our VTS (Virtual Topology Services) testbed.**

***Keywords*—Computer networks; Packet generators; captures; Network Traffic Modeling; Machine Learning; GENI; VTS**

I. Introduction

Software Defined Networking (SDN) has enabled creation of fairly complex test networks with several nodes and links in testbed environments[5]. However, the process for obtaining production network traffic data for testing novel ideas, algorithms, protocols and network functions remains a major pain point for many academic researchers. Privacy policies restrict some of the data that industry operators can share with third parties[6]. Traffic replay can be limited to the capture point perspective [4]. Therefore, many researchers rely on synthetic traffic generators [3]. At best, many popular traffic generators blast out packets at predetermined rates, or at rates based on statistical distributions of a few traffic parameters. Thus, the use of such generators may not be suitable when researchers need to carry out experiments with real production-like traffic.

We believe that intelligent network traffic modeling can be utilized to overcome the privacy issues involved in sharing traffic captures. Instead of direct sharing of the captured traffic, a model for the application behavior can be extracted from the traces through removal of specific network protocol reactions to impairments in networks such as retransmissions, acknowledgments, and other protocol messages. We describe a new framework that extracts traffic models from captured traffic accompanied with methods to generate traffic from the models in a different testbed environment. The model files provide methods and specific statistics to generate the original network traffic on any number of endpoints, on a different network running any choice of transport, network, and data link protocols. Hence, privacy of original capture is preserved and capability to test application performance on desired network topologies is achieved. We present the outcomes from our preliminary evaluation of the framework.

This research is supported in part by the NSF Grant CNS Core 1908974

II. Framework Description

Our framework (Fig. 1) consists of 4 main components.

1) *Dataset Extractor:* The input to the dataset extractor is a `libpcap` traffic trace file. The extractor performs a reassembly of the IP fragments, detects re-transmissions to remove duplicates, re-orders out-of-order packets, determines parent TCP and UDP connection, and prepares the application level protocol data units (PDUs). The extractor provides application payload data with a coarse labeling of packet header information in a csv-formatted dataset as the output.
2) *Modeling System:* The system runs various ML algorithms to generate traffic model files for the extracted datasets. The model files include methods, statistics, and other parameters required for the generation of each application traffic at the desired endpoints on a given test network. Our current implementation applies a combination of clustering algorithms, expert decision trees, stochastic and empirical distributions. A sample JSON-formatted traffic model file is in Fig. 2 (b).
3) *Packet Generation System:* The traffic model files provide the traffic generator with the application traffic patterns and

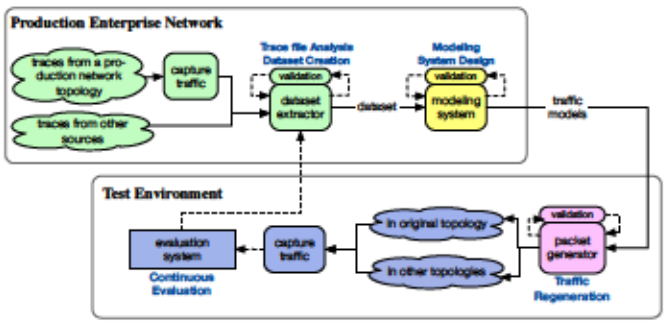
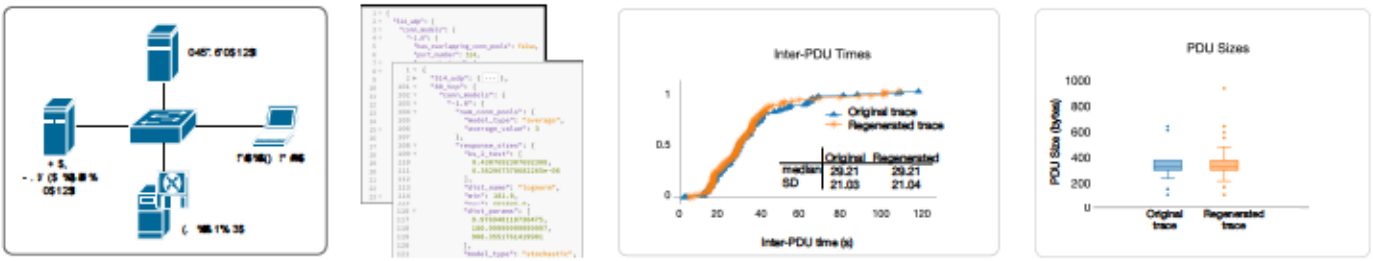


Fig. 1. Framework components along with evaluation and validation tasks.

978-1-7281-6607-0/20/$31.00 ©2020 IEEE



(a) Experiment Topology (b) Sample traffic model file (c) CDF plots for syslog app inter-PDU times (d) Box plots for syslog app PDU sizes

Fig. 2. (a) Experiment topology; (b) Traffic model file samples; (c) and (d) Preliminary evaluation results: There is a close overlap between the generated traffic and the original traffic on both the CDF plots, and the box plots.

generation methods. Our current generator implementation is portable to leverage any host operating system network stack. 4) *Evaluation system:* Regenerated packet traces are compared with the original input traces to provide an assessment of the level of similarity based on various traffic metrics.

III. Preliminary Experiment and Results

We performed preliminary experiments to demonstrate our framework in modeling and re-generating traffic for a typical web application in testbed network. To obtain an initial set of traffic traces, we set up a network as shown in Fig. 2 (a). The network has a `nginx` web server with pages of a web documentation. A client machine on the same network sends automated page requests for the documentation at random time intervals using a `selenium` web automation driver simulating a typical user. Another node serves as a `syslog` server for the log messages pushed by the `nginx` web server. Finally, a monitor node captures the traffic flowing within the network, using the span port of the bridge. Each run is about sixty minutes. We analyzed the traces for the `syslog` service.

Dataset extractor creates a csv file containing details of packet headers and connections. The modeling system creates a traffic model file as shown in Fig. 2 (b). We then create a test replica of the original network where the traffic was captured on our VTS testbed [1], running on a 64-bit 3.40GHz Intel(R) Xeon(TM) machine with 128 GB memory. The test network has five nodes, that is, 2 end hosts emulating the documentation server and the client machine, a host to emulate the `syslog` server, and a host to store the captured traffic. Using our repeatable experiment orchestration framework [2], we generate traffic on each host based on the traffic models created and we capture the regenerated traffic at the node that is connected to the span port of the bridge.

We evaluated the results of the framework by comparing the statistical parameters of the regenerated traffic and the original traffic. The preliminary results are shown in the Fig. 2 (c) and (d). The PDU size and inter-PDU time distributions are compared for the `syslog` service packets. The syslog service exhibits a unidirectional flow of messages from the client, as a second order effect of HTTP requests received by the `nginx` service. In Fig. 2 (c), the inter-PDU time values range from 1.54 to 117.69 seconds in the original trace, which compares to the range of 4.93 to 108.48 seconds seen in the regenerated trace. The PDU size distribution is displayed as box plots in the Fig. 2 (d), indicating interquartile range of 309.25 - 387.00 bytes in the original trace, which compares to the interquartile range of 309.75 to 386.00 bytes in the regenerated trace.

IV. Conclusion

The framework components are validated through statistical evaluations of traffic metrics. A portable model output generation mechanism allows for privacy-preserving methods of realistic traffic generation. The framework design is extensible, making it easy for other users to add their custom modeling and regeneration methods. Source code and samples are available in our version-controlled repositories [7].

References

[1] Nicholas Bastin. *GENI - Virtual Topology Service VTS.* 2017. URL: https://geni-vts.readthedocs.io/en/latest/ (visited on 11/08/2019).

[2] S. Baxley, N. Bastin, and D. Gurkan. *Analysis of In-order Packet Delivery Network Policy Enforcement Function.* Ed. by IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) - Second Place in GENI Repeatable Experimentation Competition. 2018.

[3] Samad S Kolahi et al. "Performance monitoring of various network traffic generators". In: *2011 UkSim 13th international conference on computer modelling and simulation.* IEEE, 2011, pp. 501–506.

[4] Lun Li et al. "Modeling for Traffic Replay in Virtual Network". In: *2018 IEEE 20th International Conference on High Performance Computing and Communications.* IEEE. 2018, pp. 495–502.

[5] Nick McKeown et al. "OpenFlow: enabling innovation in campus networks". In: *ACM SIGCOMM Computer Communication Review* 38.2 (2008), pp. 69–74.

[6] Douglas C Sicker, Paul Ohm, and Dirk Grunwald. "Legal issues surrounding monitoring during network research". In: *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement.* ACM, 2007, pp. 141–148.

[7] UH Networking Lab Staff. *UH Networking Lab - uhexp.* 2020. URL: https://hg.uh-netlab.org/netlab/uhexp/ (visited on 03/24/2020).