# NetFPGA packet generation with OMF automatization

**Steve Ataky Tsham Mpinda[1], Gabriel Diego de Aguiar Aranha[1], Jean Menossi[1], Patrick Andjasubu Bungama[2]**

[1]Departamento de Computação – Universidade Federal de São Carlos (UFSCar)
Caixa Postal: 8290-000 - 16 3351-8233 - São Carlos – SP – Brazil

[2]Departamento de Computação – Universidade Federal do Paraná (UFPR)
Caixa Postal: 80060-000 - 41 3360-5000 - Curitiba – PR – Brazil

{stevetmat, patrick.bungama, jean.menossi}@gmail.com,
talk@gabrielaranha.com

***Abstract.*** *Nowadays, the Internet and networks in general are increasingly present in our day-to-day. Since facing certain limitations regarding its growth, the network researchers increasingly offer new solutions as an attempt to predict and solve problems that may arise therefrom. The development of these solutions involves certain tests and exhaustive experiments before they become available to users. Network testbeds, currently, are the most meaningful manner to reach this goal. Aggregated with frameworks such as OMF, are making tests increasingly mature. This article is a brief presentation of the OMF, explaining what it is, how it works and how it can be applied.*

***Keywords:*** *OMF, NetFPGA, Automatization.*

## 1. Introduction

Currently, the Internet and networks in general are increasingly present in our day-to-day. Once facing some limitations due to its growth, it boosts the industry and researchers to develop new network technologies with massive previous tests and evaluations before heading to production. Thus, it is essential to use the most appropriate methodology which has a greater capacity to generate results closer to the reality. This leads us to the experimental platforms with a focus on network testbeds, which are key factors in the development and maturation of computer networks.

Although testbeds are very costly products, in term of implementation and price, and commonly suffer deactivation after a period of time as they have been designed exclusively for a specific project and this waste of resources is being pointed out by the researchers, other testbeds initiatives, however, more "generic" are rising increasingly. Taking into account the factor of generic testbeds, cOntrol and Management Framework (OMF), framework designed to be exclusively used in the so-called project ORBIT is evolving to support and fervently support new features and technologies. This article exposes what is the OMF, its structure, as well as its functioning. Accordingly, it is expected that after reading the user be able to develop and run a little experiment over a testbed based OMF framework.

## 2. Testbed Management Platforms

Over time, some different approaches regarding network testing have emerged, such as simulation, emulation, and deployment of testbeds. By using a simulation test, experiments are provided with a fully controllable environment with all the variables of a network at its disposal. The network is modeled by the tester and then simulated with the software. However, there is a variety of network scenarios, and the simulation alone cannot provide reliable results. Nevertheless, the simulation demonstrates great importance in preliminary tests, especially when they are compared with the theoretical values.

Due to the lack of reliability mentioned above, the network's test community comes to supporting the deployment of the test's installation. Testbeds emerged as a solution to this problem, as well as related software, suggesting new approaches and different goals.

Thereof, the OMF project was developed aiming at solving such problems. The OMF is a framework that provides easy installation of a software control capability to any owner of a testbed or an experimenter. Hereafter will be presented a more detailed analysis of this tool with testbed deployment.

### 2.1. COntrol and Management Framework (OMF)

The OMF framework is a software package that provides tools for control, management and measurement of testbed resources. While the testbed's owner gets access to effective management of all resources, users can easily instrumentalize their experiments, run and collect all their results in a centralized way [Rakotoarivelo, Jourjon and Seskar, 2010].

The OMF is designed to achieve reproducibility and greater repeatability in experimentation. Its main design principle focuses on the experiment description. This description contains all the configuration information and resources; then this description will delegate it to OMF, where resources are deployed and configured. The experiment is run collecting results from the user perspective.

The basic workflow presented is based on three steps (Figure 1):

1. The user uses a script to be supplied to OMF, called experiment description (ED), containing applications, operating system images, topologies, and a particular behavior of node or network settings;
2. The OMF will implant and set up itself according to this ED. When all resources are configured, the experiment is executed and the measurement datas are collected;
3. Finally, the measurement datas are stored into database on a standard format, so that they can be used for further analysis.

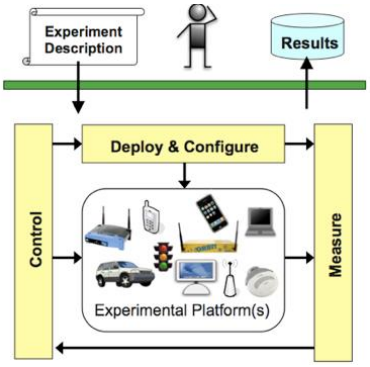


**Figure 1. OMF Functional Principal**
**[http://omf.mytestbed.net/projects/omf/wiki/An_Introduction_to_OMF]**

## 2.2. OMF Architecture

The OMF architecture, according to the used version (in this case, version 5.4) is structured as shown on Figure 2. Three main entities retain control of the testbed. The Experiment Controller (EC) is the entity to which users have access. As stated above, an experiment description is passed to this entity in order to run on a Testbed. At this point, the EC will ask the Aggregate Manager (AM) for the resources initialization, loading images from disk (if necessary) and communicating with the Resource Controller (RC) to issue commands of the network's configuration or applications' initialization. Once all the resources are active and installed, the experiment starts.

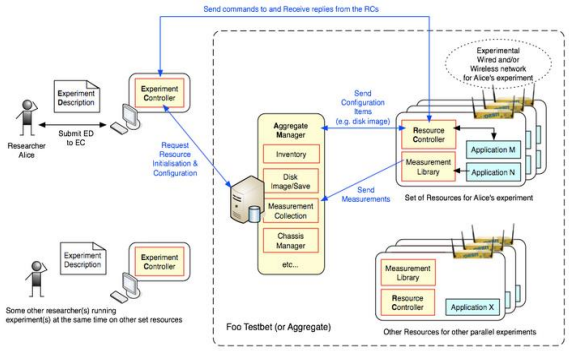


**Figure 2. OMF Architecture Diagram**
**[http://omf.mytestbed.net/projects/omf/wiki/An_Introduction_to_OMF]**

As an experiment is running, the EC may require to nodes to perform their applications or modify their parameters experiment / application. The nodes respond through the RC module. When applications are still running, results will be collected and sent to the AM and saved to a database, which later will be accessed by AM services periodically. The correct OMF implementation requires a control and experiment network. Traffic control should be separated, otherwise the results can be inconsistent due to the background traffic created by the control entities.

## 2.3. Experiment Controller (EC)

Also known under Node Handler (NH), the role of EC is to allow users running their experiments. The user gets in contact with the EC to execute commands such as loading an image to either a node or nodes, or performing an experiment, for example.

The EC collaborates with AM only when the resource's initialization occurs as in the case of loading the disk images. As the resources are configured, the EC will communicate with nodes sending commands according to its experiment description.

The EC is aggregated as a program to be installed on terminal users. It holds responsibility for implementing tasks such as configuration, deployment and experimentation. These tasks often are slow and require manual entry whenever tests are done, but using the OMF all these items are controlled through the language created for this purpose. It is the interface of an OMF-enabled testbed, called OMF Experiment Definition Language (OEDL). This language is part of the EC, the main module of the Experiment Controller.

The OEDL is a Domain Specific Language (DSL) that allows a user instrumentalizing his experiment, making all configuration and usual tasks implementation more simple and as automated as possible. Through this language, the user is able to describe his experiment at a higher abstraction, as the logic centered on the script. Furthermore, the syntax is similar to OEDL [Gunther 2010]; nonetheless, being a DSL, it was designed for people with general programming skills to easily create their scripts.

## 2.4. Resource Controller (RC)

This module is the controlling entity of all available nodes in the testbed. Its main objective is to answer commands issued by the EC. For instance, when a defined group into OEDL script defines a resources path, the EC sends the data path and values of resources to the correspondent RC, which will manage and translate to the appropriate command, which in turn will configure the network interface.

All communication uses the Extensible Messaging and Presence Protocol (XMPP), while the communication model between OMF entities is defined by the OMF Protocol.

### 2.5. OMF Protocol

In early versions of OMF, communication between different entities were performed using a simple text protocol over a TCP / multicast server. In addition of implementation constraints, in a future environment in which different testbeds are interconnected (federation), this approach faced a number of problems.

Entire communications infrastructure experiment was reformulated, thus introducing the Extensible Messaging and Presence Protocol [Kaya *et al*., 2014 and Perrotta, 2010.). It consists of an Internet Engineering Task Force (IETF) standard XML protocol that is used primarily in applications presence, messaging and realtime.

## 2.6. Publish-Subscribe Schema

All communication between resources and experimenters is done through XMPP using the publication and signature extension [Miller 2012]. OMF represents a testbed, which is formed by the "System" resources that will be distributed in "slices". Each feature listens to actions within the system group. When a user wants to perform an experiment, it is made a "slice" request with "Resources" to use. Such as described earlier, when an OEDL script is loaded, groups will be created to make use of testbed resources. Each group also has a corresponding node. The RC will sign its OEDL group and assigned

resources, where it will receive commands configuration. The EC communicates to all nodes that creates for recognizing the groups.

Currently, the EC is responsible for creating the slice and its resources, although this is being expected to be created by third-party frameworks with intent to use an OMF testbed.

## 2.7. Measure Collection

Collection of measurement data is a major challenge to work with testbed experiments on a large scale. Many times, datas may be in log files, whose format may vary. All log files must be collected from all the resources and passed through some sort of serialization in order to extract the most relevant data. This slow collection process adds overhead across the experimentation workflow, and thereby decreases the whole purpose of the experiment, focusing less on the application or routing protocol when testing.

The OMF Workgroup comes with a collection measurement structure to solve these problems, i.e., the Organization Model Language (OML) [Singh 2005]. This structure provides means for an experimenter defines the measuring points, the processing of results, empowering it to organize its data in a single database, avoiding fragmentation to have such a numerous files log across all resources. The efforts of the OML provide a "generic" simple tool that is capable of measuring all, instead of being restricted to metric network.

## 2.8. Measurement Library Client

Doing experiments properly involves the integration of applications with OML. Its main advantages are centralization of previously explained results and filtering mechanism of energy that comes with the OML. This can be done by either changing the source code of application or defining a wrapper on top of the application. OML libraries were made with a simple integration on mind. If one's application is integrated with OML, the RC will take care of the rest and the user gains access to filtering mechanisms that allows to users, with metrics, the control of datas. A filter consists of injecting data into the database that will pass through a series of arithmetic calculations (average, sum, etc.). For example, instead of injecting the total sample captured in average, a smaller number of samples may be injected into the database. Users decide when this process happens, specifying a sampling rate / interval and filters to apply.

The OMF and OML are completely independent. Although OMF components provide an integration with OML when both are present on the same machine. In such cases, the results from the database are created automatically in the file system without the need for any input. Moreover, AM provides web services related to experience results.

## 3. NetFPGA Concept

While systems may be implemented with software capable of sending and receiving some of the incoming and outgoing packets under Gigabit per second, using the software itself is not recommended for switching, routing and processing all traffic that exists in high-speed networks [Lockwood *et al.,* 2007].

The Field Programmable Gate Array Net(FPGA) is a physical platform that allows fast development and prototyping of network applications. Its open distribution

consists of gateware, hardware and software, in addition of source code and scripts (which allow application development). Its basic distribution consists of reference router, a Network Interface Card (NIC), and a Linux router accelerated by hardware. Moreover, there are other projects being developed by users contributors such as NetFlow Probe, the OpenFlow switch and PacketGenerator [Covington *et al.,* 2009]

Inside NetFPGA there is a common Xilinx FPGA device. Around this, there are four memory devices - two static RAMs (SRAMs) and Two Double Data Rate second generation (DDR2) SDRAM. One of its components are the main physical network inputs (four ports in total), which transmit and receive packets through Ethernet cables. Due to its structure, it is possible to connect more than one NetFPGA in series [Lockwood *et al.,* 2007].

## 4. NetFPGA experiment scenario

To test the NetFPGA automatization through OMF and measure the network traffic, a script must be created, whose purpose is to generate traffic for nf2c0 interfaces of a virtual machine and capture the data sent by the interfaces through nf2c1 interface to another virtual machine; both remain in the same physical machine. The script is configured to send 1.000 UDP packets from one interface to another, checking whether all packets were sent and received correctly using a rate of 3 kbps. Through this scenario, one may use a project called Packet Generator that will inject packets on a network with the capability to observe packets that passes by it, all throughout the study of parameters of a script created for this purpose.

### 4.1. Script Parameters

Briefly, Packet Generator project, through a pcap file chosen or designed by the user, dumps into the network a sequence of Ethernet frames. The delay between frames and data rate are optional and specified by the user on the script, also being possible to capture the data sent to obtain statistics on these packages; thereby one can analyze the performance and verify whether datas leaving a network are correct. To generate traffic, the packet_generator.pl file, provided by the above Packet Generator project in an OMF environment shall be used and being automatized following the instructions of script below:

1. -q<queue number> < pcap file>: pcap File loaded and sent to the queue.
2. -r< queue number > <rate>: Rate in each queue (Kbps).
3. –i<queue number> <Iteration number>: Number of iteration per queue.
4. –d<queue number> <delay between packets>: Delay between packets (ns).
5. –c<queue number> < delay between packets >: Capture delay.
6. –pad: Allows populating data to reach 64 bytes.
7. –ns: Presents time with nanosecond precision.
8. –wait: Expects a signal from another process and allows synchronization of multiple generators.

The script will generate traffic by the nf2c0 interfaces and capture data sent by that interface via the nf2c1 interfaces. For sending package interfaces between the NetFPGA the following command is used:

packet_generator.pl –q0 http.pcap –c1 captured.pcap – i0 1000 –r0 3

Thus, we can send 1,000 packets at a rate of 3 Kbps from one interface to the other and see if they are received correctly.

1. defProperty(“resource”, “netfpga”, “ID”) / resource definition

2. defApplication(‘packet_generator’) do |p| / define a PacketGenerator as an application.

3. p.path= “/root/netfpga/projects/packet_generator/sw/packet_generator.pl”/ file’s path definition

4. p.name=”PacketGenerator” / name definition

5. p.description=“injects packtes into network” / description definition.

After the general definitions, one has to define the parameters that will be used:

1. p.defProperty('queue0','pcap file is sent from this queue','-q0',{:order => 1, :type=>:string})

2. p.defProperty('capture1','capture file of queue0','-c1',{:order => 2, :type=>:string})

3. p.defProperty('interaction0','interaction of queue0','-i0',{:order => 3, :type=>:integer})

4. p.defProperty('rateq0','rate of queue0','-r0',{:order => 4, :type=>:integer})

5. p.defProperty(‘ns’,‘nanoseconds', '--ns', {:order => 5, :type => :integer})

6. defGroup('Actor', property.resource) do |g|

Hither are indicated which are the values to be inserted into the application:

```
g.addApplication("packet_generator") do |p|
p.setProperty('queue0', 'http.pcap') # Show "-q0 http.pcap" should put a pcap file with 10 iterations within it.
p.setProperty('capture1', 'captured.pcap') # Show "-c1 captured.pcap"
p.setProperty('interaction0', 1000) # Show "-i0 1000"
p.setProperty('rateq0', 3) # Show"-r0 3"
p.setProperty(‘ns', true) # Show"--ns"
    end
end
```

Finally, it is set when the application will run:

```
onEvent(:ALL_UP) do |event|
 after 120 do # after 120 seconds the experiment is terminated.
 Experiment.done
  end
end
```

To run the script should be use the command: omf_rc exec <file.rb>

## 5. Conclusion

This article presented an overview of the OMF and how it can automatize the generation packet in NetFPGA by management, control and achievement of testbeds.

During the tests, great difficulties were faced with respect to several used components versioning, even though most components have suggestions as to versions, but still have problems. Due to prioritization of the latest versions, one noticed that opting for latest one, it was necessary to install components that were not described in the OMF's installation tutorial.

Another significant difficulty encountered during testing is on the interpretation of adopted triviality in the description of the components installation that supplement the project, for example, the Measurement Library and the Application Resource, resulting in several gaps that delay the project design. Concluding observations, one observed that the community does not discuss points of installation and usability on forums, giving more emphasis on bugs and enhancements, arousing, thereby, the desire to create online initiatives and contribute to the community by focusing on the installation of required components.

## References

Rakotoarivelo, M. Ott, Jourjon, G., Seskar, I. (2010) "OMF: a control and management framework for networking testbeds," SIGOPS Oper. Syst. Rev., 43, pages 54-59.

Gunter, S. (2010) "Multi –DSL Applications with Ruby", *IEEE software,* 27, pages 25-30.

Kaya, F., Eken, S., Ilhan, Z. Kavak, a., Sayar, A. Kocasarac, U., Sahin, S. (2014) "A comparative study of signaling protocols for data management and synchrononization in faith project with school level cloud proxy server deployment" *IEEE 3nd Symposium on Network Cloud Computing and Application,* pages 133-136.

Perrotta, Paolo. (2010) "Metaprogramming Ruby - Program Like the Ruby Pros". *The Pragmatic Bookshelf Programmers.* Edited by Jill Steinberg.Version 2010-1-29, Texas.

Miller, W. J. (2012) "eXtensible Markup and Presence Protocol Interface (XMPP) Standard for sensor, Actuators, and Network Devices. *IEEE Internacional Conference on Smart* Grind Engineering (SGE'12).

Singh, M. Ott, M. Seskar, I and Kamat, P. (2005) "ORBIT Measurements Framework and Library (OML): Motivations, Design, Implementation, and Features, "First Internacional Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, pages 146-152.

Gibb, G, Lockwood, J. W, Naous, J, Hartke, P. and McKeown, N. (2007) "An Open Platform for Gigabit-rate Network Switching and Routing," in Proc. Int'l Conf. Microelectronic System Education (MSE'2007), *IEEE Transactions on Educations.* pages 160-161.

Covington, G. A, Gibb, G, Lockwood, J. W, McKeown, N. (2009) "A Packet Generator on the NetFPGA Platform". *IEEE Symposium on Field-Programable Custom Computing Machines*, pages 5-7.