UNICAMP - State University of Campinas
School of Electrical and Computer Engineering (FEEC)
Department of Computer Engineering and Industrial Automation

Date: Jan 18, 2016
Working Plan: #5
Software Requirement version 0.1.0
WP Period: Jan 25, 2016 - Feb 07, 2016
Student: Anderson dos Santos Paschoalon
Advisor: Christian Esteve Rothenberg

Software Requirements and Working Plan Document

Table 1: Document version

0.1.0	High level specification of the requirements and experimental test-bed, component diagram implementation.

1 Preface

The purpose of this document is to work both as a simplified version of a requirement document, and as a working plan for the research activities. This document was organized based on the structure presented by Sommerville [1].

This approach was chosen because both activities will are directly related. A clear definition of the requirements is need for a good research, and vice-versa. This is specially true at the first stages of the project.

The text-book [1] suggests these sections for a software documentation: Preface, Introduction, Glossary, User Requirements Definition, System Architecture, System Requirements Specification, System Templates, System Evolution, Appendices. Here, is going to be used the follow structure:

1. Preface
2. Introduction
3. User Requirements Definition
4. System Architecture
5. System Requirements Specification
6. Experimental Validation
7. System Templates
8. System Evolution
9. Working Plan

10. References

In this first version, most of the requirements are defined at a high level. The definition of the requirements at a lower level for the next version, is going to a an activity of this cycle of research too. It is going to be based on solutions already proposed by the academia and open-source community. Therefore, the focus will be in on a clear and low level definition of requirements, and in the search for solutions, and continue my studies in machine learning techniques in parallel.

This document is going to be written in an evolutionary approach: on each new version, the requirements and architecture will be updated, and for it will be defined the following Working Plan.

2 Introduction

The project aims to create a software capable of "learn" patterns of real network traffic traces, and with this acquired knowledge, be able to reproduce a network traffic with the same behavior. The learning process should be implemented by direct measurement of features, or by an automatic learning process.

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 features should be generated. A synthetic traffic, with the same fundamental characteristics of the observed traffic should be created.

The project should be used as a contributor of a larger project: The "NFPA: Network Function Performance Analyzer" [2]. NFPA is an open-source benchmark tool, 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. The project here explained could contribute in this last spot: be used as an automatic generator of compact network traces defined by the user via learning process.

As guideline, the main requirements are:

- Create "compact trace descriptors". This should be a file written in machine readable format (XML, Json) capable of store useful data about flows of a real network trace. The file must have a field for each observed flow. And each one of these fields must have enough information to characterize it. This information must be compatible with settable features of network generators APIs. This data must serve as input for this/these APIs, and must generate a network traffic with the same measurable characteristics of the observed traffic, without the storage of (most of the time), huge pcap files.

- To classify the flows, will be need measure information about them, separately. Will be need a component capable of divide a trace int n flows.

- All the data will be fist stored in a database, and then analyzed for the extraction data.
- Characteristics easily measurable form the flows, as throughput e start/end time, could be directly measured and recorded. Form more complex characteristics as packet size and inter-arrival distribution, this could be made with the use of machine learning classifiers, capable of classifying flows in some labels, according to some observed patterns.
- The project should be tested in practice in a NFV environment. The purpose is use it together with NFPA [2] [3]. The advantage is a complete set of tools for benchmark, that could be used to analyze how the network is responding to the created traffic, and if it actually has the same characteristics of the observed trace.
- Using the NFPA, should be defined a test-bed and some experiments to observe the differences in the network when are used traces pcap traces and synthetic generated flow with a poor refinement of features (for example, a limited number of flows, and default for packet size distribution, and a constant bit rate, loss or time-extension of the flows). The firsts tests comparing results of pcap traces and a manually defined traces of a network generator, could be already made. Then, the same tests should be applyed over the traffic generated by the project.

3 User Requirements Definition

- The system user must be able to require that a network trace should be analyzed, and through the analysis of the collected data, a compact trace descriptor should be generated. The analysis should be made in real-time (observing a trace directly from the network interface) or through a pcap file already saved.
- The user must be able to require that a synthetic network traffic should be produced through a compact trace descriptor.

4 System Architecture

In the figure 1 is presented the system component diagram proposed, composed by five components: *FlowSplitter*, *FlowDb*, *TraceAnalyzer*, *FlowGenerator*, *NetworkTrafficGenerator*.

Each part is described below. New components, especially new subsystem should be added to the project as low level requirements are defined, aiming reuse of code.

- ***FlowSplitter***: this component is responsible to split the collected network traffic into flows, and store all useful data of each one of these flows in a database. (may require an additional subsystem component)

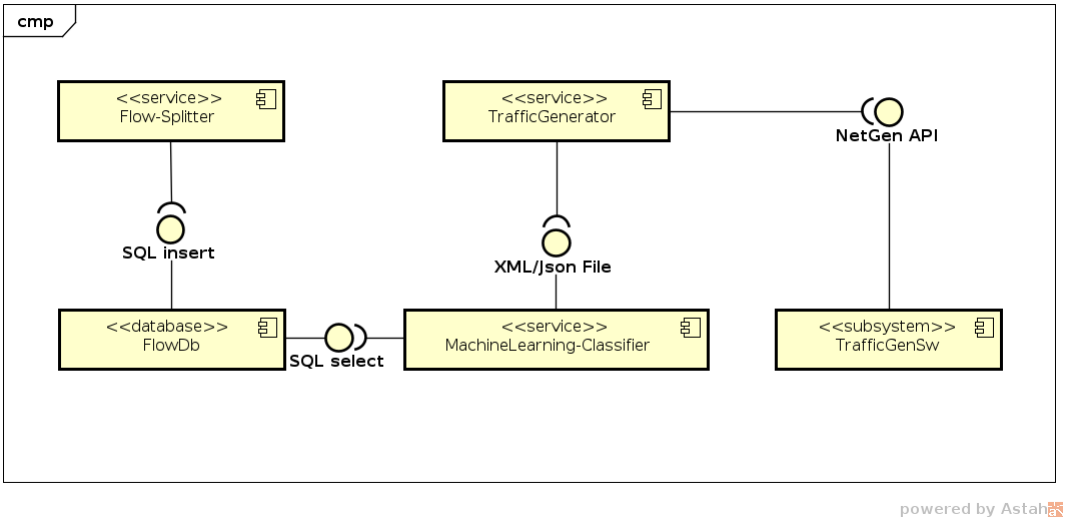

Figure 1: Component diagram of the project.

- ***FlowDb***
  - 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 FlowSplitter, and read by the TraceAnalyzer module.
  - The database must store a table for each trace (experiment) captured, and for each table, a subtable for each trace flow. A graphical representation of the database is presented in the figure 2.
  - According to the SQLite specifications [4], it must be the better option for a database: it is simple and well-suitable for amount of data smaller then terabytes.

- ***TraceAnalyzer***:
  - 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, 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.

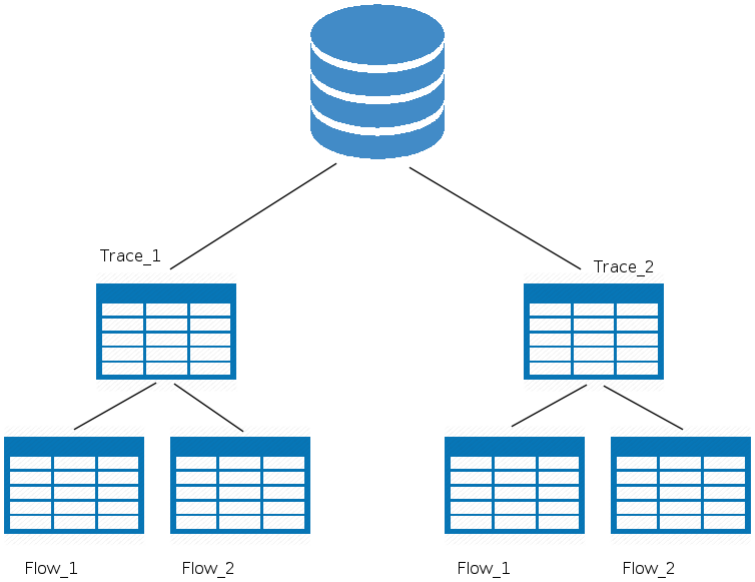

Figure 2: Database representation.

  - Stochastic features, like PSD (packet-size distribution) e IDTD (Inter Departure Time Distribution), should be determined by some technique of machine learning, still not defined. Using it, will be possible to classify on what sort of PSD or IDTD the flow fits.

- ***FlowGenerator***

  - The FlowGenerator should be able to read the data stored in the file created by the TraceAnalyzer component, and use it as parameters to the synthetic traffic generation.
  - In the first release 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 polimorfism.
  - To the incorporation of this project as a contributor of the NFPA [2] [3] 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.

- ***NetworkTrafficGenerator***: A netowork traffic generator subsystem that should provide its API to the FlowGenerator component. In the first release should be used the D-ITG, because it has a large amount of configurable settings, and has a very simple API. In the next version it should be used the NFPA API for traffic generation as well.

It is already possible to define the implementation possibilities for each component, according to the specification. They are in the table 2

Table 2: Component implementation

FlowSplitter	not defined
FlowDb	SQLite
TraceAnalyzer	not defined (Python or C++ API for machine learning)
FlowGenerator	C++ (D-ITG API)
NetworkTrafficGenerator	D-ITG and NFPA
CLI	Klish (http://code.google.com/p/klish/) (C++ and XML)

5 System Requirements Specification

to do! *(low-level description of the requirements and interfaces to be implemented)*

6 Experimental Validation Methodology

to do! *(experimental setup to be used with NFPA to validade the project to evaluation of network functions metrics of loads produced by real network traffic traces, manually and default configured synthetic traces, and the synthetic traces created by a compact trace descriptor)*

7 System Templates

to do! *(software diagrams of the project: use case diagram, class diagrams, sequence diagram)*

8 System Evolution

Table 3: Software version

version	description	date
0.1.0	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	mm/dd/yy

9 Working Plan

References

[1] I. Sommerville, *Engenharia de Software*. Av. Ermano Marchetti, 1435, São Paulo - SP, CEP:05038-001: Pearson Addison-Wesley, 8 ed., 2007.

[2] 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.

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

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

Tasks

t1 Study of NFPA project:
(1) Construction of a experimental setup in order to evaluate the performance of network functions with different sets of trace loads, configured in different ways.
(2) Define a topology with a set of nodes, and on such points the traces will be collected and produced.

t2 [t1] Report writing: Write a report with the obtained results, feasibility and perspectives about the experimentation of the project.

t3 Bibliography review:
(1) Trace split process (tools, libraries and techniques);
(2) Network data collection (tools and libraries);
(3) art state in traffic classification (techniques, tools);
(4) Stochastic models for packet size distribution and Inter Departure Time.

t4 [t3] Write a brief bibliography review with tables about each of the contents of task t3.

t5 [t4] FlowSplitter implementation prototype (v0) (for pcap files)

t6 Requirements extraction for the *FlowGenerator* component, and interface definition (Compact trace descriptor template), prototype version (v0).

t7 [t6] *FlowGenerator* implementation (v0)

t8 [t1] Database Requeriment extraction and definition (v0).

t9 [t8] Database implementation (v0)

t10 Coursera: Practical Machine Learning

t11 Coursera: Machine Learning

Week Work Plan

	Tasks	Sts	Week 01/25	02/01	02/08	02/15	02/22	02/29	
Practical tasks									
t 1	Study of NFPA project	To do	■	■	■	■			
t 2	[t1] Report writing	To do					■	■	
t 3	Bibliography review	To do	■	■	■				
t 4	[t3] Write review	To do				■	■	■	
t 5	[t3]FlowSplitter prototype	To do					■	■	>>
t 6	Requirements extraction *FlowGenerator*	To do	■	■					
t 7	[t6] *FlowGenerator* implementation (v0)	To do		■	■	■	■	■	
t 8	[t1] Database Requirement(v0).	To do			■	■	■		
t 9	[t8] Database implementation (v0)	To do					■	■	>>
t 10	Coursera ML	To do	■	■	■	■	■	■	
t 11	Coursera Practical ML	To do	■	■	■	■	■	■	