Project Acronym	**Fed4FIRE**
Project Title	**Federation for FIRE**
Instrument	**Large scale integrating project (IP)**
Call identifier	**FP7-ICT-2011-8**
Project number	**318389**
Project website	**www.fed4fire.eu**

D3.6 - Report on third cycle developments of the infrastructure community

Work package	WP3
Tasks	T3.2, T3.3, T3.4 ,T3.5
Due date	31/01/2016
Submission date	
Deliverable lead	Ciro Scognamiglio (UPMC)
Version	
Authors	Ciro Scognamiglio (UPMC) Brecht Vermeulen (iMinds) Carlos Bermudo (i2Cat) Ali Hammad (UnivBris) Aris Dadoukis (UTH) Donatos Stavropoulos (UTH) Chrysa Papagianni (NTUA) Vassilis Karyotis (NTUA) Symeon Papavassiliou (NTUA) ChangWoo Kim (NIA) Thierry Rakotoarivelo (NICTA)

	Leandro Navarro (UPC) Almudena Díaz Zayas (UMA) Pedro Merino Gomez (UMA) David Larrabeiti (UC3M) Raquel Aparicio Morenilla (UC3M) Johann M. Marquez-Barja (TCD) Nicholas Kaminski (TCD) Paolo Di Francesco (TCD) José Luis García Dorado (UAM) Adnan Bekan (JSI) Igor Ozimek (JSI) Tomaž Šolc (JSI) Łukasz Łopatowski (PSNC) Michał Giertych (PSNC) Bartosz Belter (PSNC)
Reviewers	Julien Lefeuvre (INRIA) Peter Van Daele (iMinds)

Abstract	This document describes the work done in WP3 to support cycle 2 of Fed4FIRE.
Keywords	Services, applications, experiment control, monitoring

Nature of the deliverable	R	Report	X
	P	Prototype	
	D	Demonstrator	
	O	Other	
Dissemination level	PU	Public	X
	PP	Restricted to other programme participants (including the Commission)	
	RE	Restricted to a group specified by the consortium (including the Commision)	
	CO	Confidential, only for members of the consortium (including the Commission)	

Disclaimer

The information, documentation and figures available in this deliverable, is written by the Fed4FIRE (Federation for FIRE) – *project consortium under EC co-financing contract FP7-ICT-318389 and does not necessarily reflect the views of the European Commission. The European Commission is not liable for any use that may be made of the information contained herein.*

Executive Summary

The Fed4FIRE architecture addresses the needs of the experimenter communities in three key areas: experiment lifecycle, measurement and monitoring, and trustworthiness. The architecture is evolving as time progresses and the requirements of the federation (including its sustainability) become better understood. The architecture is being implemented in a series of cycles. This document describes the work done to implement the features of cycle 1 in the architecture-focused test facilities PlanetLab Europe, w-iLab-t, VirtualWall, OFELIA, Nitos, Netmode, Norbit and Koren.

Ideally, all test facilities would have been able to implement all features of cycle 2 of the architecture. However the technical level of difficulty associated with implementing some features differed between test facilities, and therefore a prioritised subset of the ideal cycle 2 features became the target. Fed4FIRE is an evolutionary project and this process of prioritisation of development is ongoing, driven by the principle of offering benefits and value to experimenters. In some cases this has involved temporary work-arounds to ensure that critical functionality is available. The table below gives the status of each test facility at the end of cycle 2.

	PLE	VirtualWall	w-iLab.t	OFELIA		NITOS	NETMODE	KOREN	NORBIT	PerformLTE	C-Lab	UltraFlow	LOG-a-TEC	10GTrace-Tester	FAVORITE	PL-LAB
				i2cat	Bristol											
ASSOCIATED																
LIGHT FEDERATION																
ADVANCED FEDERATION																
BASIC FUNCTIONALITY																
Documentation																
Testbed description																
Basic Tutorial																
Policies																
Basic experiment w/o extra approval																
Basic support via google group																
Facility Monitoring																
First Level Support																
Testbed interface for experimenters																
SFA AM approach																
GetVersion support																
ListResources support																
Cover full cycle																
Provide AM URLs to testbed directory																
Docs: RSpec models description																
Speaks-For credential support (AM)																
Own Testbed interface approach																
Deploy own testbed service																
PKCS#12 F4F credentials support																
Include API in service directory																

Docs: API usage																
YourEPM integration (API RAML desc.)																
Speaks-For credential support (service)																
IPv4/IPv6 Connectivity																
SFA AM																
Testbed SSL service endpoint																
Nodes / Resources																
ADDITIONAL ENHANCEMENTS																
Infra. mon. (federation services)																
Infra. mon. (per experiment, AM/OML)																
Advanced reservation			in progress													
SLA		in progress	in progress				cycle 2 SLA									
Reputation																
Perrmanent storage																
Experiment control																
images with OMF																cycle 3
AMQP server																cycle 3

PDP																cycle 3
Layer 2 connectivity																
VLAN stitching																
Tunnels (in RSpec)																
OK																
In progress																
Not currently planned																
Not applicable																
On the roadmap / Post Cycle 3																

Acronyms and Abbreviations

AM	Aggregate Manager
API	Application Programming Interface
CA	Certificate Authority
EC	Experiment Controller
FLS	First Level Support
FRCP	Federated Resource Control Protocol
GENI	Global Environment for Network Innovations
NEPI	Network Experiment Programming Interface
OCCI	Open Cloud Computing Interface
OMF	Control and Management Framework
OML	OML Measurement Library
OMSP	OML Measurement Stream Protocol
QoS	Quality of Service
PDP	Policy Decision Point
RSpec	Resource Specification
SFA	Slice Federation Architecture
SNAA	Sensor Network Authentication and Authorization
VM	Virtual Machine
WSN	Wireless Sensor Network
XMPP	Extensible Messaging and Presence Protocol

Table of Contents

1 Introduction

This document describes the work done in workpackage 3 of the Fed4FIRE project to implement the features agreed for cycle 2 of the Fed4FIRE architecture in the test facilities PlanetLab Europe, w-iLab-t, VirtualWall, Ofelia, Nitos, Netmode, Norbit, Koren, PerformLTE, C-Lab, UltraFlow and the new testbeds LOG-aTEC, 10GTradeTester, FAVORITE and PL-LAB that joined Fed4Fire recently before the end of Cycle 2. The period of this report is up to month 28 of the project.

The architectural features are described in deliverables D2.4 [1], D5.2 [2], D6.2 [3] and D7.2 [5]. The features described on these documents represent the ‘ideal’ state of cycle 2 of Fed4FIRE. However it was recognised within the project that the technical level of difficulty associated with implementing some features differed between test facilities, and therefore a prioritised subset of the ideal cycle 2 features (described in milestone MS3.4 [6]) became the target. Fed4FIRE is an evolutionary project and the process of prioritisation of development is ongoing, driven by the principle of offering benefits and value to experimenters. In some cases this has involved temporary work-arounds to ensure that critical functionality is available.

The document is structured as follows:

- This introduction reviews briefly the functionality proposed in cycle 2 of the project
- Subsequent sections detail the progress in each test facility in WP3 for the three architectural areas of experiment control, experiment monitoring and trust and security.
- The progress towards cycle 3 goals is outlined for each test facility.

The Fed4FIRE architecture for cycle 2 of the project is described in deliverable D2.4 “Second federation architecture”. The architecture addresses the needs of the experimenter communities in three key areas: experiment lifecycle management, measurement and monitoring, and trustworthiness.

1.1 Experiment lifecycle management

Experiment lifecycle management covers the ability for experimenters to discover and provision resources, and to control running experiments across federated test facilities. This is implemented in the architecture by centrally provided features such as a portal, and by features implemented within each test facility, and is described in [2].

The decision was taken to adopt Slice Federation Architecture (SFA) as the standard for resource description and discovery, and the Federated Resource Control Protocol FRCP standards for experiment control. Consideration was given to the relative ease with which these protocols could be adopted by a facility, and hence a uniform level of acceptance was not specified in cycle 1 across all test facilities. For example many test facilities already used SFA and hence had little or no work to do to be compliant with the architecture. Others had considerable work to do to match the concepts of their existing resource management framework to SFA and implement changes, and hence were not expected to become SFA compliant in cycle 1.

Fed4FIRE has adopted Control and Management Framework (OMF) [7] for experiment control.

1.2 Measurement and monitoring

Measurement and monitoring has three purposes in Fed4FIRE:

- To collect data on the health of the underlying facilities so that they can be kept up and running. This is called **facility monitoring** in Fed4FIRE
- To collect performance statistics on the resources used by an experiment, which are of interest to the experimenter. This is called **infrastructure monitoring** in Fed4FIRE
- To collect experiment-defined data, also for use by the experimenter. This is called **experiment monitoring** in Fed4FIRE.

The approach to cycle 2 is described [3]; facility monitoring was the priority for cycle 2. The decision was taken to adopt OML as the standard for providing monitoring for all three types of information stream.

1.3 Trust and security

A well-defined trust and security framework is essential for a federated network to operate. The specifications for cycle 2 are described in [4]. In cycle 2 the focus was on a basic security framework for resource discovery and provisioning, following the decision to adopt SFA.

The focus on cycle 2 is user management and authentication. Users can obtain X.509 certificates from Fed4FIRE identity providers. The certificates can be presented to any test facility, which makes a decision as to whether it trusts the identity provider that signed the certificate. A test facility does this by reference to directory of root certificates from Fed4FIRE ID providers. The trust decisions are made at the test facility level.

2 PlanetLab Europe

2.1 Facility description

PlanetLab Europe is the European portion of the publicly available PlanetLab testbed and is a part of the OneLab experimental facility.

Established in 2002 PlanetLab is a global network of computers available as a testbed for computer networking and distributed systems research. As of December 2011, PlanetLab was composed of 1024 nodes at 530 sites worldwide. Each research project runs a "slice" that gives experimenters access to a virtual machine on each node attached to that slice. See the PLE statistics page for more details: http://onelab.eu/index.php/testbeds/onelab-testbeds/planetlab-europe/ple-statistics.html.

Accounts are available to persons affiliated with corporations and universities that host PlanetLab nodes. Those who join PlanetLab Europe have access to the entire system. They also participate in the initiatives built around PlanetLab in Europe.

PlanetLab members actively participate in developing tools for the greater good of the community, and as a result each user has a wide choice of tools to use in order to complete regular slice maintenance tasks.

There are a number of free, public services that have been deployed on PlanetLab, including CoDeeN, the Coral Content Distribution Network, and Open DHT.

PlanetLab Europe operates under the direction of Timur Friedman of UPMC Sorbonne Universités, working in collaboration with the Institut National de Recherche en Informatique et en Automatique (INRIA).

PlanetLab Europe and OneLab

PlanetLab Europe is a key testbed within the OneLab experimental facility for Future Internet technologies. OneLab is extending PlanetLab Europe into new environments, beyond the classic wired internet. OneLab is deepening PlanetLab Europe by incorporating new monitoring tools. OneLab is federating PlanetLab Europe, both with other PlanetLabs worldwide and with other types of testbeds.

The OneLab experimental facility is funded by several different research projects, including OpenLab and NOVI within the European Commission's FIRE Unit, along with national and international projects F-Lab, FIT and FIBRE.

2.2 Basic Functionality

2.2.1 Documentation

PlanetLab Europe provided testbed level documentation. Additionally tutorials, documentation and reference manuals for users and developers are available on the PlanetLab Europe web site.

2.2.2 Identity Provider

PlanetLab Europe supports X.509 certificate identity authentication and a certificate-based user identity management has been implemented and integrated.

2.2.3 Certificate Directory

PlanetLab Europe provides a self-signed root certificate to the certificate directory and also fetches the other certificates to trust them.

2.2.4 PDP

PlanetLab Europe resources are controllable also via OMF, currently PLE supports OMFv6. But because the user has always access to the resource via SSH and has administrator privileges on the node, PLE wouldn’t benefit from an access control mechanism based on PDP.

2.2.5 Facility Monitoring (FLS)

PlanetLab Europe uses a separate tool, but tailored for PlanetLab deployments, named MyOps, which allows implementing escalation policies. For example, if one site has a node down, first, messages are sent to the technical contact, then to the PI, and after a while the site loses some of its capabilities (fewer slices). All this workflow is entirely automated under MyOps, which also provides raw data on the status of nodes – see http://planetlab.eu/monitor/site.

Nagios is also used as an infrastructure monitoring and alerting solution. Nagios provides information on the status of the infrastructure and generic services like DNS, SSH or HTTP. Services specific to PlanetLab testbed are also monitored: SFA AM, Registry, OMF etc.

Both sources have been used to provide monitoring information to the Fed4Fire FLS service by the use of OML streams.

2.2.6 Experiment Monitoring

Slicestat data can be queried directly on a node in real time; additionally a service collects those information in a central server with a time resolution of 15 minutes. The resulting data can be queried either directly (data is stored on a PostgreSQL database) or through a JSON based RESTful interface that can be used to easily retrieve statistics on usage and load averages.

Experimenters can use those services to monitor the trend (in terms of resource usage) of their running experiment.

The metrics are collected internally on a time-based database that can be easily queried for building statistics and graphs. The measurements can also be sent with an OML stream to the TU-Berlin OML server instance provided for the federation.

2.2.7 Testbed interface for experimenters

In Fed4FIRE the mechanism chosen for implementing resource discovery, requirements, reservation and provisioning is the Slice Federation Architecture (SFA) standard (specifically GENI AM APi v3), together with ontology based resource specifications (Rspecs).

PlanetLab Europe has a native implementation of SFA that has been upgraded and enhanced during Cycle 3 developments.

2.2.7.1 SFA AM approach

SFA AM in PlanetLab Europe has been implemented natively, the distribution package is called SFAWrap.

2.2.7.2 Own Testbed interface approach

PlanetLab Europe also has a native API implementation based on an XMLRPC protocol. This API provides also functionalities for managing testbed resources, users and participants.

2.2.8 IPv4/IPv6 Connectivity

PlanetLab Europe resources are already available on the public Internet and are accessible via SSH with a public IPv4 address. IPv6 support is also available partially on the testbed. IPv6 support on the resourced depends on the site member PlanetLab Europe, in general if the site hosting the nodes (resources) has IPv6 deployed in its infrastructure then the nodes will be accessible also with an IPv6 address. At the time of writing roughly 80 resources have an IPv6 address.

2.3 Additional Enhancements

2.3.1 Infrastructure monitoring

PlanetLab Europe uses to aggregate measurement sources such as TDMI and others. One difficulty encountered by the PlanetLab Europe operations team was that the potentially very useful data gathered by MyOps are not easily accessible through an API or other query-based techniques, and so MyOps does not lend itself to the creation of a gateway so as to be made available through TopHat. There clearly was a need for both: (1) aggregating data about nodes from various places (MyOps being one, CoMon slicestat being another one, and we found quite a few other sources of potentially high interest), and (2) providing a decent querying interface to these data. In a first step, we leveraged the internal "tags" mechanism right in the MyPLC DB to extend it with such external data. In a federated world, and in particular with respect to MySlice, it might make sense to design a separate tool for hosting this aggregated data.

Regarding monitoring of experimentation metrics, OML was considered to be a suitable candidate to deploy, and, thus, has been deployed in PlanetLab Europe.

We run internally an instance of an OML database that collects metrics from resources and monitoring services and we send this information to the OML instance maintained by TU-Berlin and used in the federation.

2.3.2 Reputation

Infrastructure monitoring information described above, provided as an OML stream, is used for the reputation broker.

2.3.3 Experiment Control

In PlanetLab Europe the means of interacting with one's slivers is through ssh. In addition, a slice can optionally be created as "OMF-Friendly" in which case it is possible to control its related slivers through an OMF Experiment Controller. PLE OMF support has been upgraded to the latest version 6.0.

A tool has also been developed, NEPI (http://nepi.inria.fr) that supports experiment control in PlanetLab Europe.

2.3.4 Layer 2 connectivity

GRE based tunnels have also been established with some of the testbeds (NITOS, NETMODE), those tunnels can be established on the fly between one or more PLE resources and a remote testbeds and would allow direct interconnections between the reserved resources. Implementation of this service is in a finalization phase.

3 Virtual Wall

3.1 Facility description

See also http://doc.ilabt.iminds.be/ilabt-documentation/virtualwallfacility.html

The Virtual Wall is hosted at and operated by iMinds iLab.t. It actually exists out of 2 testbeds:

- Virtual Wall 1 contains 190 nodes (pcgen1 and pcgen2 nodes)
- Virtual Wall 2 contains 134 nodes (pcgen3 and pcgen4 nodes)

The hardware can be used as bare metal hardware (operating system running directly on the machine) or virtualized through openVZ containers or XEN virtualization. XEN Virtualization comes into two flavours: using shared nodes (non-exclusive) where VMs are running on physical hosts which are shared with other experimenters, or using physical nodes (exclusive) which are exclusively used by your experiment. You have full control on all XEN parameters as you have root access to the DOM0.

Network impairment (delay, packet loss, bandwidth limitation) is possible on links between nodes and is implemented with software impairment.

Multiple operating systems are supported, e.g. Linux (Ubuntu, Centos, Fedora), FreeBSD, Windows 7.

Some of the nodes are connected to an OpenFlow switch to be able to do OpenFlow experiments in a combination of servers, software OpenFlow switches and real OpenFlow switches.

You can check the number of free resources on each testbed at https://flsmonitor.fed4fire.eu.

The hardware of Virtual Wall 1 consists out of:

- 90x pcgen1 nodes: 2x Dual core AMD opteron 2212 (2GHz) CPU, 8GB RAM, 4x 80GB harddisk, 4-6 gigabit nics per node (20 nodes are wired to a display in the lab) (nodes n17x-xx)
- 100x pcgen2 nodes: 2x Quad core Intel E5520 (2.2GHz) CPU, 12GB RAM, 1x 160GB harddisk, 2-4 gigabit nics per node (nodes n14x-xx)

The hardware of Virtual Wall 2 consists out of:

- 100x pcgen3 nodes: 2x Hexacore Intel E5645 (2.4GHz) CPU, 24GB RAM, 1x 250GB harddisk, 1-5 gigabit nics per node (nodes n095-xx, n096-xx, n097-xx)
- 6x pcgen1 nodes with 6 gigabit nics
- 28x pcgen4 nodes: 2x 8core Intel E5-2650v2 (2.6GHz) CPU, 48GB RAM, 1x 250GB harddisk, gigabit and 10gigabit connections (still to connect) (nodes n091-xx)
- 2x GPU nodes (n095-22, n097-22): 1x 6core Intel E5645 (2.4GHz), 12GB RAM, 4x 1TB disk in RAID5, 4x Nvidia Geforce GTX 580 (1536MB). Use the image urn:publicid:IDN+wall2.ilabt.iminds.be+image+dred:gpuready if you want to have a pre-installed SDK in /root

The following pcgen4 machines are equiped with RAID adapters and fast disks and are used as shared nodes, providing XEN VMs. It is also possible to have a public IPv4 address on VMs on these hosts (besides the default IPv6 public IP address):

- n091-01
- n091-06
- n091-26

3.2 Basic Functionality

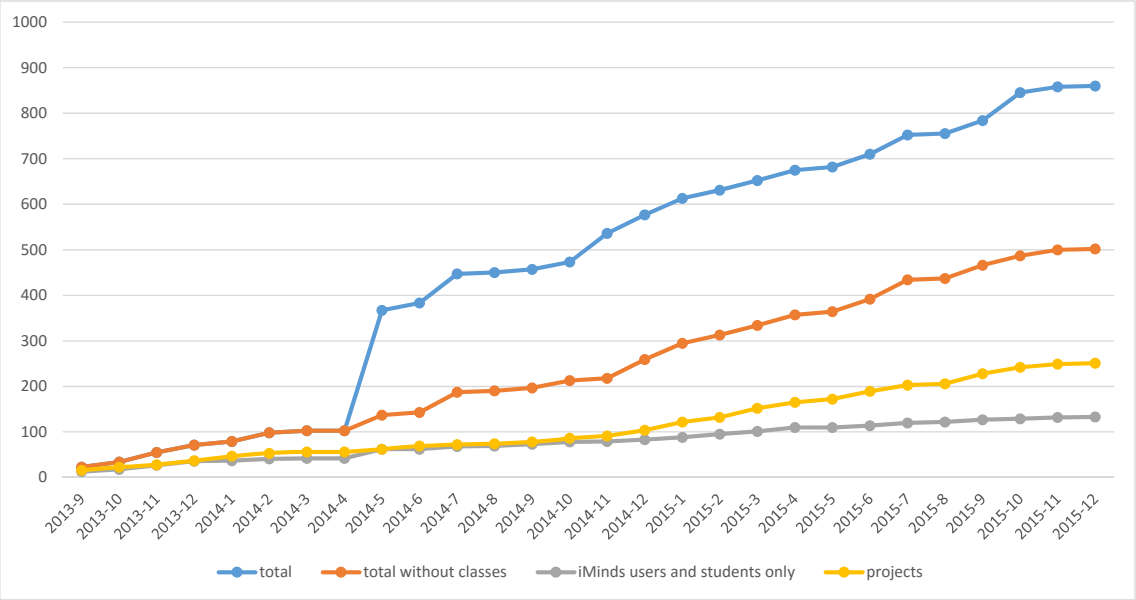
3.2.1 Documentation

3.2.2 Policies

3.2.3 User registration

iMinds has upgraded the user interface of the its identity provider, in a more user friendly user interface and now runs at https://authority.ilabt.iminds.be.

Statistics of user accounts can be seen below:



We created different colored lines to show the different types of accounts:

- In grey we have the iMinds' only users and students, which steadily increases (a little steeper increase then before Fed4FIRE)
- In orange, we have the sum of iMinds users and regular external users, this is a huge increase since Fed4FIRE
- In blue, this is the total number of accounts, including also specific accounts for classes (e.g. for Forge, for the Fed4FIRE-GENI summer school, etc).
- In yellow, we see also the number of projects that have been created. This gives an indication of the number of different experiment topics/experiments that are done on the facilities. A project exists out of multiple people and multiple slices.

3.2.4 PDP

A prototype of the PDP is installed at the Virtual Wall.

3.2.5 Facility Monitoring (FLS)

Available from cycle 1, now is done through AMv3.

3.2.6 Testbed interface for experimenters

3.2.6.1 SFA AM approach

The Virtual walls support AMv3, and this has now also become the default API through the jFed tool. All new features mentioned above are available through the API. The slice and member authority APIs (getting credentials, creating slices) have been upgraded to the common federation API v2

(http://groups.geni.net/geni/wiki/CommonFederationAPIv2) so they are now also in line with a common standard (such as the AM API for the provisioning and reservation of resources).

3.2.6.2 Own Testbed interface approach

3.2.7 IPv4/IPv6 Connectivity

All nodes are reachable through public IPv6 addresses. For XEN VMs a public IPv4 address can be requested through the API.

3.3 Additional Enhancements

3.3.1 Infrastructure monitoring

The sum of all switch bandwidth is reported to a central OML server and can be used for infrastructure monitoring and SLA verification.

3.3.2 Experiment Monitoring

Documentation for use of OML is available (see http://doc.fed4fire.eu/oml.html)

3.3.3 Advanced reservation

Not yet available.

3.3.4 SLA

The sum of all switch bandwidth is reported to a central OML server and can be used for infrastructure monitoring and SLA verification. Per sliver availability information is available.

3.3.5 Reputation

Per sliver availability information is available.

3.3.6 Permanent Storage

Possible through a central NFS server. A prototype of cloud storage has been developed, but is not yet available to the users. We hope to finish this before the end of the project though.

3.3.7 Experiment Control

We have moved from an XMPP server to an AMQP server to support OMF. The AMQP server is much more stable.

iMinds also deployed labwiki at http://labwiki.test.atlantis.ugent.be:4000

We have install scripts for OMFv6 and images where OMFv6 is pre-installed.

3.3.8 Layer 2 connectivity

The Virtual Walls have implemented layer 2 stitching and a lot of connectivity has been set up in cycle 2, see the figure below. Besides this native layer 2 connectivity, there is also support for GRE tunnels over the internet.

4 W-iLab.t

4.1 Facility description

w-iLab.t is hosted at and operated by iMinds iLab.t. The wireless testbed is located in an unmanned utility room (size: 66m x 22.5m). There is almost no external radio interference. At this location, hardware is hosted at 60 spots. Every spot is equipped with:

- 1 embedded PC with 2 Wi-Fi a/b/g/n interfaces and 1 IEEE 802.15.1 (Bluetooth) interface
- a custom iMinds-Rmoni sensor node with an IEEE 802.15.4 interface
- an "environment emulator" board (enabling unique features of the testbed including the triggering of repeatable digital or analog I/O events at the sensor nodes, real-time monitoring of the power consumption, and battery capacity emulation.

There are two additional possibilities:

- A number of cognitive radio platforms (including USRPs) as well as specialized spectrum scanning engines are available at this location. This enables state-of-the art research in the field of cognitive radio and cognitive networking.

20 mobile robots

4.2 Basic Functionality

4.2.1 Documentation

4.2.2 Policies

4.2.2.1 PDP

A prototype of the PDP is installed at the Virtual Wall. If this is in production, we will be able to use it also for w-iLab.t.

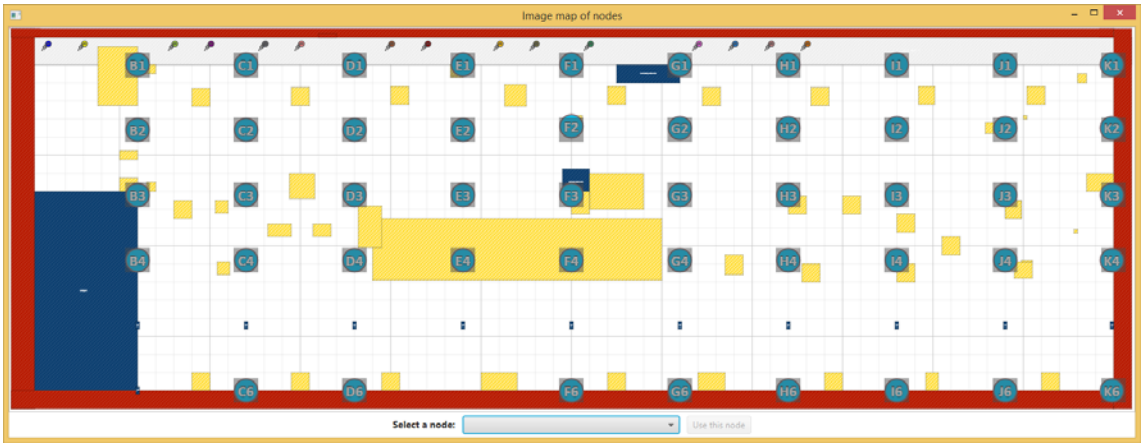
4.2.3 Facility Monitoring (FLS)

Available from cycle 1, now is done through AMv3.

4.2.4 Testbed interface for experimenters

4.2.4.1 SFA AM approach

w-iLab.t supports AMv3, and this has now also become the default API through the jFed tool. All new features mentioned above are available through the API. It is now also possible in jFed to select the nodes on a map:



Future reservation of nodes is possible with PKCS12 certificates (a derivative of X.509 certificates) out-of-band, but not yet through the AM API.

4.2.5 IPv4/IPv6 Connectivity

All nodes are reachable through public IPv6 addresses.

4.3 Additional Enhancements

4.3.1 Infrastructure monitoring

The experimenter can reserve measurement devices himself.

4.3.2 Experiment Monitoring

Documentation for use of OML is available (see http://doc.fed4fire.eu/oml.html)

4.3.3 Advanced reservation

4.3.4 SLA

Per sliver availability information is available.

4.3.5 Reputation

Per sliver availability information is available.

4.3.6 Permanent Storage

4.3.7 Experiment Control

We have moved from an XMPP server to an AMQP server to support OMF. The AMQP server is much more stable.

iMinds also deployed labwiki at http://labwiki.test.atlantis.ugent.be:4000

We have install scripts for OMFv6 and images where OMFv6 is pre-installed.

4.3.8 Layer 2 connectivity

Layer 2 connectivity only through GRE tunnels currently.

5 OFELIA

5.1 Facility description

OFELIA testbed aims to offer IaaS with virtualization (virtual machines) and SDN infrastructure (OpenFlow-enabled switches, including optical switches) resources.

Experimenters can request virtual machines hosted in many servers linked to SDN switches. Experimenters can also define a topology over the testbed network over which their traffic will run. Experimenter's traffic is isolated from other experiments and they can control the behavior of the switches over it.

OFELIA facility in Fed4FIRE project is composed of two islands of the many that formed part of the OFELIA project **[12]**: the i2CAT island and University of Bristol (UnivBris) island.

Both islands are similar as they comprise servers to host virtual machines to deploy experiments and OpenFlow-capable L2 switches. Although, each island has its peculiarities, so they will be described in detail by separate in the following sections.

5.1.1 i2CAT island

i2CAT OFELIA island offers the following resources to the experimenters as shown in Figure 5.1:

- 3 virtualization servers to deploy Debian 6 virtual machines to act as source or sink of traffic or to deploy any software required for the experiment.
- 5 OpenFlow-enabled switches (NEC IP8800) forming a mesh network. Experimenters can define subnets isolated from the others by VLANs and whose switches behavior is controlled by OpenFlow controllers deployed in the virtual machines.

i2CAT is linked to other Fed4FIRE islands such as the UnivBris OFELIA island or iMinds' VirtualWall.

OFELIA resources are behind a VPN without public IPs. UnivBris is providing a public gateway for these resources, so i2CAT and UnivBris islands are linked through a L2 VPN. This VPN allows access to i2CAT resources from a public IP. iMinds' VirtualWall is also connected to this VPN so it also offers direct access to i2CAT's resources. i2CAT-iMinds link offers 6 different VLANs, meaning that up to 6 different flows can be defined between both islands.

Monitoring of i2CAT island is done through UnivBris also. UnivBris is providing public addresses for i2CAT island's AMs under the VPN, so Fed4FIRE FLS monitoring system checks ping, GetVersion, Login Status, etc. through these public addresses. UnivBris redirects traffic incoming to these addresses to the private ones from i2CAT inside the VPN. Infrastructure Monitoring is done locally at i2CAT island. An OML script gathers information about the status of the servers and switches plus the amount of HD and memory available in the servers and sends and injects these data into the infrastructure DB at TUB.

Functionality of components in Figure 5.1 is described in Table 1. Being **Verdaguer**, **Rodoreda** and **March** the Virtualization servers equivalent to **cseedelphi** and **cseebosra**.

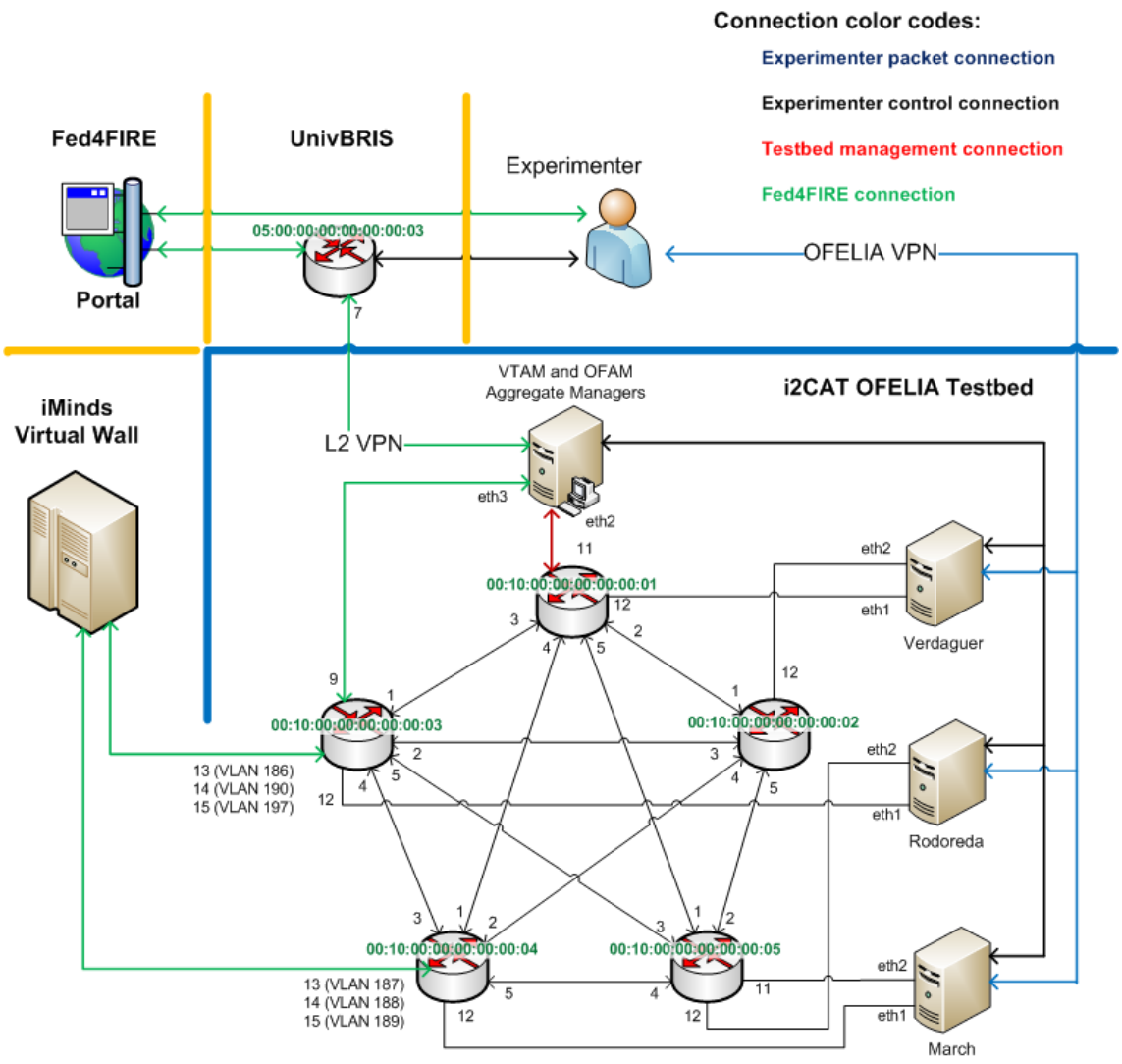


Figure 5.1: i2CAT OFELIA Facility

5.1.2 UnivBris island

Figure 5.2 shows the general architecture of the UnivBris OFELIA island for Fed4FIRE. In terms of OpenFlow networking resources, there are 4 packet switches (**NEC IP8800** running on OpenFlow v1.0) and 3 optical switches (**ADVA FSP3000**) that are available for experimenters to reserve and use for their experiments. ADVA optical switches are interconnected through a 192×192 beam steering Polatis fiber switch that is used as optical backplane switch. This switch offers the possibility to switch the optical signal through long fibers (50 Km) to enable more scenarios for experimentation. The testbed also offers compute resources in terms of virtual machines that can be created on two virtualization servers (their internal names are: **cseedelphi** and **cseebosra**) in the testbed. These virtual machines can be created and instantiated by the experimenters and act as the source and sink of traffic for the

OpenFlow network. The short description of the role of each component of the testbed is described in **Table 1**.

Required parameters are missing or incorrect.

Figure 5.2: General architecture of Ofelia (UnivBris) for Fed4FIRE.

Table 1: Description of each component inside the UnivBris OFELIA island

Component	Symbol	Description
Packet Switch		OpenFlow packet switch (model NEC IP8800) on which selected OpenFlow ports can be reserved and used by experimenters.
Optical Switch		OpenFlow optical switch (model fixed-grid WDM ADVA) on which selected ports and wavelengths can be reserved and used by experimenters.
SSH gateway		SSH gateway through which experimenters can have access to their virtual machines from public internet. The IP and port of the SSH gateway is 137.222.204.15 and 22 respectively.
Virtualization Servers		Virtualization servers on which experimenters create and instantiate their virtual machines.
Optical Flowvisor		Optical Flowvisor is the software which allows many experimenters to concurrently use the testbed. For the packet part of the testbed, this is achieved by slicing the packet switches based on VLAN and assigning a separate VLAN to each experimenter. For the optical part of the testbed, a set of unreserved wavelengths is given to each experimenter.
VTAM and OFAM Aggregate managers		VTAM and OFAM aggregate managers are software components which are used to manage virtualization servers and network resources that are available for experimenters.
Monitoring server		Monitoring server collect details about the state of the different components of the testbed and publishes it on the First Level Support website of the Fed4FIRE.
Fed4FIRE portal		Fed4FIRE portal allows experimenters to list and reserve resources on the testbed as well as other testbeds inside the Fed4FIRE consortium.

Fed4FIRE First Level Support (FLS)		Fed4FIRE First Level Support website provides an overview of the state of all Fed4FIRE testbeds. The details include, the ping latency, GetVersion and ListResources tests, internal status of the testbed and when was this conducted.

5.1.3 Functional description

Figure 5.3 shows a more functional view of the OFELIA islands. At the first bottom layer (hardware layer), there are the hardware devices on which the experiments will run. The hardware devices can be categorized into two categories: network and compute resources. The network resources consist of the OpenFlow switches while compute resources consist of virtualization servers where the experimenters' virtual machines will be located. The hardware devices are virtualized/sliced through the virtualization components in the virtualization layer. The network resources are sliced through a program called "FlowVisor" which can slice both packet and optical switches based on packet header fields and physical properties such as in-port and wavelength. The compute resources are virtualized through the use of XEN tools to create Debian-based virtual machines for experimenters. The third layer is the "Resource Control" which is in charge of authenticating requests for virtualized resources and actually allocating and provisioning the accepted requests. The last layer is the Experimenters' layer which offers different tools for experimenters to request resources for experimentation on OFELIA islands.

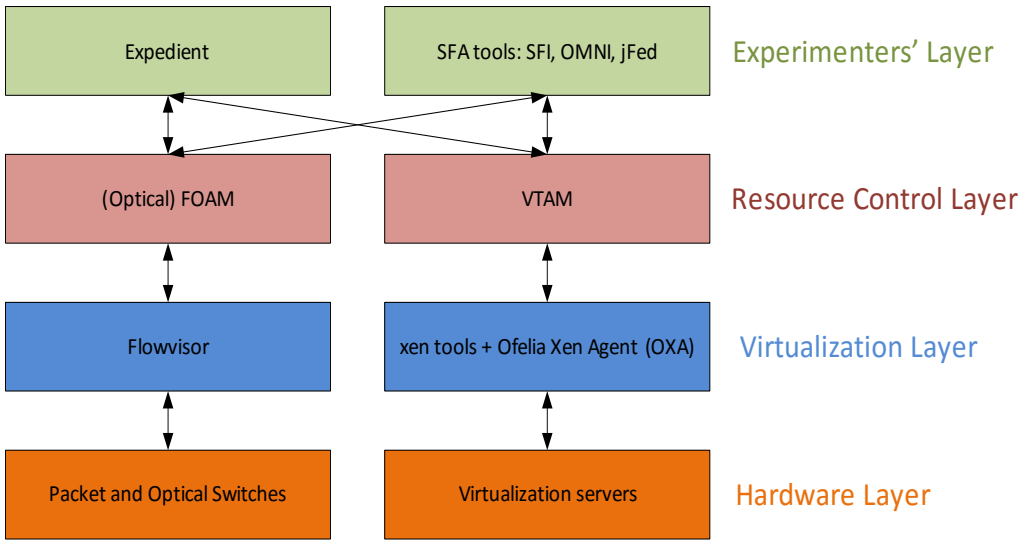


Figure 5.3: Functional view of Ofelia for Fed4FIRE.

5.2 Basic Functionality

5.2.1 Documentation

A detailed documentation and manual for the UnivBris OFELIA testbed are provided through the testbed website: https://univbrisofeliaf4f.blogs.ilrt.org/. This website provides the necessary

information and instructions required by experimenters to know how to experiment on the testbed. Detailed examples of resource advertisement and slice reservation RSpecs are provided in this website.

In a similar way, i2CAT OFELIA island documentation is provided at https://github.com/fp7-ofelia/ocf/wiki/Working-in-Fed4FIRE . This website is part of the wiki page of the OCF (OFELIA Control Framework) repository, where the code is publicly available, and complements the information provided in UnivBris' webpage. This wiki includes detailed description of the i2CAT island, as well as instructions on how to request resources and samples of the RSpecs from the Fed4FIRE point of view.

5.2.2 Policies

5.2.3 Facility Monitoring (FLS)

The basic status of the state of the aggregate managers (AMs) (i.e. OpenFlow networking AM and Compute resources AM) of OFELIA testbed can be found at FLS dashboard: https://flsmonitor.fed4fire.eu/. The "GetVersion" field indicates whether the relevant aggregate manager can be contacted (green color) or not (red color). The numeric value of the "Free Resources" field does not provide meaningful information since the resources at the testbed are virtualized and sliced for experimenters.

The internal status of the island on the FLS website is gathered from the testbed through OML streams. The monitoring of the island is done through the Zenoss monitoring framework which is installed on a physical server in the testbed. Monitoring software (developed in Python) is used to interact with the Zenoss framework to query and collect the status of the devices. This software feeds the collected monitoring data to the Fed4FIRE FLS website (https://flsmonitor.fed4fire.eu/) for both UnivBris and i2CAT Ofelia islands. A cron job in Linux is used to run the monitoring script every 5 minutes to send monitoring data to the FLS monitoring website.

5.2.4 Testbed interface for experimenters

5.2.4.1 SFA AM approach

The SFA approach for the aggregate managers in cycle 3 has continued the path stated in previous cycles. In cycle 2, SFA GENI v3 was developed for both OFELIA AMs (Virtualization AM and OpenFlow AM) and started to be tested. During cycle 3, testing was completed and GENI v3 was fully deployed.

OFELIA AMs also use standard API RSpecs, although the way the Virtualization AM parses it does not use all the RSpec fields. For example, API RSpecs allows request a VM specifying the OS image to be loaded, size of the VM HD, number of cores, etc. However, OFELIA VT AM uses a default OS image and HD size, ignoring what is specified in the RSpec. But as it parses correctly the RSpec, the AM could be modified in the future to create VMs taking into account all the specifications defined in the RSpec.

5.2.4.2 Own Testbed interface approach

OFELIA testbed has its own interface called OCF (OFELIA Control Framework) [16] which allows to control OFELIA own resources. This OCF has its own web interface called Expedient that enables experimenters to create and run experiments within the OFELIA facilities. Expedient communicates with the AMs using the OFELIA API, which was based on the GENI API but was not fully compatible.

OFELIA interface is kept as an active tool and it can be used to request OFELIA resources but it does not allow managing other federated testbeds' resources. For that, the adaptation of the GENI SFA API v3 was included in the AMs as an independent interface able to interact with federation tools as jFed and MySlice Portal.

5.2.5 IPv4/IPv6 Connectivity

All nodes are reachable through IPv4 addresses only. IPv6 is not available.

5.3 Additional Enhancements

5.3.1 Infrastructure monitoring

Infrastructure monitoring for OFELIA testbed has been implemented in third cycle by gathering information with a python script and OML streams are used in order to populate a central Fed4FIRE database hosted at TUB.

The python script, running as a service in the main servers of both OFELIA islands, checks hourly the status of the island's servers and switches (up or down) and, for the virtualization resources servers, measures the free HD and RAM available. These gathered data are injected into OML streams and sent to the TUB infrastructure monitoring OML DB.

5.3.2 Advanced reservation

OFELIA testbed does not offer advanced reservation. Resources are provided when requested (if available) and are reserved for the experimenter until the expiration date defined in the request or when the experimenter releases them.

5.3.3 SLA

Although SLA is not planned to be implemented in OFELIA testbed and there is no integration with SLA Dashboard, infrastructure monitoring data sent to the OML database provides information about the status of the components of the facility (servers and switches). With this monitoring data it is possible to evaluate the availability of the testbed resources and thus define SLAs based on it.

5.3.4 Experiment Control

In OFELIA, once the resources are provisioned, the VMs can be directly controlled by accessing them via gateway for SSH. Once the VMs are accessed, experimenter has root privileges to install any software required for the experiment.

The OpenFlow switches are also controlled by providing them the address of a controller tool that modifies their flow tables according to the experimenter's needs.

The OpenFlow switches are also controlled through a SDN controller that is able to modify their flow tables according to the experimenter's needs. The address of the controller in each experiment should be specified in order to establish the communication between this controller and the switches in the experiment. Usually, this controller is deployed in one of the provisioned VMs, although it can be deployed anywhere reachable through internet. OFELIA provides a pre-installed controller tool in the VMs such as NOX **[13]**, although there are several other possibilities available (e.g. POX [14] and RYU [15]).

5.3.5 Layer 2 connectivity

As commented in section 5.1, OFELIA resources are not publicly available to the internet. AMs from both islands have a public IP provided by UnivBris.

Both islands are also connected through GEANT VLANs to iMinds VirtualWall, which is already connected to other Fed4FIRE testbeds. This provides OFELIA connectivity with all other testbeds. If needed, specific connections with other testbeds can be set up.

If resources are provided through jFed, it is possible to access through ssh to the requested VMs just by double clicking over the VM icon in the jFed screen. In the background, what jFed does is to ssh to an iMinds VM which is already in the OFELIA VPN and from this VM, ssh to the requested OFELIA VM. This process is transparent to the experimenter who only sees a direct ssh connection to the OFELIA VMs.

6 NITOS

6.1 Facility description

UTH operates NITOS Future Internet Facility, which is an integrated facility with heterogeneous testbeds that focuses on supporting experimentation-based research in the area of wired and wireless networks. NITOS is remotely accessible and open to the research community 24/7. It has been used from hundreds of experimenters all over the world.

The main experimental components of NITOS are:

- Three wireless experimentation testbeds that consist of 100 powerful nodes (some of them mobile), that feature multiple wireless interfaces and allow for experimentation with heterogeneous (Wi-Fi, WiMAX, LTE, Bluetooth) wireless technologies.
- A Software Defined Radio (SDR) testbed that consists of Universal Software Radio Peripheral (USRP) devices attached to the NITOS wireless nodes. USRPs allow the researcher to program a number of physical layer features (e.g. modulation), thereby enabling dedicated PHY layer or cross-layer research.
- A Software Defined Networking (SDN) testbed that consists of multiple OpenFlow technology enabled switches, connected to the NITOS nodes, thus enabling experimentation with switching and routing networking protocols. Experimentation using the OpenFlow technology can be combined with the wireless networking one, hence enabling the construction of more heterogeneous experimental scenarios.

Figure 4.1: NITOS outdoor Testbed.

Figure 5.2: NITOS RF-Isolated Testbed.

The testbed is based on open-source software that allows the design and implementation of new algorithms, enabling new functionalities on the existing hardware. The control and management of the testbed is done using the cOntrol and Management Framework (OMF) open-source software. NITOS supports evaluation of protocols and applications under real world settings and is also designed to achieve reproducibility of experimentation.

6.2 Basic Functionality

6.2.1 Documentation

NITOS documentation is publicly available through this URL: http://nitlab.inf.uth.gr/doc. Using this documentation a new experimenter can be introduced to the way he can use NITOS; how to reserve resources and execute basic experiments with the various technologies that NITOS supports (WiFi, LTE, WiMax, OpenFlow, etc). This documentation is constantly being updated in order to meet the demands of experimenters using NITOS.

6.2.2 Policies

NITOS provides access to the federation experimenters without further approval once they have a valid slice from Fed4FIRE. Furthermore, support is provided to all the experimenters through the dedicated google group.

6.2.3 Facility Monitoring (FLS)

In order to monitor NITOS facility, information is gathered and simple OML streams are used to populate a database with this information (already implemented in the first cycle).

6.2.4 Testbed interface for experimenters

NITOS is currently supporting multiple interfaces for the experimenters to allocate and reserve resources.

- SFA CLI tools: using command line tools (sfi, omni, etc) an experimenter can get an advertisement of all the resources and then reserve resources to conduct their experiments.
- Fed4FIRE portal: experimenters can access the Fed4FIRE portal (https://portal.fed4fire.eu) with their browsers, for a more user friendly experience, in order to get an advertisement of free resources and reserve them.
- jFED: jFED is a java based Framework for testbed federation. NITOS is being supported by jFED.

6.2.4.1 SFA AM approach

NITOS has jointly developed with NICTA a framework called Broker that supports SFA v2 API with extended v3 RSpecs. During the third Fed4FIRE development cycle, Broker has been extended with an SFA v3 API.

6.2.4.2 Own Testbed interface approach

Every testbed supports different technologies, thus there is a need for special tools that support those technologies. In this context, for NITOS a new portal (http://nitos.inf.uth.gr) has been developed with various tools that can help an experimenter conduct experiments in any of the NITOS testbeds with a variety of technologies supported by NITOS (WiFi, LTE, WiMax, OpenFlow, etc).

6.2.5 IPv4/IPv6 Connectivity

IPv4 is being supported through a gateway server for each one of NITOS testbeds.

6.3 Additional Enhancements

6.3.1 Infrastructure monitoring

Infrastructure monitoring has been already implemented in cycle 2.

6.3.2 Advanced reservation

The Broker Framework used in NITOS is capable of supporting advanced reservation features like future reservations and both bound and unbound reservation requests.

6.3.3 Reputation

NITOS is providing all the needed information to the reputation service, thus NITOS is compatible with the Fed4FIRE reputation service.

6.3.4 Permanent Storage

Every Slice has its own home folder which has a quota of 40 GByte. At the same time an experimenter can save images (in ndz format) that can be later loaded on nodes, without any size limitation.

6.3.5 Experiment Control

For experiment control, OMF v6 is supported by NITOS, along with the Broker Framework, which has an XMPP interface that supports FRCP messages.

6.3.6 Layer 2 connectivity

UTH maintains L2 connections through GEANT with Starlight (USA) and with the following institutions:

- i2Cat
- UNIVBRIS
- iMinds
- NTUA

7 NETMODE

7.1 Facility description

NETMODE Wireless Testbed consists of a control server and 20 wireless nodes. The wireless nodes are scattered around the 3rd floor (indoor nodes) and the roof (outdoor nodes) of the Electrical and Computer Engineering building at the National Technical University of Athens, thus allowing for wireless networking experiments under various topologies and traffic conditions. NETMODE wireless testbed is provided as a service to experimenters by reserving nodes and wireless channels for a specific timeframe (currently unconstrained). The floor plan along with more information on the wireless testbed is provided in http://www.netmode.ntua.gr/testbed/

18 Alix-based (Nodes 1-18)

- alix3d2 board

- 100Mbit Ethernet port

- 2 802.11 a/b/g interfaces

- 1GB flash card storage device

2 PC-based (Nodes 19-20)

- Intel Atom CPU

- 1Gbit Ethernet port

- 2 802.11 a/b/g/n interfaces

- 250 GB hard disk

7.2 Basic Functionality

7.2.1 Documentation

During the course of cycle 3, the NETMODE Testbed description has been updated (http://www.netmode.ntua.gr/testbed/) and the required testbed tutorial is available online (http://www.netmode.ntua.gr/).

7.2.2 Policies

The current federation architecture allows each testbed to enforce its policies at a testbed level once experimenters contact their AM. The majority of the testbeds currently allow any experimenter without further approval to use their resources, once they have acquired a valid Fed4FIRE certificate; that is the case also for NETMODE testbed with its current version of Aggregate Manager (NITOS/NICTA Broker). However during the course of cycle 3, the AM has been extended in order to support quota per user group. Therefore NETMODE will incorporate it in the next AM upgrade, as soon as the stable version has been released.

7.2.3 Facility Monitoring (FLS)

Facility monitoring is supported since Cycle 1. This type of monitoring provides information about the availability status of the NETMODE testbed. It is based on monitoring the key components of the facility. Zabbix is used to monitor the status information of the NETMODE resources (i.e. per node

whether up and running or down). An OML wrapper script (Ruby) transforms the data from Zabbix format into OML streams. This wrapper fetches the required information on the resources from Zabbix and then injects the data into OML streams with the help of oml4r, the ruby client library for OML. The OML streams are transported to the OML server located at the federation level (at iMinds) as a central collection endpoint. This is in charge of calculating the overall availability status of NETMODE testbed and push status (Green, Red, or Amber) to the Fed4Fire FLS.

7.2.3.1 SFA AM approach

NETMODE exposes testbed resources through SFA AM APIv2 with RSpecs v3 extended with reservation information (leases). This is achieved using the latest release of the (NITOS/NICTA) Broker, which acts as an Aggregate Manager.

During Cycle 3, the developers of the NITOS/NICTA broker have been working towards providing a stable version of the Broker supporting SFA v3 API. As soon as it is officially released NETMODE AM will be upgraded to the new version.

With regards to adopting the ontology based information model developed in WP5 (Open-Multinet ontology - OMN), in Cycle 3 the NETMODE advertisement RSpec is translated to Resource Description Framework (RDF) graph using the OMN library. Using appropriate testbed-specific inference rules the knowledge base is extended; the knowledge base is stored to a Sesame repository and is available via a SPARQL Protocol And RDF Query Language (SPARQL) endpoint. More details are provided in Deliverable 5.6.

7.2.4 IPv4/IPv6 Connectivity

NETMODE resources are accessible via SSH using either an IPv4 gateway, that is the testbed's experiment controller (vnews.netmode.ntua.gr), or IPv6 directly at the reserved nodes.

7.3 Additional Enhancements

7.3.1 Infrastructure monitoring

Infrastructure monitoring is supported since Cycle 2. This type of monitoring provides information about the infrastructure (substrate or virtual) resources to the federation services. The monitoring information depends on the federation service requirements, for example the reputation service is interested for resources used (physical or virtual) during an experiment's lifetime while the reservation broker is interested in historical measureable characteristics of physical resources.

Zabbix is used to monitor the status information of the NETMODE substrate resources (per node whether up and running, or down). An OML wrapper script (Ruby) transforms the data from Zabbix format into OML streams. This wrapper fetches the required information on the resources from Zabbix and then injects the data into OML streams with the help of oml4r, the ruby client library for OML. The OML streams are transported to the OML server located at the federation level (at TUB) as a central collection endpoint. This endpoint is used for retrieving Infrastructure monitoring data via the Data Broker (Manifold OML GW).

7.3.2 Advanced reservation

The NETMODE testbed supports in advance reservation of resources, via multiple experimenters tools (MySlice, jFed, omni etc) . Specifically in the case of the Fed4FIRE portal (MySlice) the use of the Reservation Broker is possible for the experimenter, supporting unbound requests for resources to the NETMODE testbed (since cycle 2). During cycle 3 this process has been further enhanced as documented in Deliverable 5.6.

7.3.3 SLA

The SLA tool prototype has been deployed at the NETMODE testbed in cycle 2. A singe Service Level Objective is supported (node availability). During Cycle 3 the SLA tool prototype at the NETMODE testbed and corresponding APIs have been thoroughly tested and further enhanced.

7.3.4 Reputation

The NETMODE testbed has been added to the Reputation Service in cycle 2, advertising a single technical service (node availability) and a non-technical service (overall experience).

7.3.5 Permanent Storage

The testbed's Experiment Controller (vnews.netmode.ntua.gr) hosts experimenters' home directories, which can be accessed at all times.

7.3.6 Experiment Control

Experiment Control is supported since Cycle 1. NETMODE testbed supports OMF-6.0 experiment and resource controllers. Therefore the experimenter can use OEDL (OMF Experiment Description Language) to describe and execute his experiments.

7.3.7 Layer 2 connectivity

Layer2 connectivity (static VLANs via GEANT) has been established between the NETMODE testbed (and its Gateway) with other Fed4FIRE testbeds (NITOS and w-iLab.t).

GRE based tunnels have been tested between the PLE – NETMODE infrastructures as a test case for interconnecting testbeds of the federation also supporting GRE tunnels.

PlanetLab Europe supports the creation of a tun/tap interface at the sliver level through the vsys1 interface; this permits to execute privileged code in the LXC or VServer container and thereby create such interfaces. However, mainly due to upgrades to the underlining PLE infrastructure, at this moment GRE is not directly supported from vsys. The only alternative at the moment is to set up the tunnel on the resource as an administrator (thus provided by the infrastructure owner).

On the other hand, NETMODE testbed nodes are behind NAT (Experiment Controller - EC). In the case examined the GRE tunnel is terminated at the EC. Once again the action requires administrative rights.

Based on the above an experimenter currently should explicitly ask for a link between a PLE resource and NETMODE and the administrators of both testbeds will set it up.

```
PLE node (132.227.62.99)
ip tunnel add gretest mode gre remote 147.102.13.123 local 132.227.62.99 ttl 255
```

[1] https://www.usenix.org/event/atc11/tech/final_files/Bhatia.pdf

```
ip addr add 10.0.6.2 dev gretest
ip link set gretest up
ip route add 10.0.6.0/24 dev gretest
ip route add 10.0.0.0/24 dev gretest

NETMODE EC (147.102.13.123)
 ip tunnel add gretest mode gre remote 132.227.62.99 local 147.102.13.123 ttl 255
 ip addr add 10.0.6.3 dev gretest
 ip link set gretest up
 ip route add 10.0.6.0/24 dev gretest
```

A disadvantage of this configuration is that the tunnel would be accessible on the PLE node by all logged users, and the same stands on the NETMODE gateway.

8 Exogeni NORBIT

8.1 Facility description

The NICTA NORBIT testbed is not maintained anymore, thus there is no plan to develop and maintain its support of the AM API required for Fed4FIRE federation. However, NICTA is now hosting an ExoGENI rack, which supports the AM API and thus is made available as a federated testbed to Fed4FIRE.

ExoGENI (http://www.exogeni.net) is based on an extended Infrastructure-as-a-Service (IaaS) cloud model with coordinated provisioning across multiple sites and a high degree of control over intra- and inter-site networking functions. Most researchers will use a standard cloud computing stack to instantiate and manage virtual machines.

This structure enables a network of private ExoGENI IaaS clouds to operate as a hybrid community cloud. ExoGENI combines this multi-domain cloud structure with rich networking capabilities through direct Layer 2 site connectivity to national circuit backbone fabrics, linkages to other national and international networks, and OpenFlow-enabled dataplanes within each site.

Provisioning individual compute resources (virtualized and bare-metal) from rack resources. Users will be able to supply boot images for virtualized instances; bare-metal instances will be limited to a few vetted images. We support Linux (bare-metal and virtual) and Windows (virtual, possibly also bare-metal).

The ExoGENI NICTA rack consists of a head node and 10 6-core worker nodes. It is based on IBM System X servers, and also features two additional IBM storage servers.

8.2 Basic Functionality

8.2.1 Policies

8.2.1.1 PDP

NICTA has developed and deployed its own version of the PDP.

8.2.2 Facility Monitoring (FLS)

Operational since cycle 1, now extended towards the ExoGENI Rack.

8.2.3 Testbed interface for experimenters

8.2.3.1 SFA AM approach

Exogeni supports the AM APIv2.

8.3 Additional Enhancements

8.3.1 Experiment Monitoring

NICTA is the main developer of OML and as such NICTA offers many OML-instrumented applications such as iperf, collectd, gpsd, ping, or WattsUp? Power Monitor.

The complete list of OML-instrumented applications available to experimenters is located at:

http://mytestbed.net/projects/omlapp/wiki/OML-instrumented_Applications

8.3.2 Experiment Control

The ExoGENI NICTA rack provides OMF v6 (experiment orchestration), Lawiki Interface (experiment life-cycle management), OML v2.11 (measurement collection).

8.3.3 Layer 2 connectivity

Layer-2 connectivity to Internet2 was previously available through a dedicated link provided by the Australian Research and Education Network. However that link is no longer available to NICTA. NICTA is currently working with US (GENI) and EU partners (Fed4FIRE) to re-establish a similar L2 link functionality through tunneling over L3 with dedicated network applicances.

9 KOREN

9.1 Facility description

KOREN (KOrea advanced REsearch Network) is a non-profit testbed network infrastructure established for facilitating research and development and international joint research cooperation. It provides quality broadband network testbed for domestic and international research activities to the industry, academia, and research institutions, enabling testing of future network technologies and supporting R&D on advanced applications. KOREN has multiple islands in Korea, spread across several cities. OpenFlow islands are located in three major Cities, Seoul, Pusan and Daejeon.

Each KOREN OpenFlow island comprises OpenFlow-capable L2 switches, servers with virtual machines and Juniper switch that are attached to WDM (Wave Division Multiplexing) switch. Juniper switch is a gateway to dynamic circuit network (also acknowledged as DCN, ION or Autobahn) that provides the on-demand WAN connectivity. One hundred VLANs are dedicated for dynamic circuit switching and it makes KOREN afford one hundred concurrent experiments over WAN.

9.2 Basic Functionality

9.2.1 Policies

An identity provider with SFA X.509 certificates is supported. Currently, in KOREN we have generated a self-signed X.509 root certificate and every user of KOREN has a unique certificate signed by that root certificate. This root certificate is possible to upload F4F's certificate directory.

9.2.1.1 PDP

It is not available if it's not essential, since a user can access to the resources via SSH and get the privilege of administrator on the node.

9.2.2 Facility Monitoring (FLS)

KOREN testbeds have been in use on the OML stream for facility monitoring about simple facility information. To enhance facility monitoring framework for experiment measurement, we have provided the facility monitoring data stored in the backend of OML server and injected the stream of data to the central OML server of FLS.

KOREN customized OML script for specified OML data such as ICMP, SSH, as well as basic sFlow data. Thus KOREN are now sending the internal status of the KOREN islands to the First Level Support website dashboard every 10 minutes. The status of the SFA-enabled aggregate manager is displayed as colors which are red/amber/green on the FLS website dashboard.

9.2.3 Testbed interface for experimenters

In the Fed4FIRE mechanism chosen for implementing resource discovery, requirements, reservation and provisioning is the Slice Federation Architecture (SFA) standard (specifically GENI AM API v3), together with ontology based resource specifications (Rspecs).

In the first cycle an instance of the generic SFA wrapper has been deployed in production state and its operation has been evaluated as robust and bug free. In addition to this wrapper, KOREN has developed Openstack plugin by using the existing plugin and exposed testbed resources through SFA by implementing an XML-RPC interface as part of Openstack AM, supporting AM v3 API with Rspecs GENI v3. This AM of KOREN is deployed in development state and it is used for testing purposes. Besides, Openflow resources in KOREN are discoverable by certified any F4F management system.

9.3 Additional Enhancements

9.3.1 Infrastructure monitoring

KOREN has currently upgraded its monitoring system to Zabbix. Previously, it was using PerfSonar. At this stage, Zabbix monitors CPU load and Utilization of the hosts on KOREN OpenFlow islands. This monitoring information will be provided to the central Fed4FIRE collection server as required from WP6.

As KOREN upgrade proceeds, sFlow also monitors OpenFlow switches and legacy switches. Flow stats and other intrinsic OpenFlow stats are already available to OpenFlow Controller, but KOREN will also provide the stat information of the physical switch, such as packet count per port, error rates on physical port, through sFlow monitoring system. It is not decided if the other partners need this extended information. But if such demand arises, we'll provide it through OML.

9.3.2 Experiment Monitoring

To use KOREN testbeds for experiments, users typically need to monitor switch(s) and VMs. For intrinsic OpenFlow switch monitoring, users' own OpenFlow controllers monitor the flows set and other parameters. This type of monitoring is useful because many Openflow applications, which sit above core of OF controllers, need to have access to this monitoring information to function properly. Although KOREN already provides this type of monitoring, we consider we may need additional monitoring capability so that we can monitor OpenFlow switch itself and associated VMs. At the moment, we are not aware of the availability of such tools. And if necessary, we may come up with the solutions that would help the experimenters to monitor. This is one of the issues KOREN would pursue in future cycle.

9.3.3 Experiment Control

In KOREN, once the resources are provisioned, the VMs can be directly controlled by accessing them via gateway for SSH. The Openflow switches are also controlled by providing them the address of a controller tool that modifies their flow tables according to the experimenter's needs.

It is being studied to include FRCP in the VMs, so they can be controlled remotely in the future cycle.

9.3.4 Layer 2 connectivity

KOREN resources are already available on the public internet and are accessible via SSH with a public IPv4 address. And KOREN is ready to provide the L2 connectivity to the F4F partners through L2 over L3 tunneling.

10 PerformLTE

10.1 Facility description

PerformLTE tesbed follows a holistic approach combining different type of equipments, LTE radio access emulators equipments, Evolved Nodes B (eNBs), User Equipments (UEs) both commercial and engineered to provide measurements, and an Evolved Packet Core (EPC) emulation system. All these elements can be combined and experimentation can be performed in all the components of a LTE network. In general terms LTE connectivity is provided through three different solutions, each one with a focus on a different research aspect, moving between emulation and real-world environments, three main scenarios can be differentiated based on the radio access type:

1. Scenario based on a LTE radio access emulator (traditionally use for conformance testing) which provides a full LTE end to end emulation, including channel emulation with different fading profiles and operation in all the standardized LTE bands, both FDD and TDD. These equipments provides configuration of multiple levels of the LTE Radio Access Network (RAN) stack, so researches can study the effect of different parameters as well as the motorization of the full network.

2. Scenario based on commercial off-the-shelf eNBs connected to a commercial EPC. This scenario provides configurations very close to the one provided by operators. Experimenters are able to carry out their test in a realistic and dedicated LTE deployment.

3. Scenario based on commercial LTE networks, in which the testbed offers devices connected to commercial mobile networks. Devices are equipped with TestelDroid, a tool which enables the monitoring of radio information and IP traffic. These information can be correlated in order to troubleshoot IP issues related on radio impairments.

10.2 Basic Functionality

10.2.1 Documentation

Testbed documentation is available at: http://www.morse.uma.es/performnetworks.

10.2.2 Policies

Everyone with a valid F4F certificate can execute the basic experiment without extra approval.

10.2.3 Facility Monitoring (FLS)

The monitoring script checks the availability of the PerformLTE AM and PerformLTE EC. The status of the AM is tested using nmap and checking that the AM service is running at IP morse.uma.es and port

11005. The status of the EC is tested using nmap and checking the SSH service is running in the standard SSH port in the IP address of EC.

These measurements (availability and timestamp of the check) are injected in "tcp:flsmonitor.ilabt.iminds.be:3003" using oml4py procedures.

Fed4Fire Testbeds

Testbed Name	Ping Latency (ms)	GetVersion Status	Free Resources	Internal Status	Aggregated Status	Login Status
10G Trace Tester	41.82	SUCCESS	1	SUCCESS	SUCCESS	SUCCESS
BonFIRE	18.37	SUCCESS	11	SUCCESS	SUCCESS	SUCCESS
Bristol openflow	12.13	SUCCESS	5	SUCCESS	SUCCESS	SUCCESS
Bristol VTAM	32.99	SUCCESS	2	SUCCESS	SUCCESS	SUCCESS
C-Lab	51.63	SUCCESS	92	SUCCESS	SUCCESS	SUCCESS
ExoGENI NICTA	293.72	SUCCESS	22	SUCCESS	SUCCESS	SUCCESS
FUSECO	17.02	SUCCESS	6	SUCCESS	SUCCESS	SUCCESS
i2CAT openflow (SDNRM)	11.19	SUCCESS	5	SUCCESS	SUCCESS	SUCCESS
i2CAT VTAM (CRM)	11.99	SUCCESS	3	SUCCESS	SUCCESS	SUCCESS
Koren	274.66	SUCCESS	3	SUCCESS	WARNING	FAILURE
LOG-a-TEC	35.37	SUCCESS	2	SUCCESS	SUCCESS	SUCCESS
NETMODE	85.56	SUCCESS	20	SUCCESS	SUCCESS	SUCCESS
NITOS Broker	90.80	SUCCESS	57	SUCCESS	SUCCESS	FAILURE
Perform LTE	60.83	SUCCESS	1	SUCCESS	SUCCESS	SUCCESS
Planetlab Europe	37.91	SUCCESS	250	SUCCESS	SUCCESS	FAILURE
SmartSantander	57.07	SUCCESS	0	SUCCESS	SUCCESS	no data
UC3M optical	41.79	SUCCESS	0	SUCCESS	SUCCESS	WARNING
Virtual Wall 1	0.26	SUCCESS	63	SUCCESS	SUCCESS	SUCCESS
Virtual Wall 2	3.88	SUCCESS	54	SUCCESS	SUCCESS	SUCCESS
Virtual Wall 2 (openflow)	0.90	SUCCESS	2	SUCCESS	SUCCESS	SUCCESS
w-iLab.t 2	5.02	SUCCESS	1	SUCCESS	SUCCESS	SUCCESS

Figure 6 PerformLTE at FLS monitor

10.2.4 Testbed interface for experimenters

UMA has developed an Aggregate Manager (AM) for PerformLTE testbed compliant with AM GENI v3 API. The implementation is based on GCF project which provides a reference implementation of the GENI Aggregate Manager API. The Aggregate Manager implementation has been tested using jFed Probe GUI tool. The AM provides a federated SSH access to the Experiment Controller (EC) of the testbed.

The RSpec definition of the testbed provides a monolitic specification of the Experiment Controller (EC) of the testbed. This definition can be used to gain SSH access to Experiment Controller (EC) of PerformLTE using, for example, the jFed Experimenter GUI tools.

The EC also contains reference experiments described in OEDL. The experimenters can modify and launch their customized experiments using omf_ec procedures available at the EC.

10.2.5 IPv4/IPv6 Connectivity

Public IP for the Aggregate Manager : performlte.morse.uma.es, port: 11005

Public IP for EC (EC supports SSH login provided through the AM): performlte.morse.uma.es, port: 11004

10.3 Additional Enhancements

10.3.1 Infrastructure monitoring

10.3.2 Advanced reservation

10.3.3 SLA

10.3.4 Reputation

10.3.5 Permanent Storage

All the data received by the Experiment Controller is stored in a PostgreSQL database accessible from the SSH session. It will create several tables for experiment metadata, logs and so on. It will also create one table for each Measurement Point enabled, in which every sample is going to be stored.

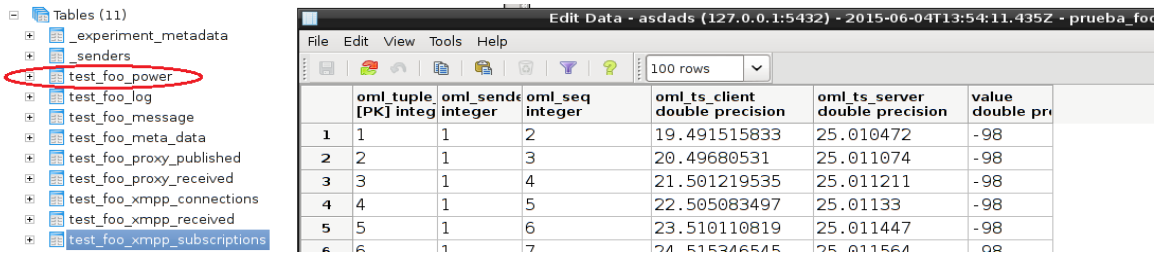


Figure 7 Data stored in the SQL database

10.3.6 Experiment Control

Our infrastructure uses the OMF framework to control all the resources available at the testbed. To run experiments the experimenter have to write the experiment using OEDL, an Experiment Description Language that in reality is a subset of the Ruby programming language. The script is executed by the Experiment Controller (EC) available at the testbed, a console application available at the testbed. SSH access to the EC has been introduced in the previous section.

Scripts are executed by using the console executable omf_ec (short for “OMF Experiment Controller”).

The Experiment Controller interprets the script and trigger the actions that have to be performed by each one of the resources. XMPP server is used for the delivery of these actions to the Resource Controllers. The logical architecture of doing an experiment in our testbed is shown in the Figure 8:

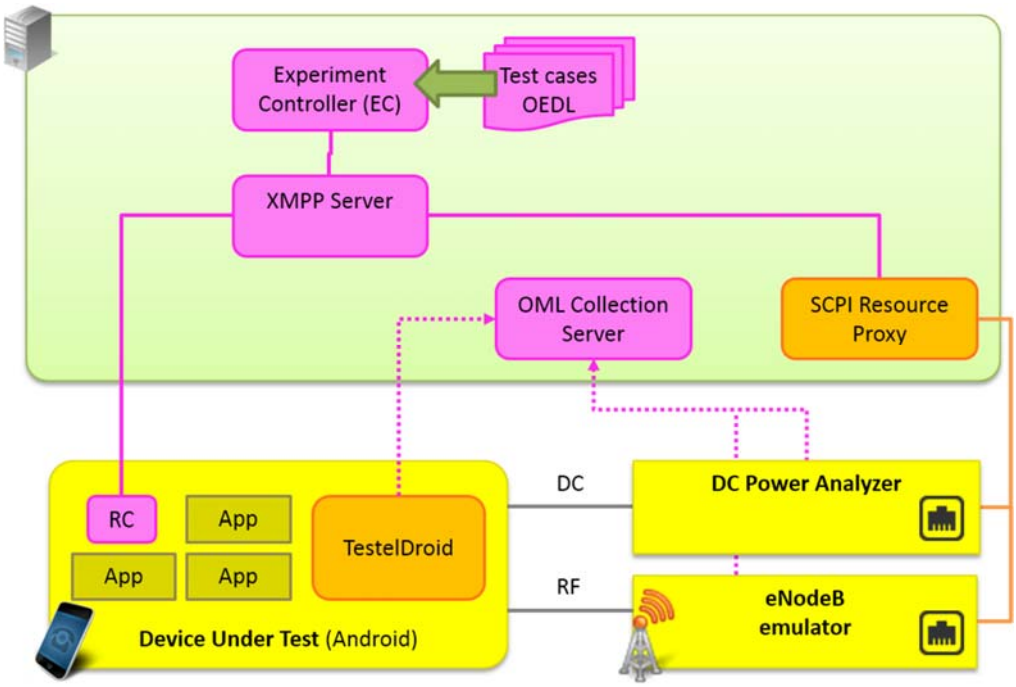


Figure 8 PerformNetworks OMF control

The measurements collected during the experiments are stored in a PostgreSQL database for further extraction.

10.3.6.1 Control of Android devices

The management of Android devices is based on the Resource Controller implemented by NITOS for Android operating system.

TestelDroid monitoring tool for Android devices is a tool developed by UMA. Advanced monitoring capabilities provided by TestelDroid can be managed through the Android Resource controller.

The following The following excerpt of code code provides the OEDL definition of TestelDroid monitoring capabilities which can be used during an the experiment:

```
defApplication('TestelDroid') do |app|
app.description = 'Simple Definition for the TestelDroid application'
# Define the path to the binary executable for this application
app.binary_path = {package: 'com.ad.testel', service: 'AutomationReceiver'}
# Define the configurable parameters for this application
app.defProperty('EXTRA_NETWORK', 'Log network information', '', {:type =>
'parameter',
:action => 'CONFIGURE', :dynamic => true})
app.defProperty('EXTRA_TRAFFIC', 'Log network traffic', '', {:type =>
'parameter', :action
=> 'CONFIGURE', :dynamic => true})
app.defProperty('EXTRA_GPS', 'Log GPS information', '', {:type =>
```

```
'parameter', :action =>
'CONFIGURE', :dynamic => true})
app.defProperty('EXTRA_NEIGHBOUR', 'Log neighbour cells information', '',
{:type =>
'parameter', :action => 'CONFIGURE', :dynamic => true})
app.defProperty('EXTRA_PROFILE', 'Log profile information', '', {:type =>
'parameter',
:action => 'CONFIGURE', :dynamic => true})
app.defProperty('EXTRA_PROFILE_SCENARIO', 'Profile scenario', '', {:type =>
'parameter',
:action => 'CONFIGURE', :dynamic => true})
app.defProperty('EXTRA_PROFILE_TECH', 'Profile technology', '', {:type =>
'parameter',
:action => 'CONFIGURE', :dynamic => true})
app.defProperty('EXTRA_PROFILE_CONFIGURATION', 'Profile configuration', '',
{:type =>
'parameter', :action => 'CONFIGURE', :dynamic => true})
app.defProperty('EXTRA_PROFILE_SUBID', 'Profile Sub ID', '', {:type =>
'parameter',
:action => 'CONFIGURE', :dynamic => true})
app.defProperty('EXTRA_PROFILE_PEERID', 'Profile Peer ID', '', {:type =>
'parameter',
:action => 'CONFIGURE', :dynamic => true})
app.defProperty('EXTRA_PROFILE_COMMENTS', 'Profile comments', '', {:type =>
'parameter',
:action => 'CONFIGURE', :dynamic => true})
app.defProperty('EXTRA_ENABLED', 'Logging enabled', '', {:type =>
'parameter', :action =>
'LOG', :dynamic => true})
app.platform = 'android'
end
```

The configuration of the properties is shown in the following code:

```
defProperty('network', "true", "Log network information")
defProperty('traffic', "true", "Log network traffic")
defProperty('gps', "true", "Log GPS information")
defProperty('neighbour', "true", "Log neighbour cells information")
defProperty('profile', "false", "Log profile information")
defProperty('profile_scenario', "static", 'Profile scenario ["vehicular",
"static",
"pedestrian", "high-speed"]')
defProperty('profile_tech', "HSPA", 'Profile technology ["GSM", "HSPA",
"LTE", "UMTS",
"WIFI"]')
defProperty('profile_configuration', "Mobile-Fixed", 'Profile configuration
["Fixed-Fixed",
"Mobile-Mobile", "Mobile-Fixed]')
defProperty('profile_subid', "0", "Profile Sub ID")
defProperty('profile_peerid', "0", 'Profile Peer ID ["0", "1", "2"]')
defProperty('profile_comments', "No_comments", "Profile comments (can't be
empty nor contain
blanks)")
defProperty('log', "false", "Logging enabled")
defGroup('Actor', 'one') do |g|
g.addApplication("TestelDroid") do |app|
# Configure the parameters for the TestelDroid application
```

```
app.setProperty('EXTRA_NETWORK', property.network)
app.setProperty('EXTRA_TRAFFIC', property.traffic)
app.setProperty('EXTRA_GPS', property.gps)
app.setProperty('EXTRA_NEIGHBOUR', property.neighbour)
app.setProperty('EXTRA_PROFILE', property.profile)
app.setProperty('EXTRA_PROFILE_SCENARIO', property.profile_scenario)
app.setProperty('EXTRA_PROFILE_TECH', property.profile_tech)
app.setProperty('EXTRA_PROFILE_CONFIGURATION',
property.profile_configuration)
app.setProperty('EXTRA_PROFILE_SUBID', property.profile_subid)
app.setProperty('EXTRA_PROFILE_PEERID', property.profile_peerid)
app.setProperty('EXTRA_PROFILE_COMMENTS', property.profile_comments)
app.setProperty('EXTRA_ENABLED', property.log)
end
```

10.3.6.2 Control of T2010 eNodeB emulator

The E2010 from Agilent (now Keysight) is a configurable eNodeB with measurement capabilities. In order to automatize tests it provides a Standard Commands for Programmable Instruments (SCPI) interface which is used by the RC to write to and read from the device. Almost every parameter of the E2010 can be configured using OEDL. A complete OEDL script with all the available parameters is available at the EC of the testbed.

10.3.7 As our RC is taylored to the devices of our testbed we need some extra information in the OEDL script to signal various events. We address them in the documentation available at: http://www.morse.uma.es/performnetworks.

11 C-Lab

11.1 Facility description

Community-Lab is an open, distributed infrastructure for researchers to experiment with community networks. The goal of Community-Lab is to advance research and empower society by understanding and removing obstacles for these networks and services. It was developed by the FP7 CONFINE project (Community Networks Test bed for the Future Internet, FP7 Integrated Project 2011-2015). The goal of the test bed is to supporting experimentally driven research on community networks. To achieve this goal, the CONFINE test bed integrates with and extends six existing community networks: Guifi.net (Catalonia in Spain), FunkFeuer (Vienna and Graz in Austria), AWMN (Athens in Greece), Sarantaporo.gr (Sarantaporo and 15 villages in Greece), Ninux (Several cities in Italy), Wireless België (several cities in Belgium). Community-Lab is an open, distributed infrastructure where researchers can select among more than 125 nodes, create a set of containers (a slice), deploy experimental services, perform experiments or access open data traces.

Community-Lab defines two different types of devices to ease the management of the test bed while being part of a community network:

- Research devices or just nodes are the devices deployed by CONFINE that run the applications.
- Community devices are devices connected to the community network, participating on every mechanism required by it (routing protocols, database registration, etc.).

Research devices are connected to the community network through a community device, but they do not have to adapt to the requirements of the community network. Some research devices also have additional wireless network cards for experiments, but many experiment just use the underlying community network for experiments.

Test bed nodes can run applications concurrently. This is achieved by means of Linux Containers (LXC): every application will indicate a set of nodes where it wants to run, and the research devices will create a LXC container for the application. Using PlanetLab terminology, we call each of those containers sliver and the set of slivers belonging to an application is called slice.

The testbed has native interfaces for resource management, experiment lifecycle and monitoring. These interfaces have been wrapped to provide standard interfaces promoted by Fed4FIRE.

11.2 Basic Functionality

11.2.1 Documentation

The documentation is in the Community-Lab wiki: https://wiki.confine-project.eu/usage:sfawrapper https://wiki.confine-project.eu/oml:start and https://wiki.confine-project.eu/soft:omftutorial

11.2.2 Policies

There are no specific policies due to the usage of standard interfaces. All that applies is the Acceptable Usage Policy of the testbed: https://community-lab.net/2012/09/21/acceptable-usage-policy/

11.2.3 Facility Monitoring (FLS)

C-Lab uses its own monitoring tool (http://monitor.confine-project.eu/) that has been used to provide infrastructure monitoring information about the services, and about the resources used in experiments to the Fed4Fire FLS service by the use of OML streams.

11.2.4 Testbed interface for experimenters

11.2.4.1 SFA AM approach

C-Lab uses SFAv3 and RSpec. Implementation based on the SFAWrapper enhanced during Cycle 2 development.

11.2.4.2 Own Testbed interface approach

The testbed has its own REST API for resource allocation and experiment configuration, another REST API for testbed nodes, and a third interface for monitoring information.

11.2.5 IPv4/IPv6 Connectivity

In C-Lab this is done through interaction with slivers with ssh inside a decentralized VPN based on a TINC (IPv6 based) overlay.

11.3 Additional Enhancements

11.3.1 Infrastructure monitoring

The C-Lab monitor (http://monitor.confine-project.eu/) allows experimenters to use its services to monitor the trend (in terms of resource usage) of their running experiment. The C-Lab monitor collects metrics in a key-value store that can be easily queried for building statistics and graphs. The measurements can also be sent as OML streams using the OMFv6 port for C-Lab.

11.3.2 Advanced reservation

C-Lab does not allow advanced reservation.

11.3.3 SLA

C-Lab does not allow setting SLAs. Experiments are performed in best-effort mode, without guarantees. A different approach is used: Combined with historic monitoring information can be used to associate trustability metrics to different experiment runs.

11.3.4 Reputation

C-Lab does not have specific reputation mechanisms. A different approach is used: Historic monitoring information can be used to associate ratings to resources and participants.

11.3.5 Permanent Storage

Slices can reserve permanent storage in nodes for each sliver.

11.3.6 Experiment Control

It is possible to control its related slivers through an OMF Experiment Controller. OMF support has been upgraded to the latest version 6.0.

11.3.7 Layer 2 connectivity

C-Lab is a layer 3-4 testbed. A TINC (IPv6 based) overlay is provided to remotely join the local testbed network.

12 UltraAccess

12.1 Facility description

The UltraAccess Fed4FIRE testbed integrates optical access network testbed infrastructure of two research groups at Universidad Carlos III de Madrid (Spain) and Stanford University (USA):

• The UC3M access network facility (Figure 5.1) consists of a fully configurable Wavelength Division Multiplexed Passive Optical Network (WDM-PON) testbed, including a WDM-PON OLT, a multipoint fiber infrastructure equipped with an AWG, a set of Optical Network Terminals (ONTs) and high-end Fed4fire stations, providing end-users with a dedicated capacity of up to 1Gb/s. This facility can be used as both a fixed capacity setup to test high-speed multi-service residential experiments and an optical backhaul for a Fed4Fire wireless facility (to be either physically installed as backhaul or integrated remotely).

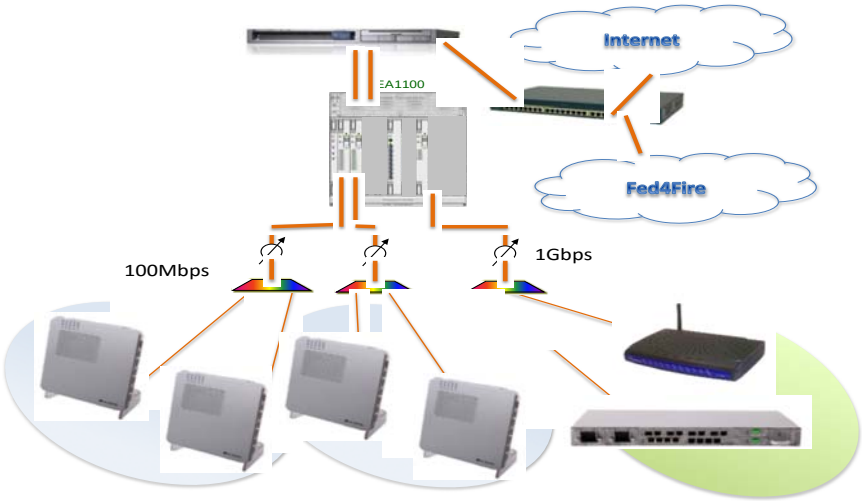


Figure 12.1: UltraAccess-UC3M Facility

• The Stanford University next-generation optical access facility (Figure 12.2) has been developed by the Photonic Network and Research Laboratory (PNRL) group and is part of the renowned NSF research project UltraFlow whose partners are Stanford U., UTD and MIT. The UltraFlow Access Network testbed is the access segment implementation of the concept of Optical Flow Switching (OFS). OFS has been proposed as an alternative to hop-by-hop electronic packet switching for large data transfers, as it can feature very low end-to-end latency in the data transfer phase at ultra-broadband speeds. In OFS, dedicated lightpaths are scheduled and dynamically allocated along the entire network path between end-systems. In UltraAccessFed4Fire the optically switched infrastructure provides 10Gb/s end-to-end connectivity between high-end Fed4Fire stations and a content server over a conventional PON fibre. UltraFlow Access transfers are fully compatible with standard GPON topology and are intended to open new business models to further exploit fibre capacity by Future Internet services demanding ultrabroadband rates. Protocol stacks and software tools to squeeze the full 10Gb/s transfer capacity are available for experimentation.

Required parameters are missing or incorrect.

Figure 12.2: UltraAccess-Stanford Facility

12.2 Basic Functionality

12.2.1 Documentation

The description of the testbed can be found at: http://www.fed4fire.eu/ultraaccess/

A tutorial for end users of the UltraAccess testbed at UC3M can be found at:

http://adscom.it.uc3m.es/ultraaccess/tutorial.html

The VMs accessible to the end users are physically located at the OLT and behind the ONTs.

A tutorial that describes how to use the UltraAccess testbed at Stanford, also named UltraFlow Access, was presented at ICTON 2015 [ICTON2015]. The network configuration deployed by Fed4FIRE at Stanford (made available until 1st of January 2016) is described in Figure 12.3.

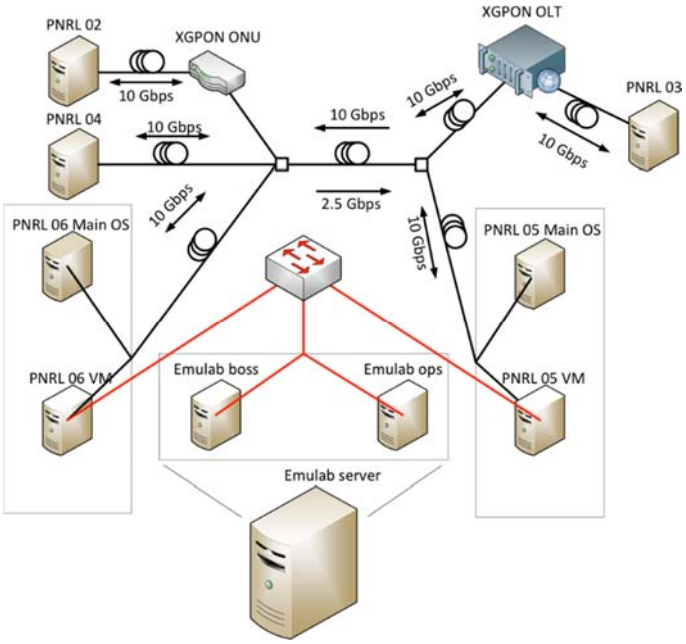


Figure 12.3: Stanford's UltraFlow Access Testbed scheme and the Federation Equipment

12.2.2 Policies

Open to experimentation to any researcher or enterprise that meets the eligibility conditions of Fed4FIRE, after previous request and approval.

12.2.3 Facility Monitoring (FLS)

A Python script to report the facility status by means of OML to the FLS monitor was developed. This script is activated on a regular basis to send an OML stream that indicates the status of the two main components of our testbed, Emulab Boss and Ops machines, to the Fed4Fire FLS dashboard:

https://flsmonitor.fed4fire.eu/. The "GetVersion" field indicates whether the relevant aggregate manager (AM) can be contacted (green color) or not (red color). The numeric value of the "Free Resources" field does not provide meaningful information since the resources at the testbed are virtualized and sliced for experimenters.

12.2.4 Testbed interface for experimenters

12.2.4.1 SFA AM approach

SFA GENI v3 is supported through the use of emulab in the set of PCs co-located with each device of the testbed.

12.2.4.2 Own Testbed interface approach

No specific API for access to experiment besides SFA was planned.

12.2.5 IPv4/IPv6 Connectivity

The UltraAccess testbed nodes are available through SSH. UltraAccess-UC3M testbed has public access via SSH in both IPv4 and IPv6. At UltraAccess-Stanford the access goes through NAT and only IPv4 is supported due to Stanford network policies. In both cases, direct node access through the jFed experimenter tool is supported.

Support for GRE tunnels is provided for experimenters demand to support virtual L2 connectivity which means that IPv6 experiments could also be run, although not natively.

12.3 Additional Enhancements

12.3.1 Infrastructure monitoring

Infrastructure monitoring is performed in two steps. First, resource availability is checked in the Emulab database and sent on an OML stream by means of a python script. Second, there is a collectd daemon running at every UltraAccess resource, which collects system performance statistics in a periodic manner by means of different plugins. Among the monitored resources of the testbed elements there is: CPU, disk, interfaces, memory, processes, uptime and users. Experimenters can customise it as needed. UltraAccess resources have also been provided with collectd-write-oml2 so that, instead of being stored in a local file, measurements can be reported to an OML database. Both experiment data recording at the UltraAccess local OML SQLite database and at the Fed4Fire OML database are supported.

12.3.2 Advanced reservation

UltraAccess testbed does not offer in-advance reservations. Resources are provided when requested (if available) and are reserved for the experimenter until the expiration date defined in the request or when the experimenter releases them.

12.3.3 SLA

No SLA supervision service was foreseen. Thus there is no integration with SLA Dashboard. However, infrastructure monitoring data sent to the OML database provides information about the status of the

components of the facility (servers and switches). With this monitoring data it is possible to evaluate the availability of the testbed resources and thus define SLAs based on it.

12.3.4 Reputation

Not in workplan.

12.3.5 Permanent Storage

1TB hard disks are available in the servers. Users need to copy their data before releasing the service.

12.3.6 Experiment Control

For most experiments, users will gain access and open virtual machines through JFED and run their own measurement tools from the root consoles. However, experiment control in UltraAccess can also be performed through OMF. This framework basically requires an experiment controller (EC) that processes an OEDL description of the experiment scenario and controls the required nodes and a resource controller (RC) daemon at every resource, which receives messages from the EC and executes the commands. Messaging between the EC and the RCs is performed via an XMPP server or an AMQP server. In UltraAccess, an AMQP server local to the testbed site infrastructure is provided for both testbed sites. Both the EC and RC functionality are included in our recommended OS image for Emulab nodes so that the experimenter is able to choose freely which node in the testbed is to take the role of EC or RC.

Regarding monitoring, UltraAccess Fed4FIRE testbed provides experiment monitoring by means of OML. The first entity involved is an OML client library that essentially provides a C API to be used inside applications, but there are also native implementations in Python (OML4Py) and Ruby (OML4R) as well as third party libraries for Java (OML4J) and Javascript/WebSocket (OML4JS). It allows the experimenter to define what are called measurement points (MPs) inside new or pre-existing applications. The MPs generate measurement streams (MSs) that can be directed to the second entity, a central data collection point that in our case is a local OML SQLite3 database server responsible for collecting and storing measurements. Experimenters may alternatively use any other OML database of their choice.

12.3.7 Layer 2 connectivity

No specific native layer-2 connectivity is available in this testbed. Layer-2 connectivity for testbed interconnection is provided with GRE tunnels. Scripts to set up these tunnels with Virtual Wall or other federated testbeds were developed and are available to experimenters.

[ICTON2015] D. Larrabeiti, L. Kazovsky, G. Rodríguez, R. Aparicio, T. Shunrong Shen and S. Yin, "Integrating a next-generation optical access network testbed into a large-scale virtual research testbed," *Transparent Optical Networks (ICTON), 2015 17th International Conference on*, Budapest, 2015, pp. 1-6. doi: 10.1109/ICTON.2015.7193424

13 LOG-a-TEC

13.1 Facility description

The LOG-a-TEC testbed, primarily aimed at Cognitive Radio and Networking experimentation, as it is depicted in Figure 13.1, consists of several clusters of permanently mounted VESNA sensor nodes that are dedicated to experimentation with spectrum sensing and radio communications within wireless sensor networks. Each sensor node in these clusters is equipped with multiple reconfigurable radio interfaces that can be used in various modes. In addition to experimentation in frequency bands for unlicensed devices, the national radio spectrum regulation also allows for experimentation in TV white spaces.

The testbed is remotely accessible over the Internet and uses a dedicated wireless management network to control individual sensor nodes in a cluster. Different approaches can be used to perform experiments, depending on the latency requirements and complexity of experimental scenarios: from high-level control using Python or graphical network stack composition to reprogramming the nodes with native applications. Radio propagation modeling tools can be used as well to plan the experiments.

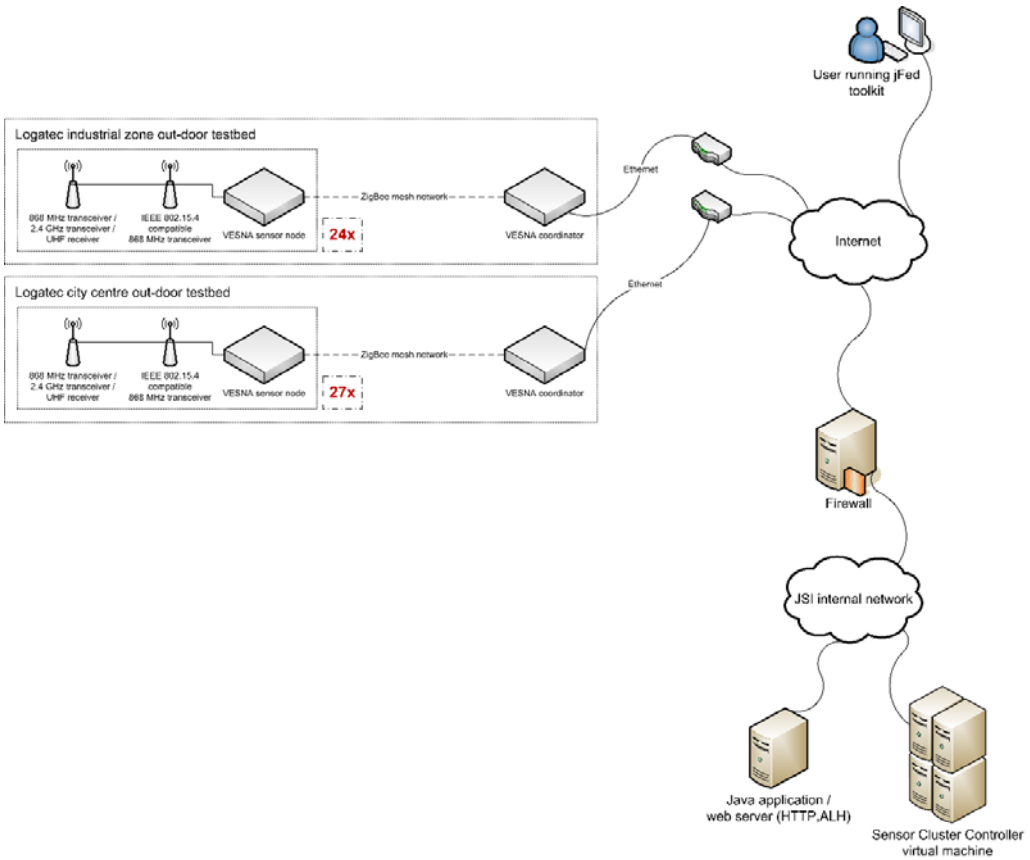


Figure 13.1 LOG-a-TEC testbed architecture.

13.1.1 VESNA device architecture

VESNA sensor nodes contain an ARM Cortex M3 CPU with 64 MHz clock, 512 kB Flash ROM, 64 kB RAM and a 2 GB SD card for code and data storage. Nodes are accessible over a non-IP based proprietary protocol based on ZigBee. ZigBee based nodes can be accessed from the IP network through a Java application running on an internal server which provides a protocol translation.

Each node has 3 hardware slots that can contain up to 3 radio modules: (a) a reconfigurable transceiver or receiver (see below), (b) Atmel AT86RF212 or AT86RF231, an IEEE 802.15.4 compatible transceiver and (c) an 868 MHz ZigBee radio.

Slot (a) can contain one of: (a1) a Texas Instruments CC1101-based sub-1 GHz reconfigurable transceiver, (a2) a Texas Instruments CC2500-based 2.4 GHz reconfigurable transceiver or (a3) a custom designed spectrum sensing receiver for the UHF broadcast band.

On the ZigBee-based nodes, radios (a) and (b) can be used for experimentation.

Sensor nodes are controlled through an HTTP-based protocol from a Linux virtual machine. A Python library is available that provides a simple high-level interface for experiment control.

13.1.2 Experimentation

Based on the requirements for the experiment (types of radios required, desired environment and node geometry, etc.) the experimenter first needs to choose a testbed cluster and a subset of sensor nodes in that cluster that will participate in the experiment.

Next, the experimenter needs to provide a program that configures the sensor nodes, controls the experiment and saves the desired measurements. The program communicates with the sensor nodes through a REST API. The program communicates with the nodes over a Java-based application that acts as a gateway.

The program is typically written in Python language because a Python module is provided that provides simple, high-level functions to control the sensor nodes. The vesna-alh-tools utility provides an abstraction over the protocol gateway and also offer some debugging tools that can be used during the development.

For more information visit: http://www.log-a-tec.eu/overview.html

13.2 Basic Functionality

13.2.1 Documentation

Detailed documentation is available on the LOG-a-TEC web page:

- The LOG-a-TEC testbed overview, http://log-a-tec.eu/overview.html
- Fed4FIRE LOG-a-TEC Tutorial, http://log-a-tec.eu/cr-etel-tutorial.html
- URLs, URNs and RSpecs , http://log-a-tec.eu/cr-etel-rspecs.html
- Additional documentation describing our solution of management protocol and modular protocol stack can be found at http://log-a-tec.eu/software.html

13.2.2 Policies

Fed4FIRE federation SFA-based usage is currently open to all users with Fed4FIRE certificates and passwords, using the jFed Experimenter (GUI) tool and its integrated SSH terminal login.

At the time of writing this document no advance reservation is integrated yet and the users are accepted on the first come / first served basis. Based on the actual usage, this access might be limited in the future, requiring a user to make an advanced reservation either by contacting the LOG-a-TEC testbed administrators via email, or possibly via automatic procedures that are planned to be integrated as a part of the existing jFed Experimenter tool.

Besides the SFA approach, the LOG-a-TEC testbed can still be operated remotely also through the LOG-a-TEC web portal. In order to get such access to the LOG-a-TEC testbed, a user first needs to acquire a user account from the LOG-a-TEC administrators. More details are available on the LOG-a-TEC webpage.

13.2.3 Facility Monitoring (FLS)

Facility Monitoring for FLS (First Level Support) sends data about the status of the LOG-a-TEC testbed to the Fed4FIRE OML server every 10 minutes. The following measurements are sent:

- **Time stamp** [data type string] in the ISO 8601-like format, compatible with the Ruby "Time.now.to_s", e.g.: 2015-11-09 15:09:40 +01:00
- **IPv4 ping check of the main LOG-a-TEC server** [data type uint32, the value is 1 if status is OK, and 0 otherwise]
- **HTTP server check (IPv4) of the main LOG-a-TEC server** [data type uint32, the value is 1 if status is OK, and 0 otherwise]
- **IPv4 ping check of the SFA server** that run SFA software and user SSH sessions (data type uint32, the value is 1 if status is OK, and 0 otherwise]
- **IPv6 ping check of the SFA (F4F) server** [data type uint32, the value is 1 if status is OK, and 0 otherwise)
- **AM port status** - check whether the Aggregate Manager port (12346) is opened on the SFA server [data type uint32, the value is 1 if status is OK, and 0 otherwise]
- **SSH port status** - check whether the SSH port (22) is opened on the SFA server [data type uint32, the value is 1 if status is OK, and 0 otherwise)
- **General status** - [RGA /Red/Green/Amber] proposed based of the above measurements:
    - 0 [green]: all the above checks have succeeded (including IPv6)
    - 1 [amber]: all IPv4 checks have succeeded, but not IPv6
    - 2 [red]: otherwise

13.2.4 Testbed interface for experimenters

13.2.4.1 SFA AM approach

The detailed tutorial is available at http://log-a-tec.eu/cr-etel-tutorial.html. The user runs the jFed Experimenter GUI, logs in with his/her Fed4FIRE certificate and password, and selects a SCC node (Sensor Cluster Controller) for the chosen sensor cluster. SCC node is implemented as a Linux (Ubuntu 14.04) user account. After performing SSH terminal login to SCC (integrated in the jFed Experimenter GUI), the user can run experiments by creating and executing custom-written Python scripts, using the LOG-a-TEC testbed's Python API library. OMF can also be used to control experiments, although for the LOG-a-TEC sensor cluster testbed it does not bring any benefits and is discouraged. Additionally, for security reasons, remote use of OMF is currently disabled. Direct use of Python scripts is the preferred and recommended way of performing experiments. The results generated by an experiment can be written to a file which can be later transferred to a remote (e.g. user's) location with standard Linux tools (such as SCP), or the installed Python OML library can be used by the Python script to stream the results directly to a remote OML server.

13.2.4.2 Own Testbed interface approach

For the purpose of communication between sensor network and the end-user located at JSI (infrastructure side) we developed a new protocol (see Figure 13.2), which was inspired by the HTTP protocol and is simple enough for fast implementation on VESNA nodes. The protocol is designed as a client-server protocol. In our case the servers are sensor nodes and the client is the server on the infrastructure side. Before we can access the resources, the VESNA gateway has to establish a connection with the infrastructure side. This is done by establishing a secure SSL encrypted socket with the server. Once the connection has been established by performing HTTPS requests, the end-user could access any resource (sensor, radio module, etc.) or procedure on any of the nodes.

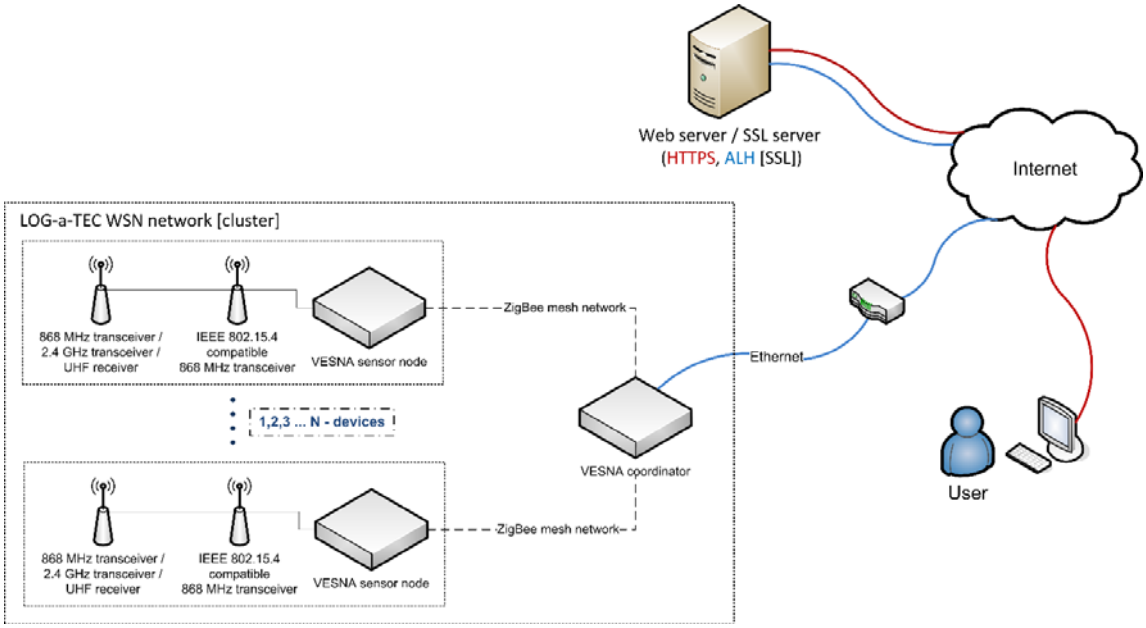


Figure 13.2 LOG-a-TEC infrastructure approach.

Access via LOG-a-TEC web portal

The testbed can be operated remotely through the LOG-a-TEC web portal. The user can select a cluster of VESNA nodes and configure them to perform sensing and/or transmission. As a result, the testbed is able to support sensing-only experiments, transmission-only experiments, and also transmission based on sensing results. The LOG-a-TEC web portal also uses the GRASS-RaPlaT tool in order to (i) provide the virtual experiment planning via simulation in order to ascertain the best setup before the actual execution in the testbed, as well as (ii) support the post processing and visualization of experimentation results.

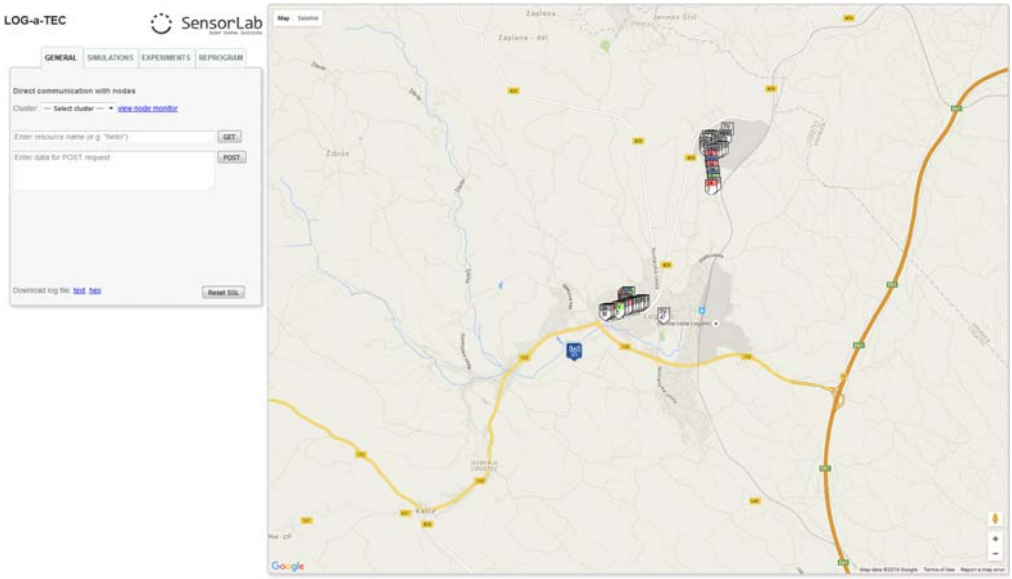


Figure 13.3 LOG-a-TEC webapp

The web app, depicted in Figure 13.3, is split into two parts. The right hand side of the graphical user interface (GUI) includes a map where one can observe locations of the nodes and, based on the color code, distinguish between their roles in the network (e.g. gateway, ISM band sensing node, TV band sensing node, etc.). The left hand side of GUI is used for the interaction between the user and the testbed. Features in this part of the web app include:

- **Choosing one of the available clusters:** either at the JSI campus or in the city of Logatec. Each cluster needs its own SSL server listening on the infrastructure side, and by changing the cluster we make a switch to a different SSL server.
- **Experiment description:** The experiment can be described in a simple text file in which we specify the GET and POST requests, which define the experiment. The commands are separated by empty lines.
- **Logging support:** All the requests and responses are stored in the request-response log file, where the spectrum sensing data is also collected in the format corresponding to the CREW common data format [ref to CREW CDF].
- **GET and POST request fields**: The web app also provides an option to send a single manually configured GET or POST request to the sensor network.

- **Remote reprogramming support**: For reprogramming the nodes, there is an option to select a binary file from a user's local environment and upload it to the server where a server side application is used to send the file to the WSN gateway.

Access via exposed HTTP API

The alternative way to access the LOG-a-TEC testbed, mainly supported for the advanced users and developers, is by calling the HTTP API. The call to the API has to meet the specified form for GET and POST requests:

https://crn.log-a-tec.eu/communicator?cluster="port"&method=get&resource="resource"

To make a GET or POST request, we have to make a call to a handler called "communicator", which is located on the web server on the LOG-a-TEC infrastructure with the domain name crn.log-a-tec.eu. The call is made over a secure HTTPS encrypted socket requiring authentication and has to include the following parameters:

The method: The method can be either GET or POST.

Resource: The last parameter in the case of GET request is the resource. This corresponds to the resource name located on the target node from one of the clusters.

Content: The POST request includes also content where we specify the reconfiguration parameters for the nodes.

13.2.5 IPv4/IPv6 Connectivity

The LOG-a-TEC testbed is reachable with both the IPv4 and IPv6 protocols directly without proxies, using public IP addresses. This includes the following nodes:

- SFA server – AM v3, https://log-a-tec.ijs.si:12346
- SSC (Sensor Cluster Controller) nodes accessible via SSH, lgt-city.log-a-tec.ijs.si and lgt-industrial.log-a-tec.ijs.si (automatic key-based ssh login is allowed only to the user to whom the particular sensor cluster has been allocated and provided).
- WWW server with on-line documentation, log-a-tec.eu (reachable via the link on the Fed4Fire LOG-a-Tec web page (http://www.fed4fire.eu/log-a-tec-jsi/)

13.3 Additional Enhancements

13.3.1 Infrastructure monitoring

The LOG-a-TEC testbed provides WSN experimental clusters consisting of constrained devices such as sensor nodes, which have limited processing and communication capabilities. Therefore, implementation of infrastructure monitoring of such devices requires additional low-level software development and there is no straightforward approach to support this feature. Currently, in addition to the Fed4FIRE FLS facility monitoring, LOG-a-TEC provides partial infrastructure monitoring via Munin, which is a network resource monitoring tool.

13.3.2 Advanced reservation

Advanced reservation is not yet implemented and is not offered as part of LOG-a-TEC testbed. The resources are allocated and provisioned on the best effort, first come / first served basis to all eligible users, which are currently all users with the jFed Experimenter Tool account. Based on the actual testbed usage, limitations might be introduced in the future, requiring some advanced reservation, either manual (by contacting the testbed administrators to get access and a time slot) or automatic using the Experimenter Tool.

13.3.3 SLA

SLA (Service Level Agreement) is not planned to be adopted in LOG-a-TEC testbed and there is no integration with the SLA dashboard; resource reservation is provided with best effort approach.

13.3.4 Reputation

Reputation is in large part based on the infrastructure monitoring, which is not adopted for the reason described above. For the same reasons, reputation service is not supported by the LOG-a-TEC testbed.

13.3.5 Permanent Storage

This is not applicable for the LOG-a-TEC sensor clusters testbed.

13.3.6 Experiment Control

OMF is installed and supported for experiment control, although currently the remote access is not allowed due to certain security consideration, and only local usage is possible. However, for the experiments with a LOG-a-TEC testbed sensor cluster, OMF does not bring any advantage and its use is discouraged. The testbed functionality is accessible via its Python API library, and experiments are performed by running custom-written Python scripts employing this library. OMF, if used, acts only as a wrapper around these scripts. The preferred and recommended way of performing experiments is by running this script directly in the SCC terminal session. To save testbed reservation time, scripts can be prepared in advance and transferred to SCC by standard Linux tools (e.g. SCP). The results generated by an experiment can be written to a file which can be later transferred to a remote (user's) location with standard Linux tools (e.g. SCP), or the installed Python OML library can be used to stream the results directly to a remote OML server.

13.3.7 Layer 2 connectivity

Layer 2 connectivity is not relevant for the LOG-a-TEC sensor clusters testbed and is not supported.

14 10GTraceTester

14.1 Facility description

10GTrace-Tester is a testbed that allows sending a given traffic trace at any speed up to 10 Gb/s line rate to a workstation that features the device under test (DUT, e.g a software router or a piece of SDN).

The tesbed's traffic player, left part of *Figure* 14.***4***, consists in a low-level FPGA-based (XGMII) design on a Xilinx Virtex-5 board capable of playing back and recording up to 1 TB of traffic at 10 Gb/s using an array of SSD disks. The FPGA is plugged into a server with a Xeon E5-2630 processors running at 2.6 Ghz and 8 GB of DDR3 RAM. Such server runs a HTTP service that receives the instructions to reproduce traffic as DTU desires instead of more complicated remote-SSH commands between player and receiver part of the testbed.

10GTrace-Tester offers a trace repository, the leftmost part of *Figure* 14.***4***, to fed DUT together with others with any normalized pcap that can be downloaded from well-known repositories such as CAIDA. Moreover, "handmade" traces allow users to test abnormal and challenging conditions such as bad FCS or minimal interframe gaps among others.

The receiver workstation, at the right of *Figure* 14.***4***, equipped with a Dual-processor server with two Xeon E5-2630 processors running at 2.6 Ghz and 32 GB of DDR3 RAM with a Intel 82599 card connected to a PCIe Gen3 slot to receive traffic. Receiving traffic is forwarded to user domain be means of a PCIe pass-through channel, such domain is VM launched on demand thanks to the KVM virtualization software.

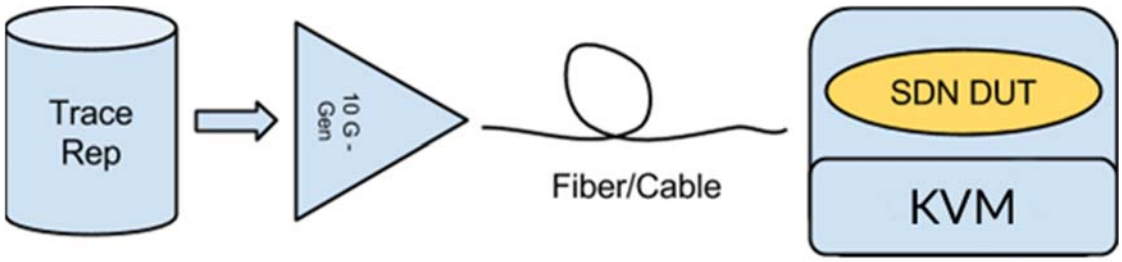


Figure **14**.*4: 10GTraceTester infrastructure*

14.2 Basic Functionality

14.2.1 Documentation

An online documentation and manual for using 10GTraceTester on the federation environment is provided through the testbed website: http://
http://www.eps.uam.es/~jlgarcia/10gtracetester.html. This website provides the necessary information and instructions to run a test and logging in the VM where it is expected users to upload and execute the DUT.

To appropriately handle traffic player, inside user VM it is available a set of typical commands as examples and a readme file that details their utility. Such example scripts serve to list, reproduce, and stop traffic player as well to know its status.

14.2.2 Policies

Users can obtain X.509 certificates from Fed4FIRE identity providers, as explained in 10GTraceTester's website.

14.2.3 Facility Monitoring (FLS)

Facility monitoring has been implemented as a Python script to report the facility status by means of OML to the FLS monitor. This script is activated every 15 minutes to send an OML stream that indicates the status of both the FPGA player and receiving workstation.

14.2.4 Testbed interface for experimenters

14.2.4.1 SFA AM approach

The SFA approach for the aggregate manager follows both GENI v2 and v3 to make it compatible to the maximum number of users. Such users are expected to rely on the Fed4Fire registration tools, such as the iMinds and Fed4Fire portals avoiding the need for registration on our testbed as in the initial phases of its development. Moreover, 10GTraceTester use standard API RSpecs to be as generality as possible to users.

14.2.5 IPv4/IPv6 Connectivity

The VM where users can deploy DUTs is available through SSH via IPv4 protocol.

14.3 Additional Enhancements

14.3.1 Infrastructure monitoring

Infrastructure monitoring checks the correct operation of both traffic player and receiving workstation. Such task is carried out by a Python script that, first, probes FPGA and the server where it is plugged are active. Then, the script checks the receiving workstation. The script executes periodically, with a frequency of every 15 minutes, and injects the result of the tests into OML streams forwarded to federation FLS monitor.

14.3.2 Advanced reservation

10GTraceTester reserves the testbed resources for the interval experimenter requested or by the time the experimenter releases the reserved infrastructure.

14.3.3 Experiment Control

In 10GTraceTester users are provided with a VM with full privileges to deploy the DUT as desired.

To control the input traffic user can send command to FPGA player by means of the HTTP service running in FPGA host. Such commands are wrapped into scripts as follows:

To list all available traces,./list_traces.sh

To reproduce a trace once, ./send_trace <trace.name>

And traffic will reproduce at maximum 10GE rate, which will be received by user VM's interface eth2.

To reproduce a trace in a loop, ./send_trace_in_loop <trace.name>

To stop the traffic player, ./stop

To get the status of the traffic generator, run: ./status

We are currently deploying an OML database at the workstation where the VM will progressively store network-performance measurements gathered when experiments take place. Such measurements will be easily available to experimenters.

15 FAVORITE

15.1 Facility description

We at Trinity College Dublin (TCD), through our telecommunications research centre (CTVR), run a cognitive radio testbed based on the Implementing Radio In Software (Iris) software-defined radio (SDR) system. This testbed supports experimentation with a mature SDR system running on a virtualized computational platform. We have organized this testbed into 16 experimentation units, each of which consisting of three parts: a virtual computational platform, SDR software, and flexible radio frontend hardware. Through this organization we encapsulate the elements required to use the Iris SDR system to construct a broad range of radio systems. Each experimentation unit is designed to flexibly serve a range of needs: Linux (Ubuntu 14.04 LTS) provides a highly configurable computation platform, Iris provides real-time radio reconfigurability, and a Universal Software Radio Peripheral (USRP) offers a broad range of wireless interfaces. Radio hardware is housed on the ceiling of our dedicated indoor testing space to provide users with a clean operating environment. Our management infrastructure allows users to deploy experimentation units to compose arbitrary radio systems and networks as desired.

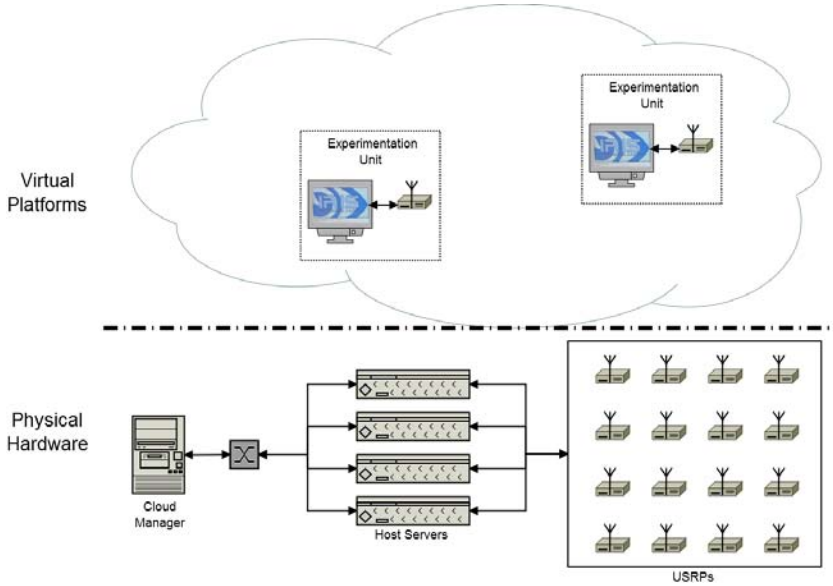


Figure 5 - System Architecture

Figure 1 displays the architecture of our virtualized testbed. In this paradigm, underlying hardware is composed into experimentation units to server users as described above. An array of servers, referred to as host servers, provide the computational power to support running Iris and supporting software with virtual machines. Each virtual machine is connected to a USRP mounted on our ceiling grid within our dedicated testing space. The cloud manager coordinates and controls virtual machines, handling the deployment of computational environments and their connection to USRPs.

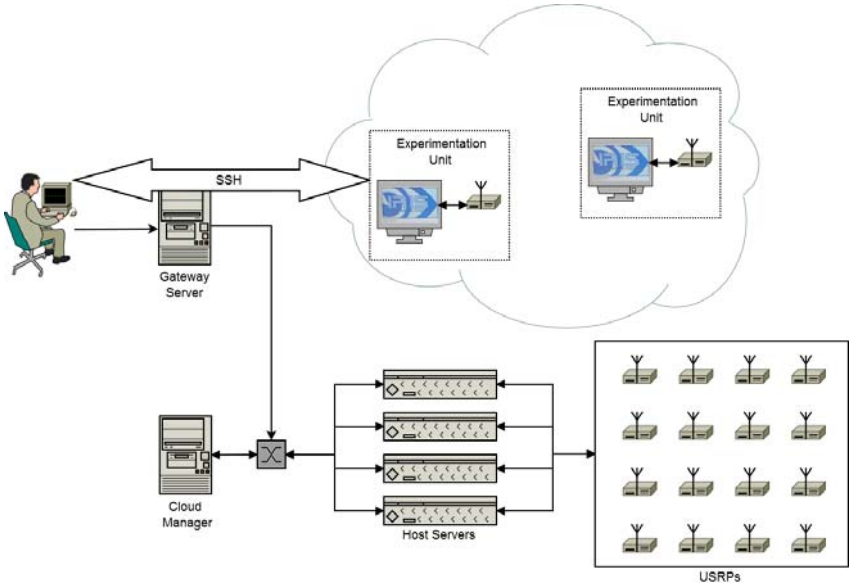


Figure 6 - User Interaction

Figure 2 displays the mechanisms by which users interact with our testbed. Users access a public website, hosted by our gateway server. This website allows the users to configure the experiment to be run, including selection of virtual machine images and USRP positions. Once the experiment is configured, the gateway machine will send a setup request to the cloud manager. The cloud manager

15.2 Basic Functionality

The gateway server internal architecture is depicted in Figure 3. We have developed 4 different components, namely: the Aggregate Manager (AM), the Coordinator, the Reservation web-interface, and the database. The arrows indicate which component can communicate with, e.g., the AM only interacts with the Coordinator internally and with the jFed to the outside world.

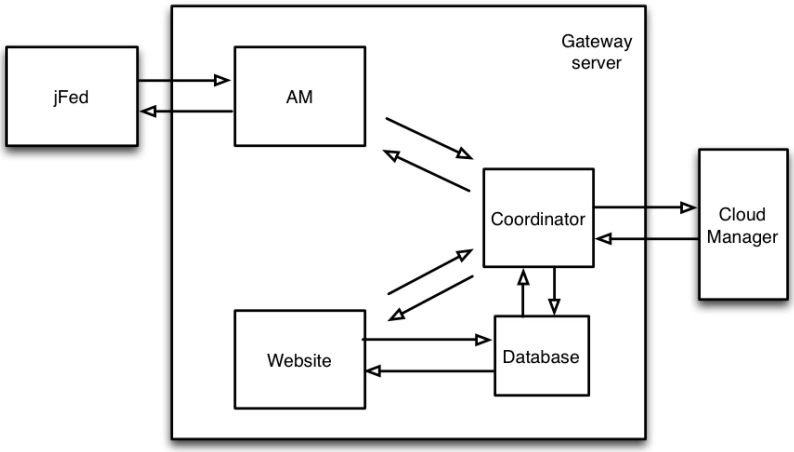


Figure 3 – Gateway Server internal architecture

15.2.1 Documentation

The user's documentation has been divided into two parts. The first concerning the registration of an account with our web portal (described in sec. 15.3.2) can be accessed at (www.ctvr-node13.scss.tcd.ie/reservation/registration.doc). Upon registration and activation of a new account, users can access the second part of the documentation regarding the usage of both the web-portal itself and the FAVORITE testbed.

This documentation is continuously being improved. Therefore, the documentation is an ongoing tasks.

15.2.2 Policies

Each USRP is equipped with a SBX120 daughtercard that allow experimenters to access a wide range of frequencies, from 400 MHz to 4.4 GHz. In order to avoid (unexpected) inter-experiment interference, here at FAVORITE we kindly ask experimenters to advertise the frequency band they intend to use. The frequency can be specified in the Rspec through the jFed, or it can be selected during the reservation through our web-portal. The frequency bands already reserved can be checked on the calendar from our web-portal. FAVORITE testbed does not provide monitoring services of the wireless spectrum, and as such, we kindly require experimenters to respect their bands specification in order to ensure the correct functioning of the testbed.

15.2.3 Facility Monitoring (FLS)

Facility monitoring focuses on determining whether the testbed as a whole is up and running. The permanent resources consists of 16 USRPs placed on a grid in the testbed ceiling and 6 physical hosts provide a total of 120 cores and 256 GB of memory. The site runs the OpenNebula Cloud Manager over the KVM hypervisor. In-depth monitoring information at hypervisor level will be likely made available to the experimenters through one of the monitoring frameworks that are already in place (e.g. Zabbix, Nagios or Collectd) and that already provides OML streams ready to be sent to the to the Federator's central OML server to the Federator's central OML server. Most requested infrastructure metrics are: CPU load, total and free memory, free swap memory, the number of VMs running on a physical node, disk IO, disk read/write, incoming and outgoing traffics on interfaces and energy usage. Currently, this work is under study.

15.2.4 Testbed interface for experimenters

15.2.4.1 SFA AM approach

The AM is the block responsible to interact with the jFed and we implemented it using the delegate mechanism built into the GENI code base as suggested by David Margery. The resulting AM is then essentially a plugin of the reference GENI code base. Below we list the functionalities implemented in the AM and a brief description of the FAVORITE API:

- GetVersion – It returns the Rspec format used at the FAVORITE's AM (currently version 3).
- ListResources – It returns a list with the description of resources at the FAVORITE testbed and their status. This functionality requires first a communication with the Coordinator to fetch the available resource from the database.
- Allocation – It creates a sliver for each resource in a request Rspec received. For each resource it checks with the coordinator whether the resource is available or not for the

entire duration of the experiment requested by the user. If one of the resources is not available (e.g., the selected resource has been reserved for another upcoming experiment) it returns an error message with the resource name unavailable and the maximum time for which it can be used. If successful, the AM extracts for the user credentials all the information relative to the experiment and passes them to the Coordinator:

- node_id – identifier of the resource requested;
- username – username of the fed4fire user.
- email – fed4fire user email.
- start_time – time at which the experiment starts (it is the current time in UTC).
- end_time – time at which the experiment ends (it is set by the user and it is converted in UCT).
- frequency – Advertised frequency band intended to use.
- type – 'fed4fire'.

At this stage, the AM adds the created Sliver(s) to the Slice and return the result to the user.

- Provision: Upon successful allocation, we enter in this function only if the previous allocation was successful. For each sliver (or equivalently, for each resource), the AM triggers a command to the Coordinator to initialize a Virtual Machine and assign to it the requested USRP in the grid by the user. It extracts from the user credentials the public key and passes it to the Coordinator.
- PerformOperationalAction: At this stage the AM only performs standard SFA operation and does not interacts with the coordinator. It returns a manifest Rspec describing the resource reserved setting a status 'geni_configuring'.
- Status: This function periodically checks the current status of all the resource in the Slice. Until all of them are marked 'geni_ready'. This functionality requires interaction with the Coordinator.
- Describe – The AM returns to the user a manifest Rspec with all the updated info necessary to access the resource on the private network. As ssh login is given through a gateway, this function returns all the information to run the ssh ProxyCommand, i.e., the IP address of the gateway (static), the IP address of the resource on the private network (it depends on the IP assigned by the Cloud Manager) and the username that will be used to login. This functionality requires interaction with the Coordinator.
- Renew – Active Slivers can be extended in duration in order to extend the duration of the experiment. If changing the duration of the experiment does not result in any overlap with previously made allocations, then all the resources on the Slice can be renewed. This functionality requires interaction with the Coordinator. The new duration of the experiment and the resource tag name is passed by the AM to the Coordinator.
- Delete – Users can stop an experiment before its scheduled end time. As such this function deletes the Slice, all the resources therein and disables access to the user if he/she does not have any other experiment running. The AM communicates to the coordinator the resources' names and username of the user who issued the delete command.

- ShutDown – This function performs standard SFA operations in case a Slice is not behaving properly.

15.2.4.2 Own Testbed interface approach

FAVORITE facilities can be accessed through our own web portal. For more details, please refer to 0.

15.2.5 IPv4/IPv6 Connectivity

Each experimenter unit runs only on private IPv4 addresses. As such, FAVORITE testbed provides IPv4 connectivity to Fed4FIRE experimenters through secure shell (ssh) from a gateway. Experiments requesting resources will have access to the gateway for the entire duration of the experiment. Once the experiment expires and the user does not have any other experiments active, connectivity will be disabled.

15.3 Additional Enhancements

15.3.1 Infrastructure monitoring

Infrastructure monitoring consists in the instrumentation of infrastructure resources to collect data on the behaviour and performance of resources, services, technologies, and protocols. This allows the experimenters to obtain monitoring information about the used resources that he could not collect himself.

Due to the highly dynamic nature of SDR experimentation that involve a high level of flexibility, we are still defining the metrics that could be relevant to different class of experiments in our testbed. Among the others, experimenters will get fine-grained information on frequency used, free memory, total storage, used storage, free storage, CPU load, CPU utilization (%), CPU count, and other network metrics.

In-depth monitoring information at the resource level will be likely made available to the experimenters through one of the monitoring frameworks that are already in place (e.g. Zabbix, Nagios or Collectd) that provides OML strems.

This task is currently under study.

15.3.2 Advanced reservation

Now we describe the Reservation web-interface that can be used to reserve resources. Through the website, users can make reservations for USRPs and select different images for simulation purposes. The website has been implemented using PHP, JavaScript, jQuery and MySQL. Some of the user management functionalities (i.e., registration, activation and authentication) are done directly on the website, while others (i.e., enabling ssh-login) are done through the coordinator described in Sec. 15.3.3. The resource management functionalities instead (i.e., create reservation, provision, description, status, delete) are done entirely through the coordinator using an internal protocol.

In this document we concentrate on the resource management, implemented mirroring the functionalities implemented by the AM. Upon registration and activation, users can perform the following operations:

- Allocation: First, a user selects the desired experiment starting date, starting time, end date, and end time. Through a query to the allocation table in the database, the website displays only the USRPs available for the time window requested. The user can select the desired resources and the frequency they intend to operate (see Figure 4). At this stage, the following information are passed to the Coordinator, which creates the entry in the database:
    - node_id – identifier of the resource requested;
    - username – username of the logged in user.
    - email – loggedin user email.
    - start_time – time at which the experiment starts (it is the current time in UTC).
    - end_time – time at which the experiment ends (it is set by the user and it is converted in UCT).
    - frequency – Advertised frequency band intended to use.
    - type – 'website'.

The website then schedule an automatic remainder email ½ hour before the beginning of the allocation to the user.

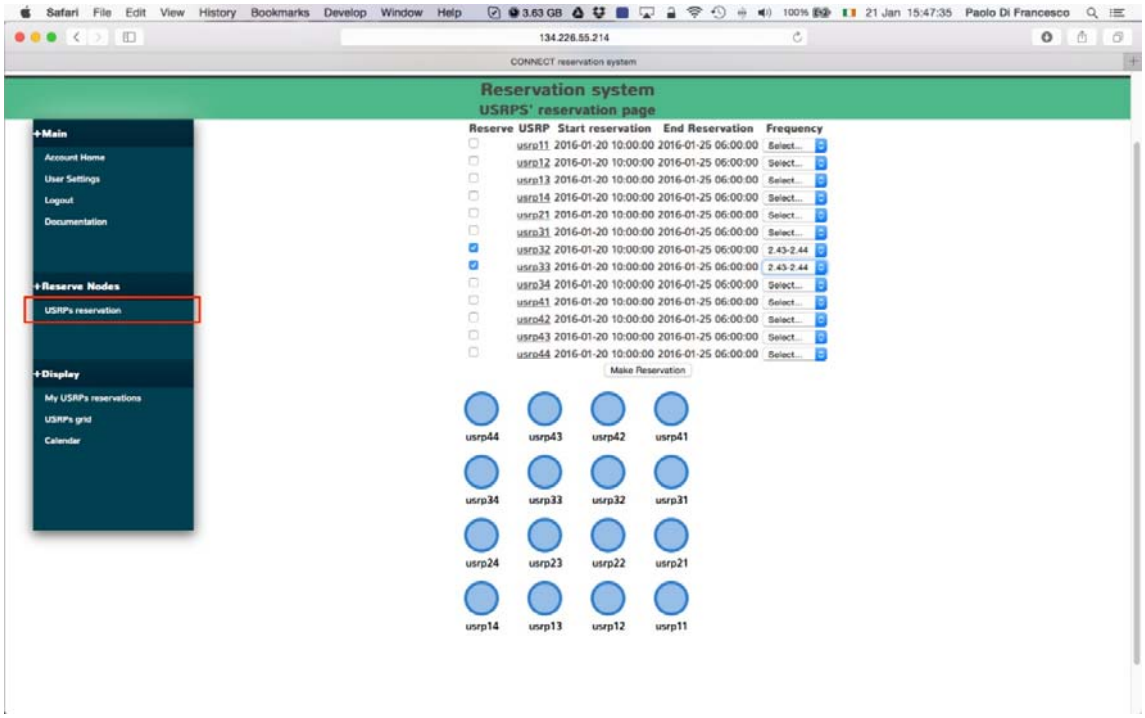


Figure 4 – USRPs reservation after selecting the desired duration of the experiment

- Provision: In the section "My USRP reservation" users can display the list of all the allocation made. The user can provision the resource they reserved only when the present time is in the time window specified by the user. The website triggers a command to the Coordinator to initialize a Virtual Machine and assign to it the requested USRP in the grid by the user. It extracts from the user credentials the public key and passes it to the coordinator (see Figure 5).

Figure 5 – "My reservations": from this menu, users can provision, obtain the private IP address of the resource by checking the Status of the USRP desired and eventually delete the reservation. Expired reservations are automatically deleted from the database and stored for quota check (not implemented yet).

- Status: In the list of allocations, users can check the status of their provisioned resource to check when it can be accessed. If the resource is ready, it returns the command to access the resource on the private network through the gateway using and ssh ProxyCommand (see Figure 5).
- Delete: In the list of allocations, users can delete any reservation previously made (see Figure 5).
- Show Calendar: Based on the jQuery plugin FullCalendar. It displays all the reservation made with the frequency booked. It is intended to be a support for the users making new reservations to avoid to operate on the same frequencies advertised by other users (see Figure 6). Currently we do not support monitoring of the frequency usage. This system is based on gentlemen agreements as explained in sec. 15.2.2.

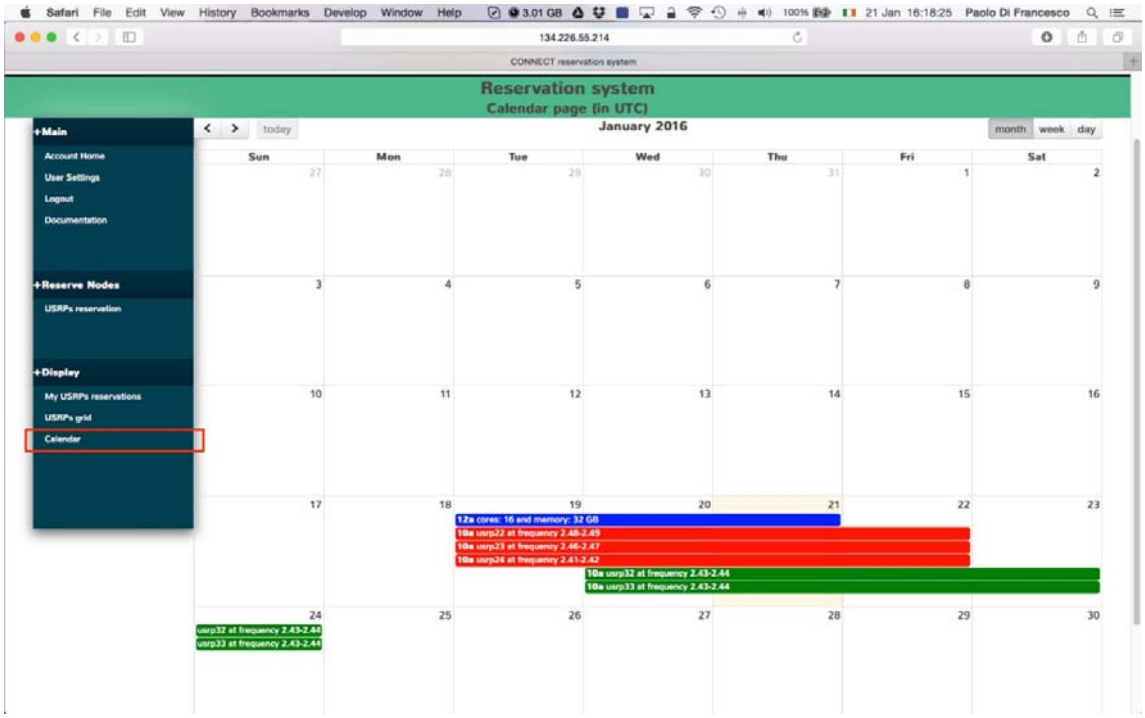


Figure 6 –Calendar: users can check the resources currently booked and the advertised frequencies that are used in order to avoid inter-experiment interference. Reservations of the logged in user are highlighted in green.

15.3.3 Coordinator

Finally here we describe the Coordinator. The Coordinator is the central piece of our design and it implements the common functionalities of the AM and the website. It is responsible to send commands to the Cloud Manager, to create/delete users in the gateway for ssh login, to modify tables in the database. All the communications with the AM and the website are through local TCP sockets. The main functionalities are listed below:

- Create_user (AM and website): it creates users on the gateway enabling access through key pairs.
- Delete_user (AM and website): it removes users on the gateway that do not have resource provisioned.
- List_resource (only AM): it queries the database and returns the list of resource and whether they are available or not at the moment the request is made.
- Allocation (AM and website): it checks if an allocation is possible and it creates an entry in the allocation table with the following parameters:
    - `id` int(11) NOT NULL AUTO_INCREMENT, reservation ID (primary key)
    - `node_id` VARCHAR(10), resource ID
    - `user_name` VARCHAR(50), user ID
    - `email` VARCHAR(50), user email

  - `start_time` DATETIME, start date and time of the reservation
  - `end_time` DATETIME, end date and time of the reservation
  - `frequency` VARCHAR(20), frequency advertised
  - `type` VARCHAR(20), type of reservation (i.e., fed4fire, website)
  - `provisioned` tinyint(1), set to 0 because the reservation hasn't been provisioned yet.
- Provision (both AM and website): It sends a request to the Cloud Manager to initialize the USRP and the associate VM and to grant access to the user specified (it also passes the user's public key). It sets the field 'provisioned' to 1 in the corresponding reservation and it grant access to the user to the gateway to enable ssh login.
- Status (both AM and website): It sends a request to the Cloud Manager which returns the status of the resource (i.e., either 'not_ready', or 'ready').
- Describe (both AM and website): It sends a request to the Cloud Manager which replies with the IP on the private network.
- Renew (only AM): It checks with the database whether the requested resource in the experiment can be extended in time. If successful, it sets the field 'end_time' to the new time in the corresponding reservation entry.
- Delete (both AM and website): It changes the field 'end_time' to the current time in the corresponding reservation entry.
- Check_expired: Periodically checks all the expired reservations and it removes them from the database (every 10 seconds).
- Release: It sends a command to the Cloud Manager to release a resource when it expires.

15.3.4 SLA

Our approach to SLA develop stems directly from the monitoring systems described above. Therefore, we consider the work related to SLA management to comprise both collection of monitoring data and the compression of this data into appropriate metrics. As described above the collection of monitoring data capabilities of the TCD facility are coalescing. Extending beyond this data collection, TCD is examining methods to process this data into consumable metrics. Currently considered metrics include availability of resources (in terms of the ratio of accepted reservations), uptime, and failure rate.

Currently the consortium is still defining the APIs necessary to access the SLA metrics.

15.3.5 Reputation

The reputation system deployment for the TCD facility is currently under study.

15.3.6 Permanent Storage

FAVORITE testbed does not provide permanent storage. Once an experiment is expired, the resource therein gets destroyed with all the data in the VM. However all the monitoring data are kept for reputation assessment.

15.3.7 Experiment Control

FAVORITE intends to support control of the experiment through one of the Fed4FIRE supported experiment control tools, OMF (v6). In order to control an experiment, the experimenter needs 3 parts of the OMF framework: Resource Controller (RC), the Experimenter Controller (EC) and either XMPP or AMQP servers to send FRCP messages between EC and RC(s).

In order to control an experiment, we will supply the OMF Resource Control (RC) pre-installed in the default image of the VMs, while for the rest of the components we are still discussing implementation details.

15.3.8 Layer 2 connectivity

TCD has no workload load on regarding this task, since, due to its capabilities and campus networking design, it is not feasible to provide such feature.

16 PL-LAB

16.1 Facility description

PL-LAB is a distributed laboratory for Future Internet experimentation built of eight laboratories interconnected over the Polish NREN – PIONIER (Figure 16.1). The major focus of experiments performed so far in PL-LAB was on how to enable Parallel Internet paradigms in future networks. The clean-slate approach has been proposed and validated in the testing infrastructure. The equipment available in PL-LAB creates opportunities for such experiments, allowing an implementation of low-level network functions in network processors and net-fpga boards.

The laboratories are geographically dispersed, and associated with leading Polish research and academic centers (Warsaw University of Technology, Poznan Supercomputing and Networking Center, National Institute of Telecommunications, Gdansk University of Technology, Poznan University of Technology, Wroclaw University of Technology, The Institute of Theoretical and Applied Informatics and AGH University of Science and Technology).

PL-LAB creates point-to-point and multipoint-to-multipoint topologies with dedicated capacity between particular laboratory equipment (e.g. servers, routers, programmable switching devices, measurement devices) located in different laboratories. Such connections constituting a PL-LAB User Laboratory, are isolated from both non-PL-LAB traffic, and other simultaneously running PL-LAB experiments. The usage of the Polish NREN high capacity network infrastructure, as the backbone network, guarantees high bandwidth and a scalable environment which can merge eight laboratories into one distributed environment. The independence from Internet infrastructure gives an advanced traffic policy management, flow controls and reliability for inter-laboratory connections.

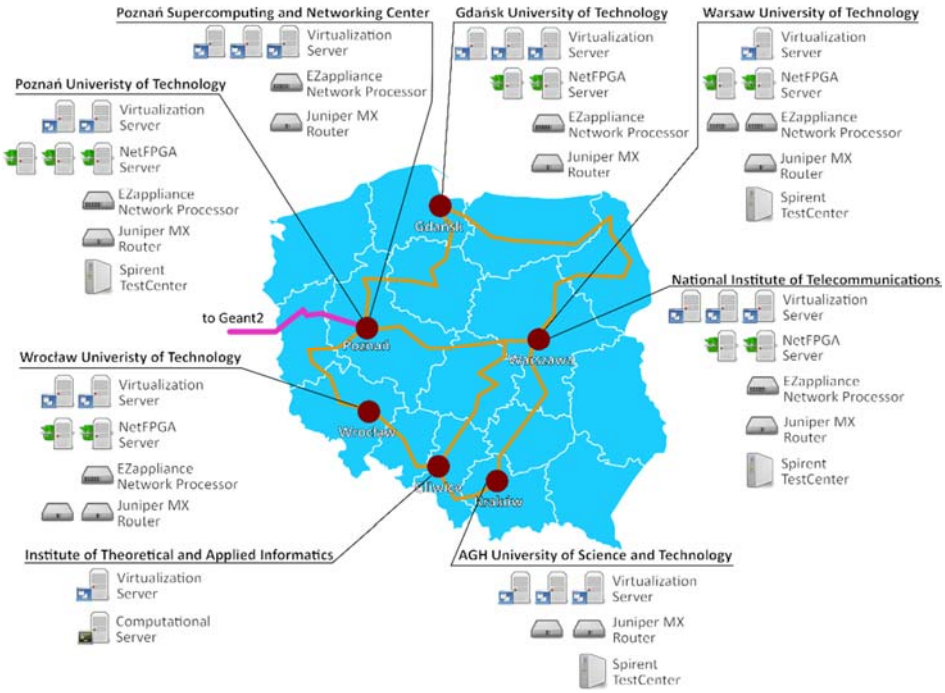


Figure 16.1: PL-LAB Infrastructure

The hardware available in PL-LAB is categorized and listed in the following table.

Servers	• 18 servers (HP ProLiant DL360)
Programmable switching devices	• 7 platforms with programmable network processor NP-3 EZappliance (EZchip) • 11 servers with NetFPGA cards (Virtex2/Virtex5)
Measurement devices	• 4 hardware-based traffic generators/measurement devices (Spirent)
Routers	• 9 Juniper MX80/240

PL-LAB resources will be shared between PL-LAB users registered in the original PL-LAB Portal and the ones with Fed4FIRE credentials using the new GENI SFA interface implementation. The actual number of available equipment will vary depending on the currently active reservations (running experiments) and other aspects like maintenance in any of the PL-LAB nodes.

PL-LAB Management System, named PL-LAB Access System (Figure 5.2), supports researchers performing experiments in PL-LAB network. Its main functionalities comprise user accounts management, experiment requests handling, automatic configuration of dedicated virtual experiment networks, enabling user access to reserved resources and infrastructure monitoring.

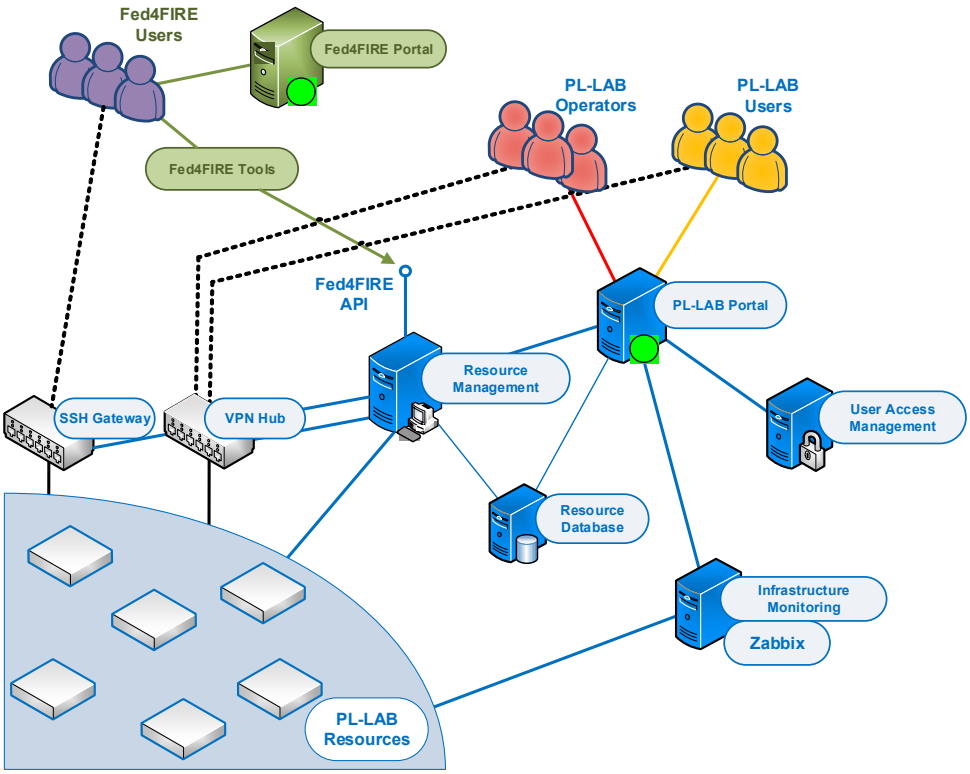


Figure 16.2: PL-LAB Access System architecture

16.2 Basic Functionality

PL-LAB Access System has been extended in order to fully integrate PL-LAB facility with the Fed4FIRE federation. FITeagle (http://fiteagle.github.io) has been used as the base software for PL-LAB Aggregate Manager (GENI v3) implementation. FITeagle provides own SFA interface implementation and offers extensible architecture allowing for integration of custom modules.

16.2.1 Documentation

Basic PL-LAB testbed description is available in the common directory at http://www.fed4fire.eu/pl-lab. Moreover a dedicated website maintained by PL-LAB itself has been setup at http://www.pllab.pl/fed4fire to include both general information on the testbed an also guidelines for the Fed4FIRE users on how to use the facility to run experiments.

A tutorial for basic experiment setup and execution in PL-LAB still needs to be finalized and will be made available at http://www.pllab.pl/fed4fire/tutorial by the end of February 2016.

16.2.2 Policies

FITeagle has been configured to support existing Fed4FIRE Identity Providers and user AA mechanism based on X.509 certificates.

The PL-LAB Access System has been enhanced to support experiment requests from Fed4FIRE users. This effort involved adaptation of the access policies and quota management mechanism. Fed4FIRE users are by default configured with a set of constraints on the reservation size and duration. Nevertheless a basic experiment can be conducted without the necessity for additional approval. Moreover, user may apply for increased quota by contacting the PL-LAB administrators.

Two PL-LAB administrators subscribed to the fed4fire-experimenters google group and are ready to provide any support related with experiment execution in the PL-LAB testbed.

16.2.3 Facility Monitoring (FLS)

The original PL-LAB infrastructure monitoring system has been extended to use Zabbix software. Central Zabbix server is hosted on a dedicated virtual machine. Zabbix agents were installed on each physical server, and SNMP service is running on each network switch and router.

A few Ruby scripts exploiting Zabbix API were implemented in order to automate server reconfiguration and retrieve collected monitoring data. One script is responsible for sending internal status of the PL-LAB through the OML stream to the FLS monitoring database. The script pulls information about states (device up/down, ssh up/down) of PL-LAB core nodes from Zabbix DB, calculates aggregated status and sends the result. Script is added to the Linux crontab and runs every 5 minutes.

The deployed PL-LAB Aggregate Manager correctly responds to GetVersion and ListResources calls generated periodically by the FLS. Moreover, the so called Login Test executed by the FLS passes as well what is an indication of the correct PL-LAB AM implementation of the full experiment life cycle.

16.2.4 Testbed interface for experimenters

16.2.4.1 SFA AM approach

Resource descriptions (RSpecs) for resources available in PL-LAB including servers, routers, programmable switching devices (servers with NetFPGA cards and NP-3 EZappliance network processors) and Spirent traffic generator/analyzer has been prepared and made available at http://www.pllab.pl/fed4fire/rspecs.

In order to allow programmable interaction with the PL-LAB Access System a new PL-LAB REST API has been implemented. This REST API provides basic set of methods for PL-LAB testbed resources discovery, experiment network reservations, resources instantiation (including SSH access configuration for Fed4FIRE users) and experiment resources release.

The FITeagle implementation has been customized in order to intermediate in communication between Fed4FIRE tools and the PL-LAB REST API. To achieve this a PL-LAB specific FITeagle Adapter has been implemented and integrated with the custom FITeagle instance. During the course of PL-LAB Adapter implementation several extensions to FITeagle core modules and SFA implementation were requested in order to fully support PL-LAB RSpecs and experiment lifecycle. Finally, PL-LAB AM implementation supports the entire experiment lifecycle.

PL-LAB offers types of resources not available in Fed4FIRE till now. These new types were integrated with the jFed Experimenter GUI application.

16.2.4.2 Own Testbed interface approach

Apart from the new PL-LAB Aggregate Manager, PL-LAB users may interact with the infrastructure using the original web based PL-LAB Portal available at http://portal.pllab.pl. In order to use the Portal a separate account needs to be registered in PL-LAB since the Fed4FIRE credentials won't be respected for login.

16.2.5 IPv4/IPv6 Connectivity

PL-LAB Aggregate Manager endpoint is accessible from the Public Internet (IPv4) at https://am.pllab.pl:8443/sfa/api/am/v3.

A dedicated SSH Proxy was deployed at PL-LAB (gw.pllab.pl) and PL-LAB supports user IPv4 SSH access to resources available on the PL-LAB internal network via the central Proxy.

PL-LAB SSH Proxy is complementary user access solution to existing PL-LAB VPN server. This approach required a modification of user access to resources by enabling additional authorization based on SSH keys. This functionality uses 3 methods depending on the resource capabilities to provide keys to the resources: (1) LDAP server, (2) directly by configuring user account and (3) providing additional SSH gateway directly connected to the resource (isolated from shared access network). The reason for providing these 3 methods of authentication and accessibility is that some of the network equipment does not support e.g. LDAP authentication or expose web interface without any authentication mechanisms so such resource should not be available directly from shared user access private network for all experiments in PL-LAB.

Information about PL-LAB SSH Proxy has been integrated within the Manifest RSpec and is supported by the jFed Experimenter GUI.

```
<services>
   <proxy for="f4fuser@10.134.1.31:22" proxy="user11@gw.pllab.pl:22"/>
   <login authentication="ssh-keys" hostname="gw.pllab.pl" port="22" username="user11"/>
   <login authentication="ssh-keys" hostname="10.134.1.31" port="22" username="f4fuser"/>
</services>
```

16.3 Additional Enhancements

16.3.1 Infrastructure monitoring

PL-LAB prepared an OML wrapper script in Ruby that exploits Zabbix API to retrieve data from Zabbix DB and transform it into OML stream. Another script capable of sending data about input/output traffic on experiment-related network interfaces of core switches to local or remote OML databases is already implemented. We still plan to use currently implemented functionality in order to integrate with other additional Fed4FIRE services related with monitoring.

In addition, PL-LAB prepared scripts for experiment-related Zabbix account creation/removal so user, including the one registered in Fed4FIRE, gets access to the customized Zabbix account (through the web interface) with the monitoring data of all physical nodes (and virtual machines) used within his experiment.

16.3.2 Experiment Control

Currently PL-LAB supports SSH connections to all resources reserved for experiment, giving the experimenter almost unlimited capabilities when it comes to run and control the experiment.

It is planned to provide basic support for experiment control by installing AMQP server and providing images with OMFv6 support by the end of the project.

16.3.3 Layer 2 connectivity

PL-LAB is setting up a 10GE fiber link directly to GEANT Bandwidth on Demand service in PoP located at PSNC. The purpose of this connectivity is mainly to provide shared data plane links for experiments that require external L2 connections to other testbeds in Fed4FIRE or projects.

Although, PL-LAB does not support current inter-testbed L2 connection creation mechanism used in Fed4FIRE, it is planned to integrate with the new mechanism utilizing GEANT BoD service that to our knowledge is currently being implemented within Fed4FIRE.

References

[1] Fed4FIRE Architecture Workpackage D2.1 “First federation architecture”
[2] Fed4FIRE Experiment Lifecycle Management Workpackage D5.1 “Detailed specifications for first cycle ready”
[3] Fed4FIRE Measuring and Monitoring Workpackage D6.1 “Detailed specifications for first cycle ready”
[4] Fed4FIRE Trustworthiness Workpackage D7.1 “Detailed specifications for first cycle ready”
[5] Fed4FIRE Service and Applications WorkPackage “MS4.1 First design specifications for facilities”
[6] Fed4FIRE Trustworthiness Workpackage D7.2 “Detailed specifications for second cycle ready”
[7] Control and Management Framework http://mytestbed.net/projects/omf/
[8] Open Cloud Computing Interface http://www.occi-wg.org
[9] SFAWrap, http://sfawrap.info
[10]GENI Aggregate Manager API Version 2, http://groups.geni.net/geni/wiki/GAPI_AM_API_V2
[11]FI-Lab, The Open Innovation Lab, http://lab.fi-ware.org
[12]OFELIA Project, https://alpha.fp7-ofelia.eu
[13]NOX, http://www.noxrepo.org/
[14]POX, http://www.noxrepo.org/pox/about-pox/
[15]RYU, http://osrg.github.io/ryu/
[16]OCF, https://github.com/fp7-ofelia/ocf