OMF: A Control and Management Framework for Networking Testbeds*

Thierry Rakotoarivelo[†]
Guillaume Jourjon[†]
Maximilian Ott[†,§]
Ivan Seskar[§]

[†]National ICT Australia (NICTA)
Alexandria, NSW 1435, Australia
first.last@nicta.com.au

[§]WINLAB, Rutgers University
New Brunswick, NJ, USA
seskar@winlab.rutgers.edu

ABSTRACT

Networking testbeds are playing an increasingly important role in the development of new communication technologies. Testbeds are traditionally built for a particular project or to study a specific technology. An alternative approach is to federate existing testbeds to a) cater for experimenter needs which cannot be fulfilled by a single testbed, and b) provide a wider variety of environmental settings at different scales. These heterogenous settings allow the study of new approaches in environments similar to what one finds in the real world.

This paper presents OMF, a control, measurement, and management framework for testbeds. It describes through some examples the versatility of OMF's current architecture and gives directions for federation of testbeds through OMF. In addition, this paper introduces a comprehensive experiment description language that allows an experimenter to describe resource requirements and their configurations, as well as experiment orchestration. Researchers would thus be able to reproduce their experiment on the same testbed or in a different environment with little changes. Along with the efficient support for large scale experiments, the use of testbeds and support for repeatable experiments will allow the networking field to build a culture of cross verification and therefore strengthen its scientific approach.

1. INTRODUCTION

Experimental platforms (or testbeds) are instrumental for the development and evaluation of new network technologies. Evaluations based on simulation provide inexpensive, yet valuable results on the performance of a new approach, or technology. However, simulators, such as NS3 [1] and OMNet++ [2], inherently make simplifying model assumptions. In contrast, convincing the networking industry and community to widely adopt and deploy new algorithms or mechanisms often requires comprehensive (and expensive) testing and analysis in real-world settings with real users. Testbeds are often considered an effective alternative, where new technologies are evaluated in controlled, but real-life like environments and scales. Many research teams currently deploy and use such testbeds in an ad-hoc and independent manner. In many cases, these testbeds are solely built and used for a specific research project, and are seldom maintained or re-used after its completion. This is a wasteful approach, which also limits independent verification of the findings by the community. The latter is a cornerstone of the scientific method.

Recognizing the critical role of testbeds in the evolution of network systems, major governmental research-funding entities have invested significant resources not only in deploying testbed hardware, but more importantly in developing new systems to support and optimize their usage on a global scale. The Global Environment for Network Innovations (GENI) [3] funded by the U.S. National Science Foundation (NSF), and the OneLab Future Internet Testbed [4] project funded by the 6^{th} and 7^{th} European Union Framework Programmes (FP6 & FP7) are major examples of such recent initiatives.

Compared to other scientific research fields, such as the life sciences, a culture of rigorous peer verification of experimental result is so far absent from the networking field. This is mostly due to the fact that even if a similar experimental infrastructure is available, there is currently no unambiguous way to describe an experiment enabling others to repeat it. Thus, to increase the scientific rigour of our field, there is a need for efficient tools and methodologies to support the investigative life cycle. Specifically, we need systematic descriptions of experiments, including the resources used, and measurements taken. As a result, one could then easily repeat the experiment in the same or different context and potentially even more important, allow others to do so.

Optimizing the usage of testbeds on a global scale is another challenging issue. Due to their high setup and operational costs, testbeds are usually limited in terms of capabilities and available technology. For example, the PlanetLab testbed[5] focuses on global overlay experimentation over the fixed wired Internet, while the Orbit testbed[6] focuses on wireless & mobile access networks. Furthermore, in many cases an experiment does not require exclusive use of an entire specific resource (e.g. PC-based node, wireless spectrum). Some solutions exist for sharing under-utilized resources such as PC-based node (e.g. PlanetLab's *slices*). However, sharing other resources such as the wireless spectrum remains non-trivial. The federation of existing and upcoming testbeds under a unified control and management framework would overcome many of these limits, allowing large scale simultaneous resources access and sharing while maintaining administrative control in the hands of the resource owner.

The c**O**ntrol and **M**anagement **F**ramework (OMF) is a

*NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

suite of software components, which provides management, control, and measurement tools & services to users and operators of networking testbeds. OMF was originally developed for the ORBIT wireless testbed at Winlab, Rutgers University [7]. Through active development and extensions at NICTA, it has now evolved into an open source framework, which supports heterogenous wired and wireless resources. OMF is one of the few candidates currently being evaluated as potential testbed framework by both the GENI and the Onelab initiatives. It is currently deployed and used on different testbed in Australia, USA [6], and Europe [8], with many of them in active use 24/7.

The main objectives of the OMF developer community is to produce a unified testbed framework over the next two years which addresses the aforementioned challenges, i.e. supporting full experiment cycle, global testbed federation, and resource sharing. The current OMF release provides significant support for experiment cycle, and initial mechanisms to facilitate the support of federation and resource sharing. This paper presents the architecture and capabilities of this current OMF release. A high deployment and large user base are essential to the success of a control and management framework. Thus to encourage researchers to install and use OMF, this paper describes it from the experimenter's perspective, and focuses on its usability. Detailed descriptions on OMF mechanisms and operations from a testbed management's perspective are not within the scope of this paper, and are available in separate documents [9].

The remainder of this paper is organized as follows. Section 2 discusses some related work. Section 3 provides an overview of the currently operational OMF release, and highlights its new features and improvements compared to the original Orbit software. Section 4 describes the process of developing an experiment for OMF-enabled testbeds. Section 5 presents some case study based on real experiments developed by students and researchers using OMF. Finally, section 6 concludes this paper and provides a summary of potential future works.

2. RELATED WORK

Emulab [10] is a large network emulator, based on a set of computers which can be configured into various topologies through emulated network links. The Emulab control framework supports three different experimental environments: simulated, emulated, and wide area networks. It unifies all three environments under a common user interface. Emulab provides tools to describe a required experiment topology and map it to actual resources. Some control tools are also provided, but they provide minimal features. Emulab and OMF share many design principles with the differences primarily shaped by a focus on different hardware and a different user community.

PlanetLab [5] is a global research platform based on more than 1000 distributed computers, which are hosted by independent organizations worldwide. It is the primary large-scale testbed used for experimental overlay and service oriented systems (e.g. distributed storage, peer-to-peer contend distribution). It is starting to move from a centrally managed testbed to a federation of more regionally focused arrangements under a common code base. This should benefit experimenters in terms of resource availability and diversity. PlanetLab provides a suite of software, which uses virtualization tools to efficiently share the global resources among simultaneous short or long-lived experiments. Similar to Emulab, these tools are essentially focused on resource allocation and access, and only provide minimal supports in describing, controlling and measuring experiments.

Open Resource Control Architecture (ORCA) [11] is an architecture based on the concept of *resource leasing*, which allows the management of diverse computing environments on a common pool of networked hardware resources such as virtualized clusters, storage, and network elements. In the context of networking testbeds, ORCA provide tools for resource providers and brokers to manage and allocate shared resources. It also provides tools for resource consumers (e.g. an experimenter) to negotiate access to, configure and use some of these resources. However, ORCA currently only provides limited support for designing and orchestrating experiments.

The current OMF release is a significant update to the original software developed to control and manage the ORBIT testbed [7]. This former software only supported the specific ORBIT hardware resources (i.e. custom built static nodes with 802.11wireless devices), and provided a limited experiment description language. In contrast, the current OMF supports a large number of different wired and wireless resources. It has a new installation process enabling its deployment on various testbeds with minimal effort [9], and a richer experiment description language, as described in the following sections 3 and 4. These features enable systematic experiment reproducibility on different testbeds, as illustrated in the section 5. Finally, major components of OMF have been re-designed towards enabling federation and resource sharing. For example, the communication schemes between entities in the original ORBIT software have been replaced by a publish/subscribe scheme enabling efficient asynchronous cross-domain message exchanges.

3. OMF SYSTEM OVERVIEW

3.1 Architecture

Figure 1 presents a system view of an OMF deployment. The components of the current OMF architecture are shown in rounded boxes. Future OMF components, i.e. under development, are presented in dashed-line rounded boxes. OMF components communicate through the Control & Management Network, while running experiments have full use of one or more Experimental Network(s), depending on their allocated resources.

The OMF architecture consists of 3 logical planes: *Control*, *Measurement*, and *Management*. In contrast with the defined GENI architecture [3], the *Control Plane* in OMF is dedicated to the control of experiment executions, while the *Management Plane* is responsible for managing the infrastructure. In other words, the *Control Plane* includes the OMF tools that a researcher uses to describe his/her experiments, and the OMF entities responsible for orchestrating it. The *Measurement Plane* includes the OMF tools to instrument an experiment, and the corresponding OMF entities to collect and store measured data. Finally, the *Management Plane* includes the OMF functions and entities to provision and configure the resources, which are provided by the testbed facilities and used by the experiments. More design and implementation details from a development perspective are available at [9].

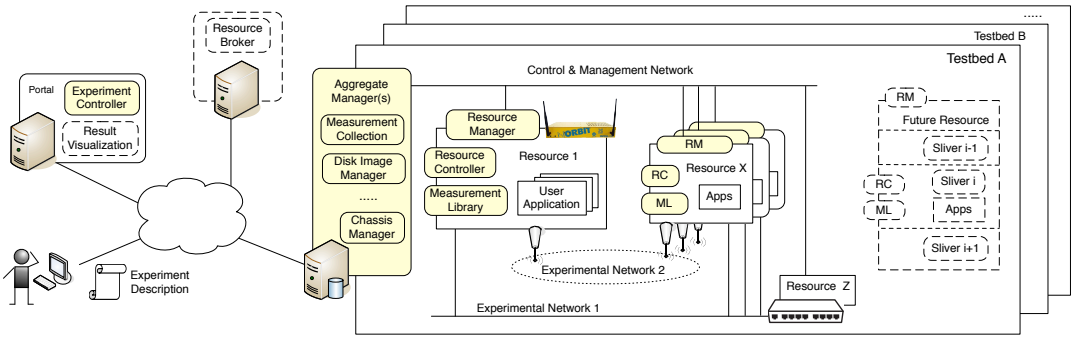

Figure 1: OMF System and Architecture Overview.

3.2 Control Plane

The role of the *Control Plane* is to provide researchers with tools and methods to systematically develop and orchestrate their experiments. A domain specific language (the OMF Experiment Description Language, OEDL) allows an experimenter, we will thereafter call Alice, to write an Experiment Description (ED), which details the resource requirements, their initial configuration, and a state machine describing the time/event-triggered actions required to realize her experiment. This state machine can be defined either along a timeline (e.g. increase the sending rate every 60 seconds) or in reaction to context changes, such as measurements (e.g. "increase sending rate until packet loss > X"), or environment changes, such as location (e.g. "start application X within 200m of specific point"). Alice then submit her ED to an Experiment Controller (EC), which is hosted on a portal. The EC is the control entity orchestrating the experiment on behalf of Alice. In the current OMF, resources allocation is made for a single designated testbed (i.e. no cross testbed experiment support), and access is controlled through time slot reservation. The EC then issues requests on the *Management Plane* to configure the resources as specified in the ED. Once the experiment prerequisites are met, the EC sends directives to the Resource Controller (RC) associated with each resource. Following the ED's state machine, the EC sends directives to the RCs, which are the control entities responsible for executing actions on the resources, such as "start application X with rate of $Ykb.s^{-1}$". When a specified termination state is reached (e.g. "after 10mins"), the EC ends the experiment.

Two additions to this existing *Control Plane* are currently under development. First, the OMF and ORCA teams have started discussions on defining a common ontology to describe testbed resources. This common ontology will provide richer resource description in OEDL, and will allow OMF entities to interact with ORCA's resource allocation and access system. The OMF team is also developing a simple Resource Broker to mediate resources between multiple ECs and testbeds.

3.3 Measurement Plane

The *Measurement Plane* consists of two parts, namely the Measurement Collection Server (MCS) and the Measurement Library (OML). OML allows recording of any relevant metrics and instrumented applications. The available measurement streams can be selected and processed locally (e.g. the mean over a time sliding window) to reflect experimenter needs. The resulting streams are then forwarded to the MCS either in real-time or batch mode to minimize interference with the experiment itself. The MCS stores them in an SQL database created for each experiment instance. The user can directly run SQL queries against it, or retrieve a data dump from it. Standing queries can feed events back into the EC to support steerable experiments.

A Result Visualization module is being developed as an addition to the Portal. This module will provide simple visualizations of the measured experiment metrics (e.g. metric against time). In many cases, these visualization will provide valuable initial feedbacks on the scheme under experimentation, with the database still accessible for more comprehensive post-processsing.

3.4 Management Plane

The *Management Plane* consists of a set of components, which provide services to manage the testbed infrastructure. Examples of such management operations are the loading and saving of a particular disk image on a PC-based resource, the rebooting/reset of a resource, the setup of an experimental network with a specific topology or characteristics (e.g. using VLANs, Traffic Shaping).

Borrowing from the GENI terminology [3], OMF combines these management services into an Aggregate Manager (AM). Thus, the AM is a collection of services, which can be deployed across multiple servers for performance and redundancy reasons. In the current release, the AM accepts request from the ECs or the testbed operator, and sends corresponding commands to the Resource Manager (RM) running on each resource[1]. The RM is responsible for performing the management operation on the resource, e.g. write image on hard-disk, create a VLAN including nodes X and Y. The software architecture of the AM and RM have been designed specifically to allow easy modular additions for new type of resources. For example, the OMF team is developing an extension for the management of routers, which will allow an Experimental Network to access external networks (e.g. the Internet, or a foreign testbed for cross-testbed experiments). Another NICTA team is developing an extension for the management of MICA2 Motes [12], which will allow wireless sensor network experiments. Finally, other OMF partners are working on the integration

[1]The RM function is provided by the vendor for some resources such as a network switch.

```
# Part 1 - Describe the resources required for this experiment

defGroup('source', [5]) {|node|

  node.prototype("test:proto:udp_sender", {
    'destinationHost' => '192.168.0.10',
    'localHost' => '192.168.0.5'
  })
}

allGroups.net.w0 { |w|
  w.type = ''g''              # Use 802.11g
  w.mode = ''ad-hoc''         # Set interface in 'ad-ho' mode
  w.channel = ''6''           # Use channel 6
  w.essid = ''simple''        # Set SSID
  w.ip = ''%192.168.0.%i'' # Set the IP address to 192.168.0.i (i=5)
}

# Part 2 - Describe the state-machine and tasks to execute
# Here we only define one state 'whenAllInstalled' and associate 8 tasks to it
whenAllInstalled() { |node|
  wait 20
  info("Start all the applications")
  allGroups.startApplications
  wait 30
  info("Stop all the applications on 'source' group")
  group('source'). stopApplications
  info("Now Stop the experiment")
  Experiment.done
}
```

Listing 1: Example of an experiment description

of WiMax base station and node resources.

In a future OMF release with resource brokerage capability, the AM will only accept management requests coming from ECs, which have previously gained access rights for the corresponding managed resources. Another feature under development is the virtualization of the resources when permitted by the underlying technology. This is illustrated by the gray "Future Resource" in figure 1. Such feature will allow multiple users to share the resource and perform concurrent experiments in different slices, similar to PlanetLab [5]. The RM will manage the concurrent slivers (e.g. create/delete, monitor memory/cpu usage), each of which running its own instance of RC and OML.

4. DEVELOPING AN EXPERIMENT

4.1 Process Overview

Let us assume that an experimenter named Alice wants to evaluate a new cross-layer ad-hoc routing algorithm. She first needs to register to an organization providing an OMF-enabled testbed (e.g. Winlab, NICTA, NITLab). Then she will develop a description of the experiment to perform using OEDL. After reserving a time-slot on a testbed, she will submit her experiment to the *Control Plane*, which will execute it. Alice can then use the *Measurement Plane* services to access and analyze her experimental results.

4.2 Experiment description and OEDL

OMF provides a Domain-specific Language named OEDL to describe an experiment. OEDL is based on the Ruby scripting language [13], and uses Ruby's meta-programming capabilities to provide experiment-specific commands and statements. As a new user, Alice does not need to know Ruby to write an Experiment Description (ED) with OEDL. She can get started with a basic programming knowledge and the use of the OEDL commands described at [9].

As introduced in section 3.2, an ED is composed of 2 parts:

- Resource Requirements and Configurations, which enumerate the resources that are required by the experiment, and describe the different configurations that need to be applied on them,
- Task Descriptions, which are essentially contained in a state-machine that enumerates the different events, states, and associated tasks to perform with the resources in order to realize the experiment

```
# Define a OMF description around 'ping'

defApplication('pingWrapper', 'pingApp') {|app|

  app.shortDescription = "This is a simple description around ping"
  app.path="/bin/ping"

  app.defProperty("dst", "Destination Address or Name", nil,
                  {:dynamic => false, :type => :string, :use_name => false})

  app.defProperty("count", "Number of probe packet to send", ?c,
                  {:dynamic => false, :type => :integer})

  app.defProperty("interval", "Time interval between packets in a flood", ?i,
                  {:dynamic => false, :type => :integer})
}
```

Listing 2: Example of an application description

Listing 1 shows an example of a simple ED. In this example, the *defGroup* block assigns a role (UDP source) to node 5, the *allGroups* block describes common attributes for all groups. These attributes can also be specified in the group definition. Finally the *whenAllInstalled* block represents the different actions to perform when all nodes are fully configured. OEDL provides a large set of other commands, which allow more sophisticated EDs than shown in this listing. For example, a *defTopology* command allows Alice to define complex and re-usable resource topologies. Other OEDL features include the support for per-group or per-resource configuration (instead of using *allGroup* as in Listing 1), and the support for more complex user-defined states.

To fully benefit from the *Control plane* features, Alice needs to develop OEDL descriptions for her software applications. These descriptions provide an interface for the *Control plane* to automate the deployment, configuration, execution, and instrumentation of Alice's software. Listing 2 shows an short example of the OEDL description of the "ping" tool. Finally, OMF provides another abstraction, named *prototype*, which allow Alice to refine her application descriptions for re-use across many different experimental cycles. Further examples and a detailed tutorials on these features are available at [9].

4.3 Experiment execution and results

As mentioned earlier, to execute an experiment Alice needs to submit its ED to an OMF Experiment Controller (EC). The EC is accessible on an OMF Portal, via a command line application called `omf`. The typical command to execute an experiment is: `omf exec myED`, where myED is the name of the file describing the ED. During the experiment execution, Alice can observe its progress or optionally interact with it on the command line or through a web browser connected to the EC's integrated web server.

To access the experiment measurements, Alice has the options of either using her own tools to retrieve and manipulate the SQL result database, or using a dedicated OMF Portal service named *result*. Through an HTTP API, the *result* service allows any given SQL query to be executed against the database, and returns its response as an XML document. This XML response can then be used in any processing and

plotting scripts or tools. Additional visualization services are currently being developed to provide users with measurement graphs, which may provides initial indications for further result analysis.

4.4 Experimenting further

An experiment is seldom an isolated event, in most cases Alice will perform many experiments with different versions of her algorithm, software, configuration parameters, scenarios, and topologies. Thus OMF has been specifically designed to assist her in managing this *experimental evaluation cycle*. Examples of OMF features in that regards include: the support for high-level descriptions of experiments with configurable parameters, the use of version-controlled packages for user application deployment, the systematic collection of measurements in databases, the support for re-usable resource topologies across different experiments, the ability to run a single un-modified ED on different testbeds with different resources and surrounding environments.

To further support users in their *experimental evaluation cycles*, the future OMF will extend to an entire experiment the concept of package management, which is present in many OSes. In this approach, an experiment will be similar to a *meta package*. It will depend on many other packages, e.g. one with the ED, one with the used applications, one with the used topologies, or the measurement database. All of these will be maintained in a version-controlled repository on an OMF Portal, which will also provides tools for easy package creation. This approach will provide support for systematic experiment life-cycle and reproducibility.

5. OMF CASE STUDY

This section presents some examples of experiments developed and performed by students and researchers using OMF. Thus, these experiments are real instances of OMF being used to evaluate the performance of some research algorithms and architectures. More details on these experiments (such as their complete EDs) and further examples on how to use OMF are available at[9].

Link connectivity measurement

This first case study experiment was developed by a student at NICTA, who studied the characteristics of wireless links between ad-hoc WiFi stations. This experiment consists of some 802.11g wireless stations (nodes $[1, x]$) exchanging beacons with each other and measuring the corresponding link quality. An additional station (node TG) was added to generate some background traffic, thus potentially impacting the link characteristics between the remaining stations. Figure 2(a) shows the result when the overlapping channels 6 & 7 are used for beacon and background traffic, while Figure 2(b) shows the use of disjoint channels 6 & 11. Edges represent the average Received Signal Strength Indication (RSSI) in dBm measured by the beaconing peer[2], wider and darker edge indicating higher RSSI value. This case study highlights OMF's capabilities to instrument experiments for systematic measurement collection.

Multi-path extension to OLSR

This second case study was developed by Jack Tsai from the University of New South Wales (Australia), who studied

[2]Only one direction of the link is shown for each node pair.

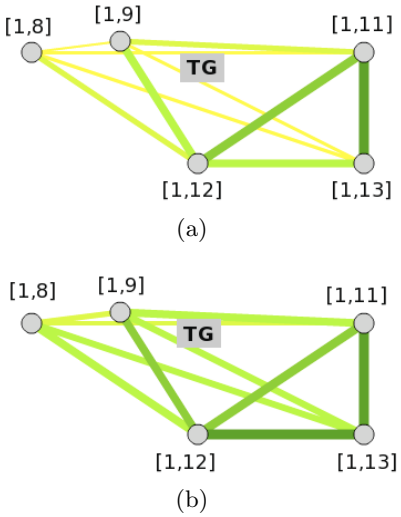

Figure 2: Link connectivity between wireless nodes

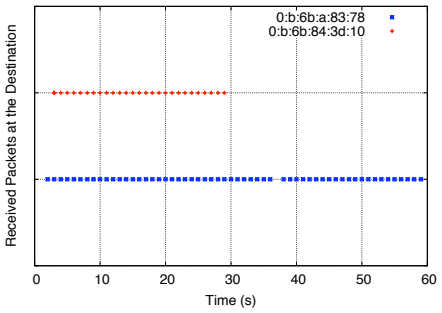

Figure 3: Received packets using Multi-path OLSR

multi-path routing protocols for ad-hoc wireless networks. He designed a QoS-based multi-path discovery algorithm [14], and implemented it as an extension to an existing Optimized Link State Routing Protocol (OLSR) implementation [15]. This prototype was then evaluated using OMF on an indoor wireless testbed. One of his experiments consisted of 5 ad-hoc stations all running the multi-path OLSR daemon. Two of these stations (nodes A & B) were out-of-range of each other, and were exchanging some traffic via multiple disjoint paths. These paths were provided by the remaining 3 nodes acting as relays. After 30s of traffic exchange, one of the relay nodes was shutdown, thus removing any related paths. Figure 3 shows the received packets at node B coming from 2 different MAC addresses (not from A), thus 2 different paths. This study demonstrates OMF's capability to create complex topology via OEDL. Indeed in many testbeds, all nodes may be in radio range of each other. However, OEDL allows the description of a topology with out-of-range nodes, and OMF includes the related mechanisms to enforce such topology with a choice of tools, such as layer 2 frame filtering or noise generation.

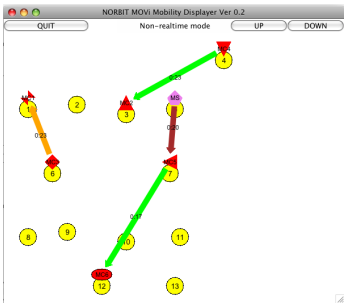

Figure 4: Traffic visualization during a MOVi-TV experiment using OMF

Mobile content distribution with MOVi

This last case study experiment was developed by Hayoung Yoon from the Gwangju Institute of Science and Technology, South Korea. He proposed a novel cross-layer opportunistic peer-to-peer content delivery architecture for mobile devices, named MOVi-TV [16]. He implemented a prototype of MOVi-TV and evaluated it on two different OMF testbeds (NICTA, Australia and Rutgers University, USA). To emulate mobility on fixed wireless nodes, he also implemented a scheme migrating applications and their states between nodes, according to a predefined trace. This case study demonstrates the support provided by OMF in managing a full experiment cycle. Indeed using the exact same ED, this student was able to systematically reproduce his complex experiment on two distinct testbeds, which provided different environments, i.e. a corporate deployment (i.e. building floors with steel and concrete walls) and a open space deployment (i.e. all nodes in line-of-sight). Furthermore, this case study also demonstrate OMF's capability of providing researchers with kernel-level access to testbed resources, hence supporting evaluations of cross-layer prototypes. Figure 4 shows a screen capture of a traffic visualization tool using the collected measurements of a MOVi-TV experiment. The comprehensive descriptions and results of these experiments are detailed in [17].

6. CONCLUSION

This paper presented an overview of OMF, a unified control, measurement, and management framework for networking testbeds. The objective of this overview was to provide network researchers a short yet comprehensive introduction to OMF's capabilities and utilization from an experimenter's perspective. In that regard, this paper first described the system architecture of OMF with its current and future components, which are relevant to an experimenter's understanding of the whole framework. It then discussed the different steps involved in developing and performing experiments with OMF. Finally, it presented some case study experiments from real research initiatives.

OMF aims at providing unified and systematic means to develop and perform full experiment cycles over a global federation of networking testbeds. Many of the capabilities required to achieve this goal are present in the currently deployed and actively-used OMF release. However, a dedicated and growing developer community [9] is actively pursuing the missing features described in the the previous sections and experimenting with many more.

Acknowledgements

The authors would like to thank Jack Tsai and Hayoung Yoon for the permission to use some of their experiments as OMF case studies. This work was achieved in the context of the OneLab project funded by the European Union 7^{th} Framework Program, and the GENI (Global Environment for Network Innovations) initiative funded by the U.S. National Science Foundation.

7. REFERENCES

[1] "NS3 Network Simulator," *at: http://www.nsnam.org/.*

[2] "OMNet++ Simulator," *at: http://www.omnetpp.org/.*

[3] D. Clark *et al.*, "GENI Design Document 06-28," *http://www.geni.net/GDD/GDD-06-28.pdf.*

[4] "OneLab: Open Federated Laboratory for Future Internet Research." *at: http://www.onelab.eu.*

[5] PlanetLab Consortium, "Planetlab: An open platform for developing, deploying, and accessing planetary-scale services," http://www.planet-lab.org/.

[6] D. Raychaudhuri *et al.*, "Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols," in *IEEE Wireless Communications and Networking Conference*, 2005.

[7] M. Ott *et al.*, "ORBIT testbed software architecture: Supporting experiments as a service," in *Proceedings of the IEEE Tridentcom 2005*, Trento, Italy, Feb. 2005.

[8] "The Network Implementation Testbed Laboratory," *at: http://nitlab.inf.uth.gr/NIT/NITLab.*

[9] "OMF, the testbed control and management framework," *at: http://omf.mytestbed.net.*

[10] B. White *et al.*, "An integrated experimental environment for distributed systems and networks," in *Proc. of the Symposium on Operating Systems Design and Implementation*, Dec. 2002.

[11] "The ORCA GENI Control Framework," *at: http://www.nicl.cs.duke.edu/orca/.*

[12] "Crossbow MICA2 Wireless Platform for Low-Power Sensor Networks." *at: http://www.xbow.com.*

[13] "The Ruby language," *at: http://www.ruby-lang.org.*

[14] J. Tsai and T. Moors, "Minimum Interference Multipath Routing Using Multiple Gateways in Mesh Networks," in *IEEE Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, 2008.

[15] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," in *IETF RFC 3623*, 2003.

[16] H. Yoon *et al.*, "On-demand Video Streaming in Mobile Opportunistic Networks," in *IEEE PERCOM*, 2008.

[17] ——, "Mobility emulator for DTN and MANET applications," in *ACM Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WINTECH)*, 2009.