Mobile Experiments Made Easy with OMF/Orbit*

Christoph Dwertmann[†]
Maximilian Ott[†,§]
Marco Gruteser[§]

Mesut Ergin[§]
Thierry Rakotoarivelo[†]

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

Guillaume Jourjon[†]
Ivan Seskar[§]

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

1. INTRODUCTION

Experimental facilites (or testbeds) are instrumental in the development and evaluation of new Internet technologies. Evaluations based on simulation and emulation do provide valuable, yet inexpensive insight into the performance of new networking technologies at large scales. However, simulators and emulators inherently make simplifying model assumptions. Thus, convincing the networking industry and community to widely adopt and deploy new algorithms or mechanisms also requires comprehensive analysis in real-world settings.

Supporting researchers in developing and conducting these experiments is challenging—particularly on mobile testbeds, where nodes may be disconnected from the control network and node movemenets may not be under the control of experimenters. Some tools have been developed for existing mobile testbeds [1], but to our knowledge no comprehensive framework has been developed.

Our c**O**ntrol and **M**anagement **F**ramework (OMF) is a suite of software components which provide control, measurement, and management tools & services to users and operators of networking testbeds to address this need. A key feature of OMF is that it facilitates describing software installations, experiment workloads and actions, and measurement collection in a single integrated experiment description (script). This experiment description can then also serves to document the experiment and can be shared with other researchers to allow repeating the experiment. We hope that this will aid the development of a culture of rigorous peer verification of experimental results and increase the scientific rigour of our field.

Specifically for mobile testbeds, OMF provides time-based scripting and automated caching of experiment results to allow experiments to continue when nodes are disconnected from the control network. Additional functions such as location-based scripting are currently under development.

OMF was originally developed for the (stationary) ORBIT wireless testbed at Winlab, Rutgers University[1]. Through active development and extensions at NICTA, it has now evolved into an independent framework, which supports heterogenous wired or wireless resources. OMF is one of the candidates currently being evaluated as potential testbed frameworks by both GENI[2] and FIRE[3] funded testbed initiatives, such as Onelab[4].

OMF is currently deployed and used on different testbed in Australia, USA, and Europe.

The following demo will showcase OMF's current capabilities with an experiment demonstrating a common investigative life cycle.

2. OMF SYSTEM OVERVIEW

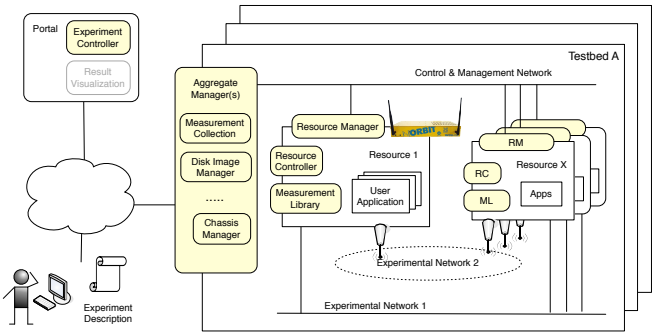

Figure 1: OMF System and Architecture Overview.

Figure 1 presents a system view of OMF. The various components of the OMF architecture are shown in light colored rounded boxes (e.g. Experiment Controller). OMF components communicate through the Control & Management Network, while a running experiment has full use of one or more Experimental Network(s), depending on the capabilities of the deployed testbed. For more details about OMF architecture please refer to *http://omf.mytestbed.net*.

For example, the outdoor testbed at WINLAB, Rutgers University was recently extended with a WiMax basestation and respective network cards in some of the mobile nodes. It is assumed that some of these mobile nodes are mounted in campus buses[5] and some in fixed locations (infrastructure locations and/or "parked cars") and are available to experimenters.

*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.

[1]www.orbit-lab.net

[2]www.geni.net

[3]http://cordis.europa.eu/fp7/ict/fire/

[4]www.onelab.eu

[5]While the bus deployment is still under negotiation, we can perform the experiments by installing the nodes in regular cars driven specifically for such an experiment as this is an operational rather than a technical issue.

The role of the *Control Plane* is to provide tools and support to researchers in developing and orchestrating their experiments. A domain specific language (OEDL) allows the experimenter to write an Experiment Description (ED).

The user submits the ED to an Experiment Controller (EC), which is usually hosted on a Portal. The EC first coordinates the provisioning and configuration of the various requested resources and then orchestrates the experiment by sending directives to the Resource Controller(s) associated with each resource. When the experiment's objectives have been reached, the EC terminates the experiment.

The *Measurement Plane* consists of two parts, namely the Measurement Collection Server (MCS) and the Measurement Library (OML). OML allows recording of many relevant system metrics as well as from instrumented applications.

The *Management Plane* consists of a set of components, which provide functions to manage the testbed resources. Examples of such management operations are the loading & saving of a particular disk image on a PC-based resource, the rebooting/reset of a resource, or the setup of an experimental network with a specific topology or characteristics.

3. SCENARIO

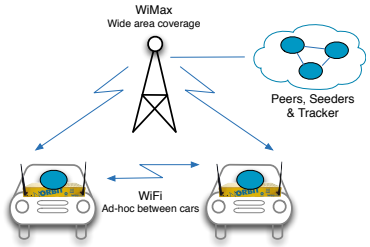

Figure 2: Demo scenario.

The demo scenario assumes that an experimenter, let us call her Alice, wants to evaluate an "ad-hoc enhanced bittorrent client for vehicular networks". She envisions that most cars in the future will be able to connect to a "mostly-on" wide area wireless network and will also be equipped with a high-speed short range radio to establish ad-hoc networks with other nearby cars. She has designed an extension to the bittorrent (BT) protocol which can take advantage of the additional, transient communication channel.

She is especially interested in the impact of the encounter duration of vehicles on various stages of a BT download.

Alice's next step is to add her protocol extensions to an existing BT client. To maintain maximum flexibility she decides to first re-factor the client in such a way that all lower-level networking as well as systems functionality is concentrated in pluggable modules. By replacing them with implementations using an event-based simulation she can first study her protocol to obtain a better understanding of its behavior and hopefully also remove most bugs. Alice already instruments her code using OML, which can also be used independent of OMF or a testbed setting.

Once she has her code tested on a local platform she prepares ED script to deploy it on the mobile testbed depicted in Figure 2. First, she includes descriptions of her own application and its dependencies and configuration options. This will allow the OMF framework to automate installation of her software on all involved components. It will also ensure that newer versions fixing the inevitable bugs will be pushed out efficiently. OMF will use the OML framework to record which versions were used by each individual experiment ensuring proper provenance of experiment results. Alice also learns that all nodes in the Orbit outdoor testbed are equipped with GPS receivers and their location is constantly recorded via OML. This will allow her to geo-register many of her measurements with little effort. Alice would then choose to configure OML to either report live the measurements or store them locally and send them when a car is next to a wifi access point in order to not overload the WiMax channel. This feature is transparent for Alice and orchestrated directly by OMF.

As measurements of the experiment itself, as well as the environment (such as mobile location) can be be performed in real-time, Alice can choose to observe the experiment and even interact with it. For instance, she can dynamically steer the experiment or conclude it when the collected measurements have reached statistically significance. Alternatively, she can simply submit the ED script to the experiment queue and wait for a notification when the experiment finishes.

While OMF provides the building blocks, the necessary methodologies and best practices still need to be developed.

4. DEMONSTRATION DETAILS

To demonstrate the various aspects of the above scenario we re-factored a BT client named RubyTorrent[6] to concentrate all socket interactions as well as message marshaling into a separate PeerChannel class. An alternative implementation of its API is based on Jeff Rose's GoSim discrete event simulator[7] which then allows us to simulate multiple clients of a swarm.

We further implemented a simple extension to BT's peer & piece selection algorithm which takes advantage of the characteristics of the two wireless access networks.

Our extended BT client can be packaged into a Debian package and uploaded to the OMF repository. describes all the parameters of an application, as well as all the OML measurement points it is instrumented with.

All this allows an experimenter to describe the entire experiment with an OIDL script and the OMF framework to select and configure all required resources, such as nodes, wireless interfaces, router & firewall settings, operating systems, drivers, and as well as software packages. Depending on the operation mode, OMF will then allow the user to execute the experiment either interactively or unattended. We will demonstrate this by executing an experiment, observing measurements in real-time and potentially changing certain parameters interactively.

More details on this demo and OMF can be found at http://omf.mytestbed.net/wiki/omf/SC09Demo.

5. REFERENCES

[1] A. Balasubramanian et al. Interactive WiFi Connectivity for Moving Vehicles. In *Proceedings of ACM SIGCOMM 08*, 2008.

[6] http://rubytorrent.rubyforge.org/
[7] http://rosejn.net/