FUTURE INTERNET TESTBEDS
EXPERIMENTATION BETWEEN
BRAZIL AND EUROPE

Grant Agreement No.: 288356
CNPq Grant Agreement No.: 590022/2011-3

FIBRE-EU

Future Internet testbeds/experimentation between BRazil and Europe – EU

Instrument**:** ***Collaborative Project***
Thematic Priority: ***[ICT-2011.10.1 EU-Brazil] Research and Development cooperation, topic c) Future Internet – experimental facilities***

D3.4 Final version of the enhanced

OMF control framework software

Author: WP3
Revised by: Leonardo Bergesio, Sebastià Sallent (i2cat)
Due date of the Deliverable: Month 34
Actual submission date: March 31st, 2014
Start date of project: June 1st 2011 Duration: 34 months
version: v.1.0

Project co-funded by the European Commission in the 7th Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

* This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration

FP7 Grant Agreement No.	288356
Project Name	Future Internet testbeds/experimentation between BRazil and Europe – EU
Document Name	FIBRE-D3.4-v1.0
Document Title	D3.4 - Final version of the enhanced OMF control framework software
Workpackage	WP3
Authors	Dimitris Giatsios (UTH)
Editor	Dimitris Giatsios (UTH)
Reviewers	Leonardo Bergesio (i2Cat) Sebastià Sallent (i2Cat)
Delivery Date	31/03/2014
Version	V1.0

Abstract

This deliverable provides documentation regarding the extensions to the OMF framework carried out during the project. It describes the overall structure of the newest version of OMF, and the enhancements made to the measurement framework, OML. Subsequently, it presents in detail developments related to specific types of resources, such as WiMAX infrastructure, OpenFlow switches and software defined radio resources. Finally, it outlines the basics of OMF Broker, the new, SFA-compliant, resource brokering scheme for OMF-based testbeds, which was developed throughout the project.

TABLE OF CONTENTS

1 Acronyms

AM	Aggregate Manager
BS	Base Station
EC	Experiment Controller
GENI	Global Environment for Network Innovation
NITOS	Network Implementation Testbed using Open Source Code
OF	OpenFlow
OMF	cOntrol and Management Framework
OML	OMF Measurement Library
REST	Representational State Transfer
RC	Resource Controller
SFA	Slice-based Federation Architecture
SNMP	Simple Network Management Protocol
XMPP	eXtensible Messaging and Presence Protocol

2 Scope

This document presents the enhancements made to the OMF framework throughout the project. It attempts to strike a balance between a high-level presentation of OMF functionalities and challenges associated with new types of resources, and a more in-depth description of technical details.

3 Reference Documents

[1] http://mytestbed.net/projects/omf6/wiki/ArchitecturalFoundation2

[2] http://mytestbed.net/projects/oml/wiki/Releases

[3] http://www.postgresql.org

[4] https://www.irods.org

[5] http://rubygems.org

[6] "GENI WiMAX wiki," Available: http://wimax.orbit-lab.org/

[7] USRP platforms: http://www.ettus.com/

[8] P. Sutton et al., "A Reconfigurable Platform for Cognitive Networks", in proc. IEEE CROWNCOM, 2006

4 Moving from OMF 5.x to OMF 6

In the official beginning of the FIBRE project, the current version of OMF was 5.3. Since then, a release of the next stable version, OMF 5.4, took place during M7 of the project. Meanwhile, a new version with major architectural changes was being designed by the OMF developing team at NICTA, resulting in a developer preview release of OMF 6 in M13 of the project. A Beta version was released later, the latter being the current working version.

OMF 6 has brought a number of key changes with it, which we describe in detail in this section. More importantly, it introduced a new model to describe and operate experimental resources, and a new messaging protocol to interact with them. This new model and protocol make it easier for third parties to extend OMF with the support of new features and specific resources.

4.1 Rationale

The design of OMF 6 was driven by a careful examination of the notion of experiments and testbeds in the networking field, as well by a clear discrimination of the entities, the actors and their interactions.

Each experiment is comprised of *resources (aka components)*, *tasks* and *observations (aka measurements)*. Similar experiments, such as the ones performed by collaborating experimenters as part of a research project, can be grouped into *slices (aka studies)*. So resources are in general assigned to slices rather than individual experimenters (of course a slice can comprise a simple experimenter). The basic actors are *resource providers* and *experimenters*. Both sides belong to *organizations*, which provide the legal and operative rules governing the interaction among everyone. The relationship between these parties is depicted in Figure 1.

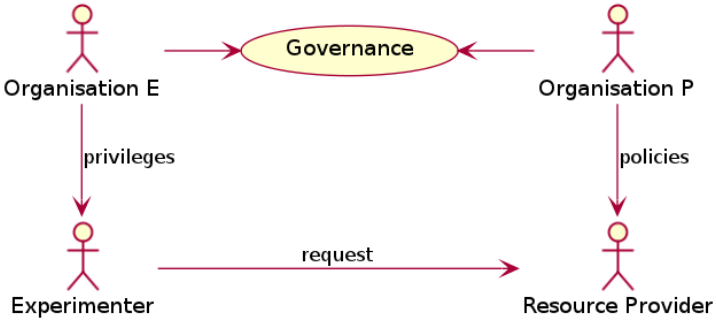

Figure 1: Relationship between basic actors

The basic idea behind the design has been “everything is a resource”, which models even entities belonging to the scaffolding around the system under study as resources (e.g. measuring devices). This approach has allowed for a more natural and generic architecture, which we describe next, starting by the resource model.

4.2 Information model

The basic building block of the model described is the resource, which is defined as an entity with:

- A globally unique id
- A type
- An unordered list of properties

Properties are defined as:

- A key (i.e. the property's id)
- A value and associated type
    - A property can be either of a simple type, or a reference to another resource

Special types of components are containers which can include other components.
An example of a resource with two properties, one of which is a reference to another resource, is given in Figure 2.

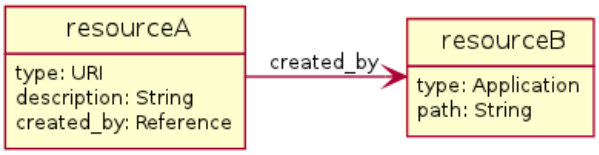

Figure 2: Example of simple resources

Each resource has a lifecycle, as depicted in Figure 3.

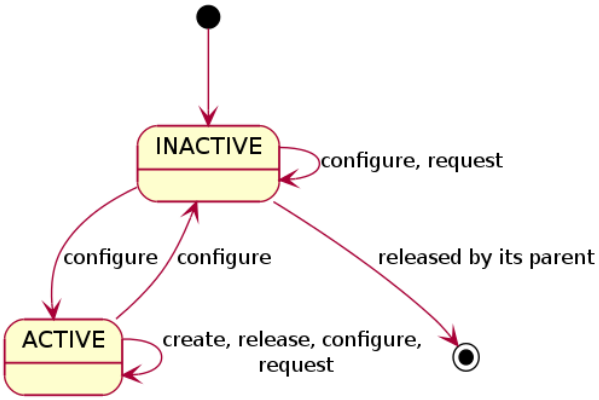

Figure 3: Lifecycle of a resource

A new component is created by sending a *create* message to an existing component. This new component is initially in the *inactive* state, and may transition to the *active* state immediately. A *configure* message requests a component to change some of its properties to new values.

The *request* message requests a component to report its current state. The component responds by publishing an *inform* message, which contains a list of its properties, their associated values and optional additional messages. To discard a component, one must send a *release* message to its parent. Upon receiving that message, the parent is responsible for ensuring the "clean" termination of that child.

A publish-and-subscribe (pub-sub) message system is adopted for handling communications between resources and the entities interacting with them. Participants can create *topics*, subscribe to them and publish messages to them. A message published to a given topic is forwarded to all subscribers of the topic. The current implementation is using an XMPP-based pub-sub system, but other alternative systems can also be used.

The basic message pattern in the OMF 6 architecture assumes that every active resource is associated with a topic (address) and all messages to and from this resource are published to this topic. Messages *create*, *configure*, *request* and *release* are sent to the resource, while the *inform* message originates from the resource. We adopt the convention that every message contains a message ID and that the *inform* message contains a list of message IDs it received since the last *inform* message.

The convention used in OMF 6 for the mapping between a resource ID and the name of its associated topic is the following: a resource with the name "foo" provided by the institution "bar" corresponds to the topic address "foo@bar".

As mentioned earlier, some resources are containers that include other resources. These containers also have globally unique names that map to specific topic addresses. All the resources that are contained within a container resource must subscribe to its topic address. This does not imply any automatic forwarding. Every entity that wants to send a message to a specific resource must be subscribed to the resource's topic.

We present some more details by means of an example. Figure 4 presents the organization of pub-sub topics for a typical scenario involving one slice with multiple experiments and multiple institutions providing resources.

- The pub-sub system has a topic for each resource provided by an institution. We define a Resource Proxy/Controller entity, which is the interface to a real resource (e.g. a VM, an application) and which handles messages for the resource and commands it. For example, R_{Res1} is the Resource Proxy/Controller for resource Res1 in Figure 4.
- A given slice X has its own topic "slice X", created by the slice's Principal Investigator. For each resource assigned to the slice there is a corresponding topic under the "resources" topic. The Resource Proxies of these resources subscribe to the corresponding topics.
- A given experimenter running an experiment Exp1 within slice X must create a topic "exp1" for his experiment, under the "slice X" topic. He must also create a topic for every resource used in the experiment, under the topic "exp1". In addition, for each

resource container defined and used by Exp1, a corresponding topic needs to be created (e.g. "group i" in the figure). All resources included in the container must subscribe to its topic.

The "Experimenter" entity in the figure refers to the software used by the experimenter to interact with the resources within a slice.

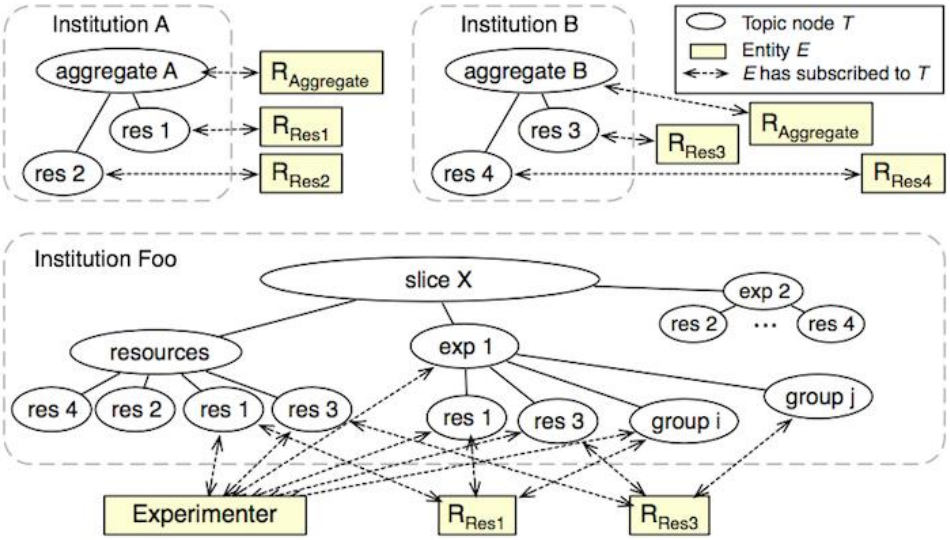

Figure 4: Organization of pub-sub topics in typical scenario example

The content of the messages is described in an XML format. For the exact protocol we refer the reader to [1]. Below we present an example with a *create* message sent to a server in order to create a virtual machine and the respective *inform* message.

```
<publish node='server1'>
<create xmlns="http://schema.mytestbed.net/6.0/protocol"
message_id="1ab3f0">
  <context_id>d234b7</context_id>
  <resource_type>virtual_machine</resource_type>
  <property key="vm_type">xen</property>
  <property key="os" type="xsd:string">ubuntu-11.10</property>
  <property>
      <key>memory</key>
      <type>xsd:int</type>
      <value>2</value>
      <unit>GB</unit>
   </property>
  ...
</create>
<inform xmlns="http://omf.mytestbed.net/6.0/protocol"
message_id="27ab23">
  <context_id>d234b7</context_id>
  <inform_type>RESOURCE_CREATED</inform_type>
  <resource_id>foo</resource_id>
  <resource_address type="xmpp">foo@bar</resource_address>
</inform>
```

5 Enhancements in the OML Measurement Framework

The main enhancements in OML introduced gradually during the project lifetime are listed below. A full list of changes introduced with the new releases can be found in [2]. It is also specified if they are related to the OML server, OML proxy server, liboml2 (client OML C library), OML4R (client OML Ruby library) or oml-apps (set of popular networking research applications wrapped with OML):

- Liboml2: New output serialization and buffering scheme in the client library to decouple backend processing from application-thread sample injects. New client lib command line option --oml-bufsize controls the amount of buffer space available in the buffer that sits between the inject thread and the output stage thread.
- Liboml2: Reconnect to server in case of disconnection
- Oml2-server: PostgreSQL [3] support is now complete and considered to be supported. oml2-server now supports --pg-connect and --pg-user to help configure the connection to the PostgreSQL server.
- Oml2-server: Introduce event hook mechanism to react to event within the server (e.g., push a measurement database to iRODS [4] when the last client has disconnected)
- OML4R: gem pushed to Ruby Gems [5]
- OML4R: Support for sending measurements to different streams
- OML4R: Support for reconnection
- Support for user-defined metadata (units, references, etc...) - C client only for the RC
- Support for globally unique reference IDs between MPs (allows the interpretation of some data in context of others) - C client only for the RC
- Support for Boolean as OML data type - C client only for the RC
- IPv6 suport in client and server
- Support for vectors in OML server and client

Oml-apps: A new package, oml2-apps-omfdefs is a convenience package has been created; it installs all OML2 application definitions in the path of the OMF EC's path so they can be used directly from your experiments as oml:app:APP, without having to copy their description in your OEDL script.

The current version of OML is 2.11.0rc, released on Nov. 11th 2013 (at the project beginning the version was 2.4.0).

6 OMF support for WiMAX

6.1 UTH testbed extension

During the project, the testbed of UTH was extended with a WiMAX component, in accordance with the DoW. The related infrastructure consists of:

- A WiMAX Base Station provided by Airspan Networks Inc. (in order to enable experimentation on top of it, features such as authentication with another server or routing via an ASN Gateway have to be disabled)
- A Greenpacket DX-250 WiMAX to WiFi access point
- Teltonika UM 6225 USB dongles
- Greenpacket UT WiMAX USB Dongles
- Intel Centrino Advanced-N + WiMAX 6250 mini pci express cards

The setup NITOS testbed is currently using is a fixed setup (employing no mobility between BSs) that does not require an ASN-GW installation. The overall topology is summarized in the following figure.

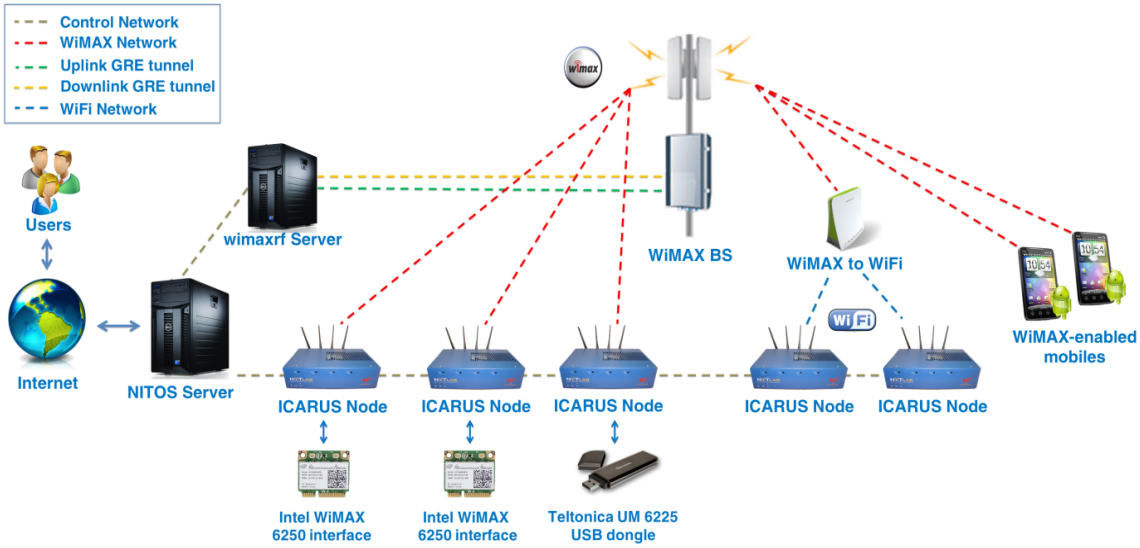

Figure 5: Architecture of WiMAX component at NITOS testbed

6.2 OMF drivers for client devices

The aforementioned types of client devices are currently exploitable in the NITOS testbed, and are used to set up complex experimental scenarios involving WiMAX technology. UTH has enhanced OMF by means of Resource Controllers (RCs) that control and instrument experimentation with these devices.

Concerning the control and management of the WiMAX nodes, additions to current RCs have been added that feature:

- Loading the appropriate WiMAX driver during experimentation

- Setting up experimentation specific parameters to the WiMAX interface (IP addresses, turning it on, connecting to the BS)
- Wrappers have been written that control the WiMAX utilities that were provided by linuxwimax.com and are integrated in the baseline image of NITOS nodes.

Concerning the connection of the Teltonika modems, a separate OMF driver has been written from scratch, which facilitates the procedure of connecting to the BS since it is not that simple as the other modules. The overall functionality that is provided by the OMF driver is summarized as follows:

- Connecting to network (frequency, bandwidth)
- Disconnecting from currently associated network
- Setting connection options (adaptive MCS, report MCS, ARQ, etc.)
- IP address and netmask
    - If a DHCP server is present, Teltonika USB device will get IP address automatically. However, the IP address is not properly displayed due to the way that the Teltonika modem operates. Therefore, we have created an application called "fixip" that can help the experimenter find out the actual IP address of the device.
    - If no DHCP server is available, the experimenter can setup a static IP directly from an OMF Experiment Description script.
    - Either way, we use telnet libraries for Ruby to communicate with the icc0 interface, which is the actual WiMAX interface.

Apart from the first two additions (loading the driver, setting up addresses etc.), that are rather straightforward as we prepared an altered version based on the existing implementation over the WiFi experimentation, the wrappers provide an interface to OMF to control resources using the wimaxcu application. The application can control the WiMAX interface, turn hardware and software radios on/off, connect the interface to a BS, scan for existing BSs, etc.

6.3 Controlling the Base Station

The Base Station accompanying software is provided by Airspan, and called Netspan. This software is using the SNMP protocol to setup parameters of the Base Station by sending requests to it that the user is sending via a web interface. However, this solution does not scale, and requires a Windows Server Installation. Therefore, UTH developed an OMF AM service that controls the Base Station via a REST-based API. UTH moved to this solution after discussing with NICTA and the RUTGERS University, New Jersey, who have developed a similar service, called wimaxrf. The extensions were based on an existing implementation [6] that was designed to run over a WiMAX BS provided by NEC that has been adopted by the GENI project in United States. However, several core extensions needed to be made, due to hardware incompatibilities. The overall extensions have been wrapped up as an OMF Aggregate manager service.

Configuration of BS parameters takes place through the web-based API that resides on the WiMAX-RF server .The user can use the wget command to get or set the required parameters.

As soon as the parameters are set, the experimenter can submit a new Experiment Definition (ED) written in OMF Experiment Description Language and submit it to OMF's EC using a simple command:

omf exec "name of ED"

The wget command that is issued by the experimenter has to be written in a specific way, depending on the parameter that he/she wants to alter.

All WiMAX-RF services are categorized in the following:

•Services to configure the Base Station

•Services to configure the Mobile Clients (Subscriber Stations)

Each one of the two categories has a list of services available to the experimenter. A description of all WiMAX-RF service is available to the user through the web-base interface, or through the wget command:

wget -qO- http://wimaxrf:5054/wimaxrf/

You can pipe the output of the wget command to the xml_pp application for a user readable XML output.

wget -qO- http://wimaxrf:5054/wimaxrf/ | xml_pp

The bs/arq, bs/harq, bs/security, bs/wireless, bs/zone services show to the experimenter a short description regarding the functionality of the service they provide, permitted values and the name of parameters which the user can configure in order to be provisioned by the BS .

The services bs/get, bs/set are two functions which require the name of parameter and value in order to configure the Base station. For example, if the user wants to change the transmission power of Base station to 40 dBm then he/she has change the parameter "txpower" of the wireless service. The command to change the power used by the BS is:

wget -qO- "http://wimaxrf:5054/wimaxrf/bs/set?txpower=40"

If the user wants to query the BS of its current transmission power he/she has to replace the bs/set with bs/get and without indicating any value.

wget -qO- "http://wimaxrf:5054/wimaxrf/bs/get?txpower"

We further explain the values and the parameters of the services which the user has to set in order to appropriately setup his/her experiment.

ARQ service

dlarq

Set its value to 1 to enable the operation of downlink ARQ. Default: 0 disable.

ularq

Set its value to 1 to enable the operation of uplink ARQ. Default: 0 disable.

HARQ service

dlharq1,dlharq2

Both parameters must be set to value 1 to enable the operation of downlink HARQ. Default: 0

ulharq1,ulharq2

Both parameters must be set to value 1 to enable the operation of uplink HARQ. Default: 0

WIRELESS service

freq

Base Station's operating frequency in KHz. Only one frequency range is supported (2530000-2630000 KHz). The default value is 2590000 KHz.

bwmode

Bandwidth mode: 4 = 10 MHz, 3 = 7 MHz, 2 = 5 MHz, 1 = 3.5 MHz. Default value is 4 = 10 MHz. When changing bwmode, you also need to change the corresponding affected settings such as fftsize, dlulratio.

dlulratio

Ratio of uplink and downlink OFDM symbols in the frame. The frame duration is dependent on the bandwidth.

10 MHz (bwmode=4)	7 MHz (bwmode=3)	5 MHz (bwmode=2)	3.5 MHz (bwmode=1)
74 ->DL=35, UL=12	73 ->DL=24, UL=09	74 ->DL=35, UL=12	73 ->DL=24, UL=09
68 ->DL=32, UL=15	64 ->DL=21, UL=12	68 ->DL=32, UL=15	64 ->DL=21, UL=12
62 ->DL=29, UL=18	55 ->DL=18, UL=15	62 ->DL=29, UL=18	55 ->DL=18, UL=15
55 ->DL=26, UL=21		55 ->DL=26, UL=21	

fftsize

Scaling of the Fast Fourier Transform (FFT) to the channel bandwidth in order to keep the carrier spacing constant across different channel bandwidths (3.5 MHz, 5 MHz, 7 MHz and 10 MHz). Constant carrier spacing results in higher spectrum efficiency in wide channels, and a cost reduction in narrow channels. Also known as scalable OFDMA (SOFDMA). FFT subcarrier numbers are 512 and 1024.

action	10 MHz	7 MHz	5 MHz	3.5 MHz
set	fc->fft=1024	fc->fft=1024	a8 ->fft=512	a8 ->fft=512
get	256 <-fft=1024	256 <-fft=1024	168 <-fft=512	168 <-fft=512

Finally the services *bs/default, bs/info, bs/inservice, bs/maintenance, bs/restart, bs/status* were created to support specific actions such as restart the0 Base Station, get management information or reset the base station to its default parameters . For example, if a user wants to set the BS from the service state to the maintenance state, he/she has to run the command:

wget -q0- "http://wimaxrf:5054/wimaxrf/bs/maintenance"

WiMAX-RF AM Mobile Stations services reference

The service mobile client shows a short description of the permitted values and the names of the parameters which the user can configure and provision to a mobile client. The same service has a parameter with the name "type" in order to set or get the values of all other parameters. For example if the user want to change the control mode of a subscribed station with a MAC address "*00:1e:42:02:1c:4a*" to a non-registered mode (not connected to the BS) then he/she has to issue the following command:

wget -q0-
"http://wimaxrf:5054/wimaxrf/mobileclient?type=set&macadd=00:1e:42:02:1c:4a&param=controlmode&value=1 "

In order to get the current control mode of the same Subscriber Station, the experimenter has to, likewise the base station services, use the get command without any arguments:

wget -q0-
"http://wimaxrf:5054/wimaxrf/mobileclient?type=get&macadd=00:1e:42:02:1c:4a&param=controlmode&value="

We further explain the values and the parameters of the services which the user has to set in order to appropriately setup his/her experiment.

MOBILECLIENT service

controlmode

With this parameter the user can set to a specific mobile station (MS) the control mode which he/she wants to the MS to operate. The default mode is "Standalone Authentication and MIB Provisioning".

Control Mode	controlmode
No Registration	1
ASN-GW	2
Stanalone	3

ASN-GW Authentication and MIB Provisioning,	4
Standalone Authentication and MIB Provisioning Integer	5

dlmodulation1,dlmodulation2

Parameter that is used to specify downlink adaptive modulation profile. Adaptive modulation is used to adjust signal modulation scheme and FEC coding depending on the RF link's SNR and is defined in the form of list of allowed modulation and coding scheme (MCS) elements. The profile has 9 elements which are specified with the following service:

Modulation/Coding	dlmodulation1	dlmodulation2
QPSK (CTC) 1/2	0	1
QPSK (CTC) 3/4	1	1
16-QAM (CTC) 1/2	2	1
16-QAM (CTC) 3/2	3	1
64-QAM (CTC) 1/2	4	1
64-QAM (CTC) 2/3	5	1
64-QAM (CTC) 3/4	6	1
64-QAM (CTC) 5/6	7	1
dynamic modulation	255	0

When the user wants to change the modulation schema, he/she need to change both dlmodulation1 dlmodulation2 to the appropriate values. Default value for the MCS profile is set to Dynamic. Adaptive modulation is used to adjust signal modulation scheme and FEC coding depending on the RF link's SNR.

ulmodulation1, ulmodulation2

Uplink MCS used in each burst profile (DCD message).

Modulation/Coding	dlmodulation1	dlmodulation2
QPSK (CTC) 1/2	1	1
QPSK (CTC) 3/4	2	1
16-QAM (CTC) 1/2	3	1
16-QAM (CTC) 3/2	4	1
64-QAM (CTC) 1/2	5	1
64-QAM (CTC) 2/3	6	1
64-QAM (CTC) 3/4	7	1
64-QAM (CTC) 5/6	8	1
dynamic modulation	255	0

Likewise the service for the downlink profile, the user has to set both parameters.

Finally the services *mobileclient/delete*, *mobileclient/monitor, mobileclient/provision, mobileclient/reregister* were created to support specific assumptions such as re-register a

mobile client or get monitoring information on the MS. For example, if a user wants to monitor a re-register the mobile client with MAC address "*00:1e:42:02:1c:4a*" then he/she has run the command:

```
wget -qO-
 "http://wimaxrf:5054/wimaxrf/mobileclient/monitor?macadd=00:1e:42:02:1c:4a"
```

7 OpenFlow related functionality

Since one of the main directions of FIBRE is related to OpenFlow related experimentation, the provision of relevant functionalities by OMF was identified as a direction of special interest in Task 3.4. For this reason, following a period of remote interaction, a researcher from UTH visited NICTA for a two-month period, in order to accelerate progress. After the visit, the interaction continued remotely. In this section we describe the results of this work.

OMF6 assumes that everything is a resource. To interact with such a resource, other entities (e.g. an experimenter) communicate with a 'Resource Proxy', which is the software interface to that given resource. So, the first step was the implementation of a resource proxy that handles a resource capable of creating virtual switches. This resource proxy is named "virtual switch factory" and produces other resources of type "virtual switch".

In the subsections below we describe two simple scenarios that demonstrate the use of these resource proxies. These are two of the most common cases that require special management and configuration for clear and transparent slicing of the OF switches. The user is able to get access to a virtual switch that includes a subset of the OF switches resources (ports, flow entries, etc). This subset is sufficient to run his/her experiments, while the other virtual switches cannot interfere with his/her virtual switch.

7.1 Interconnected OF switches

In a 'slice'-enabled testbed, experimenters should be able to reserve some nodes and run experiments using exclusively these nodes. In case that all of them are connected to a single OF switch, the 'slicing' of this switch is a trivial case. The user takes access to a 'virtual switch', which includes only the ports that his/her reserved nodes are connected to. However, there are cases that the slicing is not trivial. For example, if half of his/her reserved nodes are connected to an OF switch, and the other half is connected to another one, then the correspondent 'virtual switch' will need access to the ports that interconnect the two switches. Simultaneously, it is possible that other 'virtual switches' will need access to the same couple of ports to interconnect reserved nodes from other experimenters.

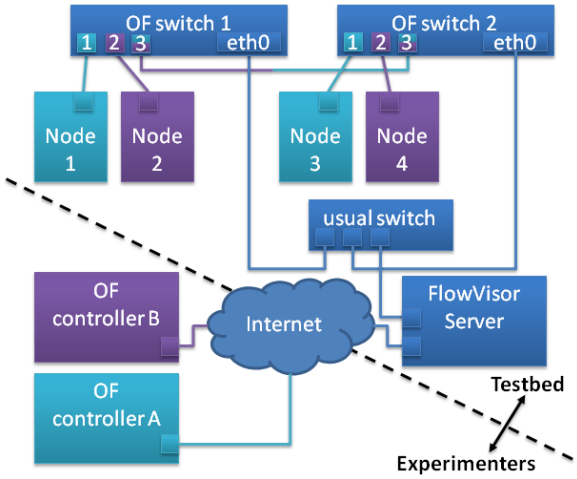

Figure 6: Interconnection of physical OF switches with trunk link

This scenario is illustrated in Fig.6, where there are two interconnected OF switches, and each of them is connected with two nodes. Experimenter A requested and reserved nodes 1 and 3, while experimenter B reserved nodes 2 and 4. Both of them requested for virtual switches that interconnect only their nodes and behave as their provided OF controller describe. Therefore, only virtual switch A should be capable to receive or forward packets to port 1 of the physical switches, and the same with virtual switch B and port 2. Up to this point, everything is clear and implemented in a straightforward way by assigning the appropriate ports to the corresponding virtual switches. The problem arises with the port 3 of both switches, which should be used by both virtual switches. In this case a more specific flow entry should be assigned to each virtual switch, which is specified from the port 3 and maybe the source MAC of the packet. In our example:

Virtual Switch A should handle the following flow entries:

- packets received from port 1 of OF switch 1
- packets received from port 1 of OF switch 2
- packets received from port 3 of OF switch 1 and source MAC is the MAC of Node 3
- packets received from port 3 of OF switch 2 and source MAC is the MAC of Node 1

Virtual Switch B should handle the following flow entries:

- packets received from port 2 of OF switch 1
- packets received from port 2 of OF switch 2
- packets received from port 3 of OF switch 1 and source MAC is the MAC of Node 4
- packets received from port 3 of OF switch 2 and source MAC is the MAC of Node 2

Unfortunately, this solution is not clear and requires a large amount of configuration commands for each added node. In order to build a clear solution, easily configurable, we decided to add flow entries to the virtual switches that are specified from the source MAC addresses and not the ports. So the correspondent sets of flow entries are the following:

Virtual Switch A flow entries:

- packets that have as source MAC, the MAC of Node 1
- packets that have as source MAC, the MAC of Node 3

Virtual Switch B flow entries:

- packets that have as source MAC, the MAC of Node 2
- packets that have as source MAC, the MAC of Node 4

The only problem with this solution is that is not sustainable under the misbehavior of MAC spoofing.

7.2 Gateway

Another common case that requires special attention is the scenario that includes two virtual switches which have to share a common link to a gateway. We assume that the traffic of each virtual switch destined to the Internet, uses a unique public IP as source IP address. So in this example:

Virtual Switch A should include the following flow entries:

- packets that have as source IP, the public IP reserved for the traffic of this virtual switch

Virtual Switch B should include the following flow entries:

- packets that have as source IP, the public IP reserved for the traffic of this virtual switch

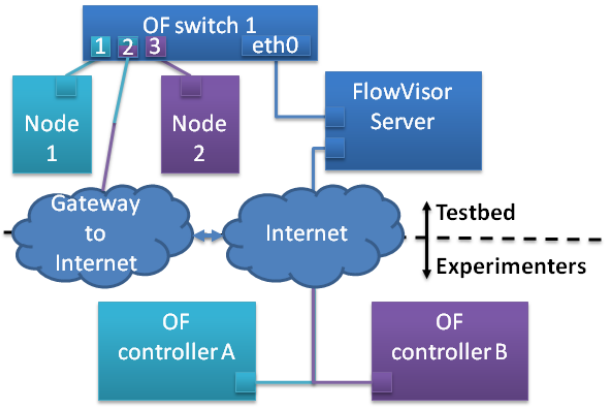

Figure 7: A shared link to a gateway to the Internet

7.3 Installation and Usage

The RCs for the FlowVisor and the OpenvSwitch are available as a Ruby Gem. Installing it is as easy as

```
$ gem install omf_rc_openflow
```

- FlowVisor creates OpenFlow Slices (slicing the flow space into separate pieces), so the corresponding RC is the OpenFlow_Slice_Factory.
- OpenvSwitch creates Virtual OpenFlow Switches on top of a Linux machine using the machine's interfaces, so the corresponding RC is the Virtual_OpenFlow_Switch_Factory.

To use them, in a Linux machine that runs FlowVisor or OpenvSwitch software, execute:

$ omf_rc_openflow_slice_factory -u xmpp://user:password@domain -i topic

or

$ omf_rc_virtual_openflow_slice_factory -u xmpp://user:password@domain -i topic

to control the FlowVisor or OpenvSwitch resource in a OMF6 Experiment Controller (EC).

8 OMF Broker

In OMF 6 an entity called “Broker” is responsible for advertising the testbed resources to the experimenters and also responsible for the reservation and the provisioning of them. Broker will be the layer of OMF which stands between the experimenter and the testbed.

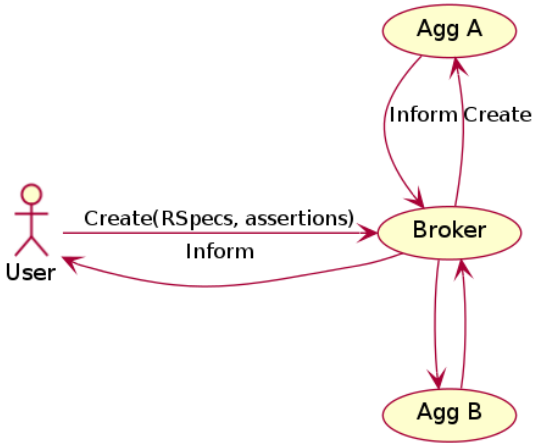

Figure 8: Experimenter and Broker interaction

Whenever an experimenter wants some resources, he sends a create message to the Broker which in turn serves the user’s request after it has successfully authenticated him. The Broker is responsible for contacting the testbed resources and provisioning them to the experimenter.

Last but not least, the Broker is the entry point for the reservation of resources too. It keeps all the necessary information regarding the reservations, in order to be able to respond to the experimenters’ requests about reserving resources.

The reservation of resources was modelled as an object under the name “lease” which itself is being perceived as a **resource**. More specifically “lease” contains all the necessary information about the reservation start and end times, the related resources attached to it and the slice which owns this lease. It can be clearly described by the OMF information model.

All the above functionality, which is concentrated in the Broker entity, is being distributed in different entities within the Broker as can be seen below.

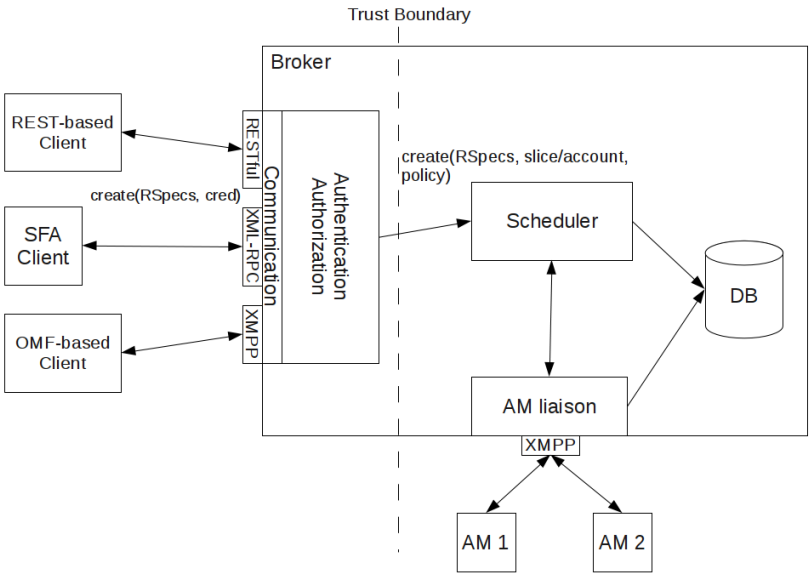

Figure 9: Broker architecture

The entities that can be seen in the Broker's architecture figure are responsible for the functionalities described so far. More specifically, the Authentication/Authorization entity is responsible for security and authentication policies in the context of an OMF testbed. The scheduler is responsible for keeping track of the leases and informing the AM Liaison for provisioning the resources. The multiple communication interfaces enable diversity in client applications which can contact the testbed with various ways. The purpose of the XML-RPC interface is to serve as an SFA interface which couples nicely with the federation requirements in WP4. The XMPP interface exposes an FRCP (Federated Resource Control Protocol) API, being able to handle all the standardized messages of the protocol. The Scheduler functionality of the Broker can be reached through the different APIs enabling the experimenters to reserve resources through an SFA client like MySlice, or through a native testbed web tool (REST API), or even with an FRCP tool like the OMF Experiment Controller.

The development of the OMF Broker was a joint effort by UTH and NICTA. It is currently installed in NITOS testbed, and in the process of being released as a Ruby Gem. Essentially, it will allow all OMF-based testbeds to share the same mechanism for resource management, replacing custom solutions (testbed-specific SFA drivers).

9 Support for Software Radio Resources

One of UTH's activities related to T3.4 has been the development of OML support for measurements captured through USRPs [7], the software radio platforms of NITOS. The USRPs in NITOS have mainly been used for obtaining accurate spectrum measurements, capturing the activity at specified frequencies with great precision. However, they can be used for any experiment requiring reconfiguration of radio parameters, thus being very powerful tools for the experimenter.

Specifically, UTH have worked with the IRIS framework [8], a highly reconfigurable software framework to setup and control USRPs. IRIS is used for constructing complex radio structures and highly reconfigurable radio networks.

A library has been created, which uses a well-defined yet simple API, which provides the experimenter with the capability to add new OML Measurement Points in the IRIS framework, apart from those that UTH have added.

As a proof of concept, UTH provide a precompiled Linux image at the NITOS testbed, that contains the IRIS framework and an OML enabled version of it. Since IRIS is a framework and the supported configurations are too many, the OML enabled version captures and sends to the OML server some measurements that are common for all different configurations, the name and number of different components (components, controllers, engines) used in an experiment and the time that each one takes to finish. The operations supported by current IRIS version are depicted in the figures below.

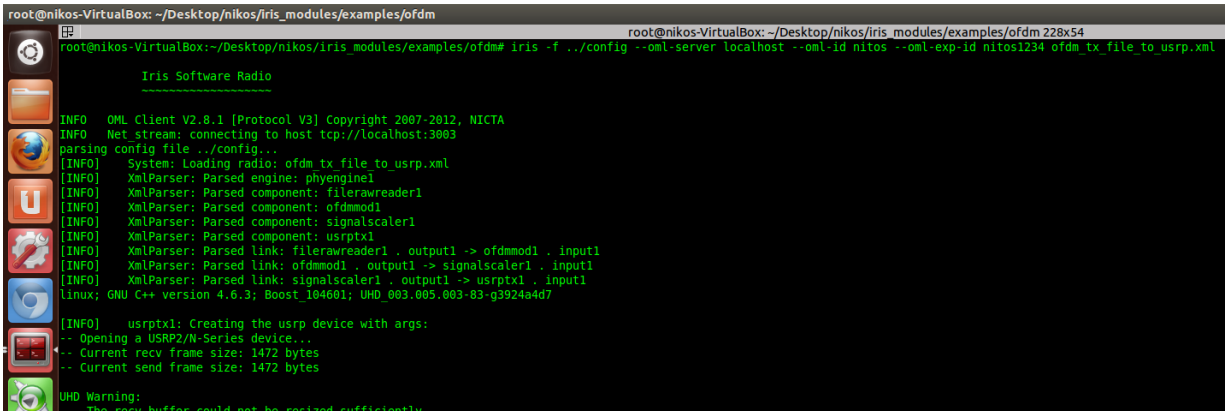

Figure 10: Calling IRIS with OML parameters

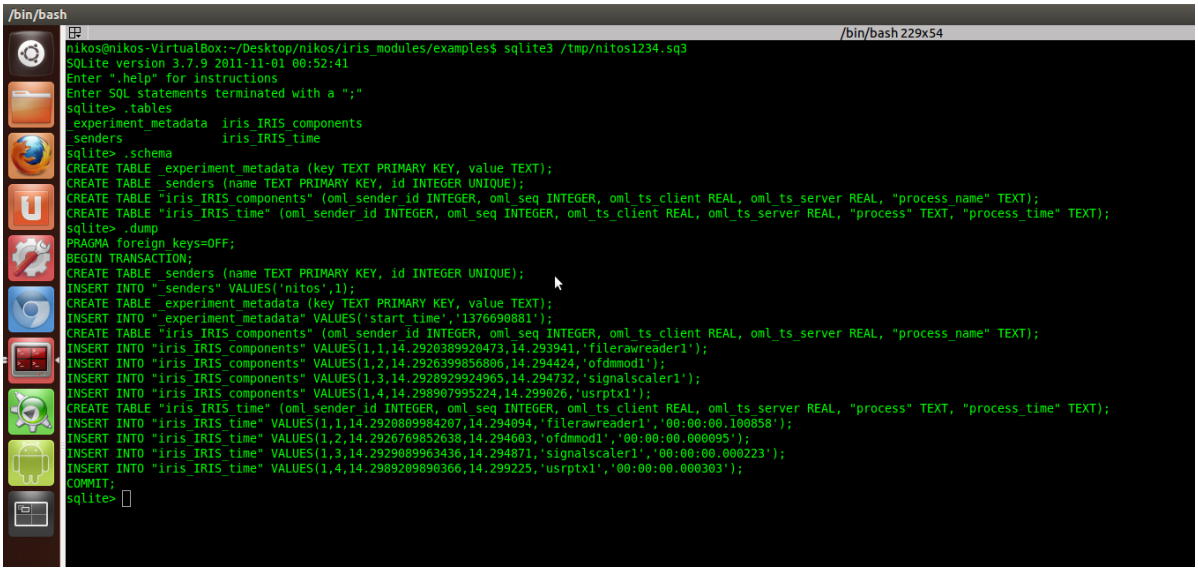

Figure 11: The OML database created by IRIS

"*This work makes use of results produced by the FIBRE project, co-funded by the Brazilian Council* for Scientific and Technological Development (CNPq) and by the *European Commission within its Seventh Framework Programme.*"

END OF DOCUMENT