Anderson dos Santos Paschoalon
Prof. Christian Esteve Rothenberg
Intrig:FEEC

Research and Development Pre-proposal Report #1

In this report are going to be defined the first guidelines for my research proposal. The Idea is produce multiples of these report over time, in order to "refine" and clarify the ideas. With the ideas cleared and organized, it should be easy to decide whether is good to proceed or not. This document should evolve to the qualification document.

1 Research guidelines

- In the field of network testing, there is a need for high precision and high rate traffic equipments, to ensure correctness of network equipment
- There are many proprietary alternatives, provided by companies, such as Ixia, Spirent, Flux and Emulex/Endace; but few open source options. The NetFPGA platform aims to fill this gap
- Many of the proposals of traffic generators over the NetFPGA make use of pcap files, that require a lot of hard disk space, an option proposes the creation of a compact traffic descriptor.
- A recent implementation over the NetFPGA 10G, the OSNT provide an alternative of creating statistically the traffic. In this field, another innovation proposal could be the use of machine learning to do this work. Make the machine learn how to create different kinds of synthetic traffic, like multimedia streamings, http requests, FTP requests, torrent download, traffic of a laboratory, etc
- Are going to be used not just information like packet size distribution, but much information as possible like IP and make addresses, protocols, and much more. To store this during the running time will be needed a sparse multidimensional matrix.
- With the generated traffic it is possible do study a system like an NFV testbed. And then compare the results with other open source tools, like OSNT, Caliper and SPG (Stanford Packet Generator).
- For this work, will be used as much pre-implemented frameworks as possible. The focus of this work will stand, in this order:
    1. Create a complete and compact, multidimensional description of different network traffics
    2. Use machine learning to create an intelligent generator of traffic
    3. Test this feature over the an commodity NIC. Then, integrate them with NetFPGA tools.

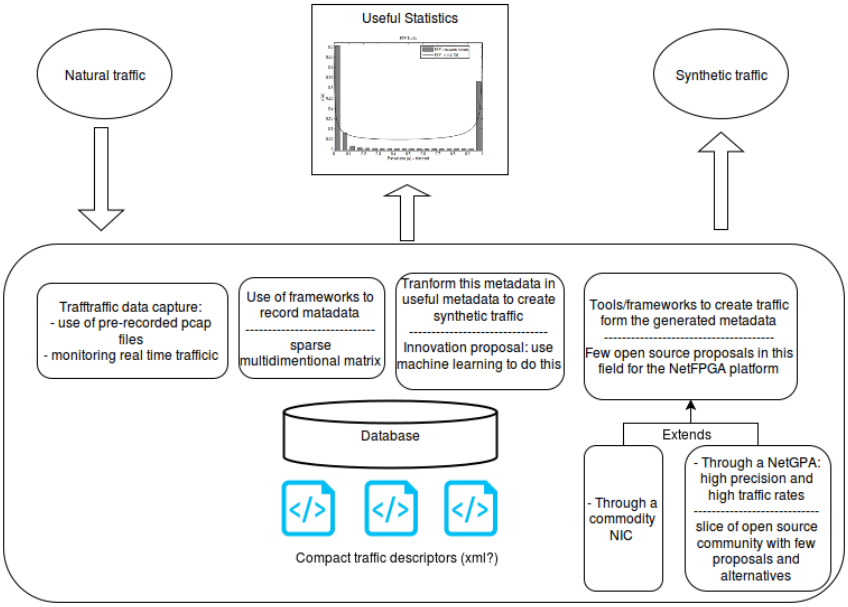

Figure 1: Simple proposal of architecture of the whole work to be done related to the traffic generator and analyzer.

2 Proposed Architecture

As mentioned before, the idea of implementation is not build a whole system from the scratch, but use frameworks and open source tools available, and just "fill the gaps".
In the figure 1 is presented a simple architecture of the proposal of research and development made before. Lets point some important checkpoints about the architecture:

- There are many available tools and frameworks for packet capture, analyze and monitoring
- There is a huge amount of work done of packet generators to run over commodity hosts. Also, there are some works made over the NetFPGA platform as well.
- The majority of the work will be focused in these key points:
  1. Create an application to extract all usefull data, and store it in a structured manner in a sparse multidimensional matrices, and store it for future use
  2. Implement a method to create synthetic traffic. As a good choice for innovation we have machine learning. Recoding the machine state, it will be possible to reproduce with the same quality a synthetic traffic.
  3. As a feature, this software should be portable and integrated with commodity and NetFPGA packet generator tools.

3 Tools and Frameworks for Evaluation

As languages for implementation, will be first considered C++/C and Python. Python in general represents a good compromise solution between efficiency and simplicity. If a grater performance should be achieved, C/C++ is a good option.
As starting points, first, must be evaluated **packet analyzer and sniffer** libraries. Here, some options to be evaluated:

- **Pcaptools** : https://github.com/caesar0301/awesome-pcaptools#linuxcmds
- **C++::libpcap** : https://github.com/the-tcpdump-group/libpcap, http://sourceforge.net/projects/libpc
- **C++::libtins** : http://average-coder.blogspot.com.br/2013/10/creating-simple-packet-sniffer-in-c.html, http://libtins.github.io/
- **Python::socket** : http://www.binarytides.com/python-packet-sniffer-code-linux/
- **Python::socket** : http://www.binarytides.com/python-packet-sniffer-code-linux/
- **Dpdk** : http://dpdk.org/doc

Some options for packet generators:

- **C++::libcrafter** : https://code.google.com/p/libcrafter/
- **Script::MoonGen** : https://github.com/emmericp/MoonGen
- **Python::sniffer** : https://pypi.python.org/pypi/sniffer
- **Python::pycap** : http://pycap.sourceforge.net/
- **Python::pyshark** : https://pypi.python.org/pypi/pyshark/0.1
- **Python::dpkt** : https://code.google.com/p/dpkt/

To implement Sparse matrices:

- **Python::sparray** : http://www.janeriksolem.net/2010/02/sparray-sparse-n-dimensional-arrays-in.html
- **C::Meschach,Cooperware,Blitz** : http://www.ibm.com/developerworks/library/l-matrix/index.html
- **C++::vector_sparse.htm** : http://www.boost.org/doc/libs/1_41_0/libs/numeric/ublas/doc/vector_spa
- **C++::SparseMatrix** : http://www.sanfoundry.com/cpp-program-implement-sparse-matrix/
- **C++::SparseLib++** : http://math.nist.gov/sparselib++/

The next step will be evaluated some of these tools, and choose the more appropriated ones.