Date: March 31, 2016
Working Plan: #6
WP Period: April 04, 2016 - May 16, 2016
Student: Anderson dos Santos Paschoalon
Advisor: Christian Esteve Rothenberg

Working Plan Document

1 Preface

The purpose of this document is present the guidelines of the next 6 weeks of my master degree work. The focus is going to be both the writing of the qualification document and the development of the first prototype of my project, along with the collection of experimental results, in order to improve the methodology. The project is going to be briefly presented in the next sections.

This document is organized in the follow sections:

1. Preface: presentation of the document
2. Introduction: brief presentation of the project motivation and goals
3. Architecture: high-level presentation of the system architecture
4. Goals: brief presentation of the goals of this development cycle
5. Next steps: future goals
6. Working plan board: table of activities

2 Introduction

Network traffic generators have a vital part in the research on network technologies. Also, it is desirable that these traffic generators are able to reproduce realistic traffic scenarios, in order to verify how each new technology being tested behave with different types of loads.

Over the years, many researchers have worked with the purpose of creating more elaborated and accurate models to reproduce the behavior of an actual network traffic. In this way, the most common models used are the stochastic (Poisson and Markov) [1], and the self-similar (factual) [1].

In parallel, many developers were able to create many different traffic generators, capable to use these accurate models, many of them with open APIs[2]. The give the user the possibility of produce customizable traffic of many different protocols of the network stack, and customize the traffic characteristics of packet size distribution and inter-departure time. As popular examples, could be mentioned D-ITG[2] and Ostinato[3], but many others could be cited [4].

Also, when is desired to test specific and more realistic loads, the use of pcap files is possible as well. Pcap files are captures of actual network packets, most of the time made with the use of sniffer tools. Two of the most commons are tcpdump and Wireshark. Then, they are used as input file of a replay tool, for example, tcpreplay[5].

The purpose of this project is fins a midterm between the use of pcap files, and user-defined loads provided by traffic generators. The goal is to create a software capable of "learn" features from real network traffic traces (live captures and pre-recorded pcap files), and with this acquired knowledge, be able to reproduce a network traffic with the same behavior, using an API of a consolidated traffic generator. The learning process should be implemented by direct measurement of features, or by estimation of a good parameter to fit the measured data.

Then, the software must record the obtained features in a machine-readable file, here called "compact trace descriptor". This data must serve as input for an API of a traffic generator. Then, synthetic flows, with the same measured features should be generated. The goal, is create a synthetic traffic, with the same fundamental characteristics of the observed traffic; for example:

- Packet size distribution;
- Inter-departure time distribution;
- Flows (number, characterization, duration);
- Packet rate, bitrate packet loss, jitter, ttl;
- L4 and L3 Protocols (TCP, UDP, ICMP, DCCP, SCTP, IPv4, IPv6).

The project should be used as a contributor of a larger project: The "NFPA: Network Function Performance Analyzer" [6]. NFPA is an open-source benchmark tool of high performance, constructed with the purpose to serve as an analysis of performance of network function, especially NFV systems. It is capable to perform statistical analyzes over network traffic, or reproduce built-in or user-defined network traces in order to simulate a network traffic. It is built over dpdk[7] and paktgen-dpdk[8].

The idea is that after the fist validation of the prototype, it could be expanded to other traffic generators. Here there are some specific challenges about how it could be used in this specific scenario, since the network interfaces used by DPDK are not visible in the operating system(just by DPDK), and at the same time try to avoid overheads and bottlenecks.

3 System Architecture

In the figure 1 is presented the system component diagram proposed, composed by five components: *Siniffer*, *SQLite* database, a *Trace Analyzer*, a *Flow Generator*, and a *Network Traffic Generator* as subsystem.

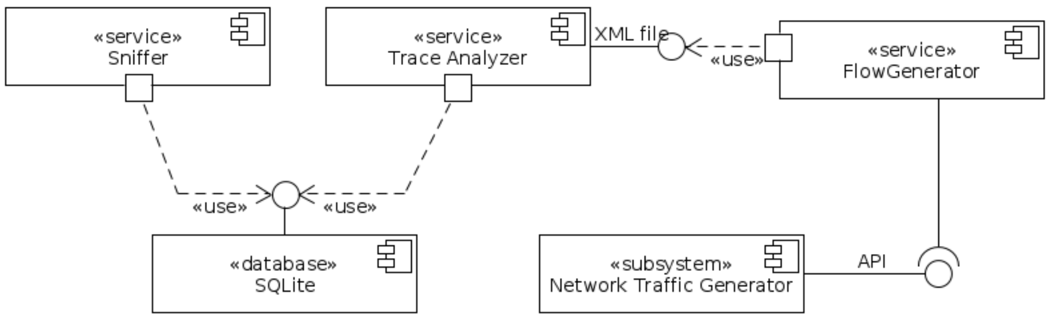

Figure 1: Component diagram of the project.

Each part is described down below.

- ***Sniffer***: this component is responsible for collect data about the network traffic, separate it intoflows, and store all useful data of each one of these flows in a database.
- ***SQLite database***
  - It should be a database responsible for store the collected data form traces, and store it organized by flows.
  - It should be written by the Sniffer, and read by the Trace Analyzer module.
  - According to the SQLite specifications [9], it must be the better option for a database: it is simple and well-suitable for amount of data smaller then terabytes.
- ***Trace Analyzer***:
  - This module aims to create the compact trace descriptor, throw the analysis of the collected data.
  - Must be able to learn features from the analyzed traces, and write them in a machine-readable file (XML/Json).
  - Flow-level features, Packet-level QoS metrics and Leyer-4 features may be directly measured from the collected data.
    - Flow-level properties: duration of flow, start delay, total number of packets, total number of KBytes.
    - Packet-level QoS metrics: bitrate, packet, RTT, Jitter, packet loss.
    - Leyer-4 features: protocols TCP, UDP, ICMP, DCCP and SCTP.
  - Stochastic features, like PSD (packet-size distribution) e IDTD (Inter Departure Time Distribution), are defined by models, in a traffic generator API. So, should be used models of regression and cost minimization, to fit

the observed data into a model, as well as possible, chosing apropriated parameters.

- ***Flow Generator***
  - The Flow Generator should be able to read the data stored in the file created by the Trace Analyzer component, and use it as parameters to the synthetic traffic generation.
  - In the first prototype of the project will be used the D-ITG API as traffic generator subsystem. But, anthers traffic generators should be supported, so is important here the use of a design pattern like factory, to extend this component.
  - To the incorporation of this project as a contributor of the NFPA [6] [10] able to provide a creation of user defined compact trace descriptors via observation, will be necessary the implementation of same methods implemented with the D-ITG framework with de NFPA traffic generator framework. Both D-ITG and NFPA implementation of this component **must** inherit the same interface.
- ***Network Traffic Generator***: A netowork traffic generator software that should provide its interface/API to the Flow Generator component..

It is already possible to define the implementation possibilities for each component, according to the specification. They are in the table 1

Table 1: Component implementation

Sniffer	shell script, tshark(scriptable wireshark) sniffer
Database	SQLite
Trace Analyzer	Python (NumPy and anothers APIs) and Matlab (for prototyping and data visualization)
Flow Generator	C++ (D-ITG API)
Network Traffic Generator	D-ITG, and in the future expand for NFPA

References

[1] W. Wu, N. Huang, and Y. Zhang, "A new hybrid traffic generation model for tactical internet reliability test," in *Reliability and Maintainability Symposium (RAMS), 2015 Annual*, pp. 1–6, Jan 2015.

[2] A. Botta, A. Dainotti, and 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.

[3] "Ostinato packet traffic generator and analyzer, homepage." http://ostinato.org/. [Online; accessed 31-03-2016].

[4] 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*, pp. 1–6, March 2014.

[5] "Tcpreplay - pcap editing and replaying utilities." http://tcpreplay.appneta.com/. [Online; accessed 31-03-2016].

[6] B. S. Levente Csikor, Mark Szalay and L. Toka, "Nfpa: Network function performance analyzer," *IEEE Conference on Network Function Virtualization and Software Defined Networks*, 2015.

[7] "Dpdk - data plane development kit." http://dpdk.org/. [Online; accessed 31-03-2016].

[8] "The pktgen application." http://pktgen.readthedocs.org/en/latest/index.html. [Online; accessed 31-03-2016].

[9] "Appropriate uses for sqlite." http://www.sqlite.org/whentouse.html. [Online; accessed 01-22-2016].

[10] "Nfpa: Network function performance analyzer." http://ios.tmit.bme.hu/nfpa/, 2015. [Online; accessed 01-21-2016].

Week Work Plan

	Tasks	Sts	Week					
			04/04	04/11	04/18	04/25	05/02	05/09
Practical tasks								
t 1	Reading: bibliography review	doing	■	■	■			
t 2	Writing: qualification	doing		■	■	■	■	■
t 3	Implementation: visual modeling in UML (1st version)	To do	■					
t 4	Implementation: the sniffer component (scripts)	doing	■					
t 5	Implementation: first version of the trace analyzer	To do		■	■	■		
t 6	Implementation: first version of the Flow generator	To do			■	■	■	■
t 7	Experimentation: Automatize data visualization	doing	■	■				
t 8	Experimentation: collect data of datatraces and partial results	To do			■	■	■	■
t 9	Presentation of partial results	To do						■
t 10	Define the next WP	To do						■