FUTURE INTERNET TESTBEDS
EXPERIMENTATION BETWEEN
BRAZIL AND EUROPE

Grant Agreement No.: 288356 (FP7) and 590022/2011-3 (CNPq)

FIBRE-EU

Future Internet testbeds/experimentation between BRazil and Europe – EU

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

D 4.3- Report on the contributions to the Federation framework

Author: WP4
Revised by: WP3
Due date of the Deliverable: Month 28
Actual submission date: 21/10/2013
Start date of project: June 1st 2011 Duration: 34 months
Version: v.1.0

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

FP7 Grant Agreement No.	288356
Project Name	Future Internet testbeds/experimentation between BRazil and Europe – EU
Document Name	FIBRE-EU D4.3
Document Title	Report on the contributions of the Federation framework
Workpackage	WP4
Authors	Serge Fdida (UPMC) Panayotis Antoniadis (UPMC) Jordan Augé (UPMC) Loïc Baron (UPMC) Thanasis Korakis (UTH) Jose F. de Rezende (UFRJ) Mayur Channegowda (UBristol) Carlos Bermudo (i2CAT) Leonardo Bergesio (i2CAT)
Editor	Jose F. de Rezende (UFRJ)
Reviewers	WP3
Delivery Date	21/10/2013
Version	V1.0

Abstract

The purpose of this deliverable is to provide a report on the selected solution and implemented/customized software for the federation of the facility.

TABLE OF CONTENTS

TABLE OF FIGURES

1 Acronyms

AM	Aggregate Manager
API	Application Programming Interface
CDS	Content Data Server
CH	Clearing House
CN-DS	Communication Networks and/or Distributed System
CNPq	Brazil’s Council for Scientific and Technological Development
CPqD	Telecommunications Research and Development Centre
EC	Experiment Controller
ED	Experiment Description
EGW	Export resource Gateway
EU	European Union
E-GENI	Enterprise GENI
FEDERICA	Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures
FI	Future Internet
FIBRE	Future Internet testbeds / experimentation between Brazil and Europe
FIRE	Future Internet Research & Experimentation
FOAM	Open Flow Aggregate Manager
FP7	Seventh Framework Programme
FPGA	Field Programmable Gate Array
Gbps	Gigabit bits per second
GENI	Global Environment for Network Innovations
GID	GENI Identifier
GMPLS	Generalized MultiProtocol Label Switching
GW	Gateway
HTTP	HyperText Transfer Protocol
ICT	Information and Communication Technologies
IGW	Import Gateway (Aggregate Manager)
IP	Internet Protocol
I&M	Instrumentation and Measurements
LDAP	Lightweight Directory Access Protocol
MS	Milestone
NICTA	Australia’s Information Communications Technology Research Centre
NITOS	Network Implementation Testbed using Open Source platforms
NOC	Network Operations Centre
NTP	Network Time Protocol
OAR	Versatile Resource and Task Manager
OCF	OFELIA Control Framework
OEDL	OMF Experiment Description Language
OF	OpenFlow

OFELIA	OpenFlow in Europe: Linking Infrastructure and Applications
OGF	Open Grid Forum
OMF	cOntrol, Management and Measurement Framework
OML	ORBIT Measurement Library
ORBIT	Open-Access Research Testbed for Next-Generation Wireless Networks
ORCA	Open Resource Control Architecture
OS	Operation System
PLE	PlanetLab Europe
R	Repository
RC	Resource Controller
REST	Representational State Transfer
RFC	Request for Comments
RM	Resource Manager
RNP	National Research and Education Network
RSpec	Resource Specification
SensLab	Very Large Scale Open Wireless Sensor Network Testbed
SFA	Slice-based Federation Architecture
SM	Slice Manager
SQL	Structures Query Language
UFF	Federal Fluminense University
UFPA	Federal University of Pará
UFRJ	Federal University of Rio de Janeiro
URL	Uniform Resource Locator
USP	University of São Paulo
VPN	Virtual Private Network
VT	Virtual Technology
XML	eXtensible Markup Language
XMLRPC	XML Remote Procedure Call
WDM	Wavelength Division Multiplexing
Wi-Fi	Wireless Fidelity
WiMax	Worldwide Interoperability for Microwave Access
WP	Work Package
WP2	Building and operating the Brazilian facility
WP5	Development of technology pilots and showcases

2 Scope

This document is the result of the task T4.2 – Contribution to the Federation framework of WP4 of the FIBRE project. The objective of this task is to select the right solutions for federation and customize them to the EU/Brazil facility.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [Bradner 1997].

3 Reference Documents

[D4.2] FIBRE deliverable document D4.2 “Report on the Federation physical interconnection”

[D4.4] FIBRE deliverable document D4.4 “Report on the Federation software tools deployment”

[demoMySlice] http://demo.myslice.info

[gitMySlice] https://git.myslice.info

[MS9] FIBRE deliverable milestone document MS9 “Second version of the enhanced OFELIA Control Framework software”

[MyPLC] https://svn.planet-lab.org/wiki/MyPLC

[MySlice] http://www.myslice.info

[OAR] http://oar.imag.fr

[omni] http://trac.gpolab.bbn.com/gcf/wiki/Omni

[openSFA] http://www.opensfa.info

[pypelib] http://code.google.com/p/pypelib/

[sface] http://git.onelab.eu/?p=sface.git;a=summary

[sfatables] sfatables: A firewall-like policy engine for federated systems. Sapan Bhatia, Andy Bavier, Larry Peterson, Soner Sevinc. ICDCS 2011.

4 Introduction

Testbeds are heterogeneous by nature as they address different technologies and objectives. Therefore, they carry the same polymorphic characteristic as the Internet. The coexistence of a highly heterogeneous set of experimentation practices and information coming from a wide variety of sources and administrative domains is a rather challenging task due to the different semantics and terminology, different usage models (best effort vs. reservations), different authorization policies, political issues, and more.

The most important step toward a global federation, which will allow experiments to access transparently all available testbed resources in a unified view, is the standardization of information and interfaces for the communication between all parties involved. For example, standardized experiment descriptions will allow a single experiment to be run in different testbeds. Standardized resource descriptions will allow experiments to browse and combine resources coming from different testbeds. Common authentication and authorization policies will facilitate the crossing of administrative boundaries. Efforts along these lines are on-going the last 3-4 years with the first steps being carried out mainly in the context of a single technology. For example, OMF and OML are standards that are being developed to formalize the experiment description and monitoring data for testbeds based on the ORBIT technology, SFA was introduced as a generic distributed federation architecture focusing on authentication, authorization, and resource descriptions, but which was initially implemented around the PlanetLab testbeds and more specifically the MyPLC software. Then different implementations tailored to other technologies like ProtoGENI or OpenFlow (FOAM) have been developed, which now ask for a more generic umbrella that will make possible the vision of a global unified view of really heterogeneous testbeds resources.

SFA appears as the most appropriate candidate for this challenging task since it has been already successful in bridging heterogeneous resources (e.g., ProtoGENI and PlanetLab) but also resources belonging to different administrative domains residing in different continents (notably the federation between the PlanetLab Central, Europe, Japan, and Korea). Eventually the Slice-based Facility Architecture or SFA will need to identify a minimal interface, a narrow waist that will enable testbeds of different technologies and/or belonging to different administrative domains to federate without losing the control of their resources. This is on-going work that has to address also political issues.

4.1 Federation, SFA and its motivations

The goal of the Slice-based Facility Architecture or SFA is to provide a minimal interface, a narrow waist that enables testbeds of different technologies and/or belonging to different administrative domains to federate without losing the control of their resources. This will allow researchers to combine all available resources and run advanced networking experiments of significant scale and diversity.

But the coexistence of a highly heterogeneous set of experimentation practices and information coming from a wide variety of sources and administrative domains is a rather challenging task due to the different semantics and terminology, different usage models (best effort vs. reservations), different authorization policies, political issues, and more.

To achieve its ambitious objective, SFA defines a distributed and secure API that allows researchers affiliated with federated administrative domains to browse all the available resources and allocate those required to perform a specific experiment, respecting the agreed federation policies. Note that SFA is more of a standard specification than a specific implementation (there are today different implementations for PlanetLab, ProtoGENI, and OpenFlow testbeds), which is being constantly updated based on the requirements that arise during its utilization.

In order for a testbed to become part of the current global federation enabled through SFA, given that a trust relationship has been established with at least one current members of the federation, there are three important technical requirements that need to be fulfilled:

1. Local testbed resources must be described with Resource Specification language of SFA (or an extension of it if the technology of the testbed is not currently supported) and exposed to the other federation members.
2. A friendly user interface must be available for researchers to be able to browse the available resources, express their requirements, and reserve the desired set of resources.
3. Resource characteristics, performance statistics and monitoring information (if applicable) should be provided to enable users to optimize their resource allocation decisions according to the objectives of their experiment.

In the following, we describe three different software modules being implemented today that aim to make it easy for testbed owners to support all above functionality focusing on the case of OpenFlow.

4.2 Experiment Control

OMF provides a generic framework for describing, configuring and instrumenting an experiment across several, potentially heterogeneous, resources. Experiment Description (ED) scripts are written in a Ruby-based domain-specific language called OEDL (OMF Experiment Description Language). Configuration and instrumentation of the experiment takes place through the interaction of a centralized component called Experiment Controller (EC), to which the user submits the ED, and Resource Controllers (RCs), residing on the involved resources. The RCs enrol into a given experiment, receive commands from the EC based on a standard protocol, and report the outcome of their actions to the EC. Communication is asynchronous and takes place through a publish-subscribe architecture.

It is relatively simple to add support for configuration of new resource types in OMF, through the addition of wrapper code for the respective configuration commands. Similarly, it is straightforward to add support for new applications running on the resources, through the addition of a simple wrapper application on top of the existing one. Therefore, OMF is a perfect candidate for experiment control across heterogeneous resources.

Furthermore, the publish-subscribe communication scheme used allows for efficient orchestration of experiments involving resources distributed across different testbeds. Peering of publish-subscribe servers, which was recently successfully tested among different OMF-enabled testbeds (NICTA, NITOS, PLE) provides a transparent connection between the different servers. From a user's perspective, this means that resources can be referred to within an ED script with use of standard hierarchical human readable names (e.g. "omf.nitos.node001"), regardless of the EC's location.

4.3 Bringing a testbed to SFA

We propose a set of freely available components that address the various issues raised by the federation of testbeds, and which have been demonstrated to operate in major federated facilities such as OneLab.

More specifically, we focus on the Slice-based Federation Architecture (SFA), which proposes a secure and distributed thin-waist allowing federation of heterogeneous architectures, and considers the slice (a set of resources associated to users at a given time) as its basic unit. SFA is a major contribution to the federation effort undertaken within the FIRE context (and is close to the GENI AM API developed in GENI), and considers such fundamental objects as resources, users, authorities and slices.

The components we describe here address topics ranging from bringing a testbed to SFA, to proposing a GUI supporting users across the whole experimental lifetime, including necessary aspects such as policies, scheduling and user management. We also insist on the importance of monitoring information, and address how to expose them to the federation.

The following requirements represent a common set of issues that any facility should come across and that should be considered in order to resolve various issues raised by the federation of testbeds:

- Control frameworks that support common interfaces or adapters
- Unified profile for certificate authority management
- Common data access interfaces
- Common resource and experimentation description language
- Independence of individual testbeds within the federated facility

4.3.1 Mapping SFA concepts to the testbed

The first step before considering any implementation is to map the notion of a SFA slice to the considered testbed, as well as the other objects such as resources, users and authorities (including naming). The latter ones are generally straightforward and consist in giving a unique/human resource name to the object, prefixed by an identifier for the testbed (e.g.

mytestbed.nodeX, mytestbed.userY, mytestbed.sliceZ) or some other authority (e.g. mytestbed.myauthority.objectT).

4.3.2 Available SFA implementations

A set of implementations are already available in several languages and can serve as a basis for developing the SFA interface to the testbed; let us cite a few:

- *Generic SFA wrapper* (python): http://www.sfawrap.info
- *ProtoGENI* implementation (perl): [no url]
- *GENI AM* (only a reference implementation)
- *FOAM*: http://groups.geni.net/geni/wiki/OpenFlow/FOAM (OpenFlow specific implementation)

4.3.3 Generic SFA wrapper

We go into more details regarding the **Generic SFA wrapper** since it provides a modular base on which to add drivers that will implement testbed specific operations. This implementation originates from the PlanetLab-specific implementation, which has been recently refactored to be used with other testbeds with minimal additions. More specifically, it already implements the various services offered by SFA (Aggregate Manager, Slice Manager, and Registry) and transparently manages the issues of authentication/authorization, routing (RSpecs splitting/merging), etc.

This implementation is convenient to make the bridge with existing testbed management interfaces, and can also be extended (though with no standard) to support fresh testbeds with no such functionality. The way the Generic SFA wrapper is designed today considers the testbed data is authoritative, and the different objects are imported from the testbed on a periodic basis (every 15 minutes, say), which means changes through the testbed management interface are not exposed immediately at the SFA level. Inversely, calls to SFA are immediately mirrored to the testbed management interface. This might be improved in future releases to allow for full synchronization of data.

A new testbed might either benefit from an existing driver, if the testbed has a similar flavour than one currently federated, or might require the development of a driver.

Existing drivers: PlanetLab, SensLab, NITOS, FEDERICA (work in progress), Teagle (work in progress).

Developing a new driver consists in writing a set of methods allowing the testbed information to be exposed under SFA. The two most important ones are the ListResource() API call that exposes the RSpecs with available resources, or the ones associated to a slice, and the CreateSliver() call that performs the actual booking of slice resources.

4.3.4 RSpecs

Independently of the chosen implementation, bringing new resources to a federation requires additions to the resource model, which in SFA is abstracted into RSpecs. While the only requirement for RSpecs is it being a XML document, the current trend is to somehow standardize its higher levels to facilitate RSpec manipulation (creation, merging, etc.).

There currently exist two main flavours of RSpecs, so called “SFA v1” and “ProtoGENI v2/v3”, on which it is possible to build upon.

4.3.5 SFA and OMF

The link between the SFA and OMF is the implementation of the omf_sfa which can be found here https://github.com/mytestbed/omf_sfa. This work is still in progress, but it is considered to be in its final stages and an initial deployment for testing purposes has been already done in NITOS.

The omf_sfa has been designed in the context of OMFv6 in which everything is considered a resource. It contains an SFA AM implementation and specifically the GENI AM v2 API. Furthermore, it has a REST interface and a native OMFv6 for the purposes of communicating with the OMFv6 Resource Controllers. The NITOS scheduler has been integrated as a scheduling entity of the omf_sfa implementation. That means that the used RSpecs (GENI v3) have been extended in order to include reservation information.

All the resources and reservation information is being kept in the database of the omf_sfa. The current information model is easily extensible towards the inclusion of new resources without the need to edit core parts of the code, like the SFA RSpecs generation code.

The last needed steps prior to production deployment of the omf_sfa are the hooking process among the OMF SFA AM and the corresponding resources. The issues that have not been resolved so far concern the authentication and authorization between the various RCs (Resource Controllers). This is something that has been tackled by design but not yet implemented.

4.3.6 SFA and OCF

OFELIA facility is an experimental facility comprising of individual entities as a federation of heterogeneous experimental facilities with a homogeneous control framework: OCF. This concept enables combining Infrastructural network resources and services of more than one administrative domain which enhances significantly the utility of the infrastructures and also enables the researcher to carry out experiments at a large scale. This is the case of two of the European testbeds in FIBRE.

OCF originally adopted Enterprise GENI (E-GENI) control framework for its facility as a base over which a lot of new features and functionalities were added as per OFELIA’s requirements. The E-GENI control framework was based on SFA.

The definitive OFELIA architecture is already finished, reason why OCF is at present suffering several modifications and enhancements (also by requirements coming from different projects). One possible deployment of this architecture is depicted in Figure 1. Nevertheless, some of the identities shown (EGW, IGW) are more logical than really implementable. This will be explained in following sections.

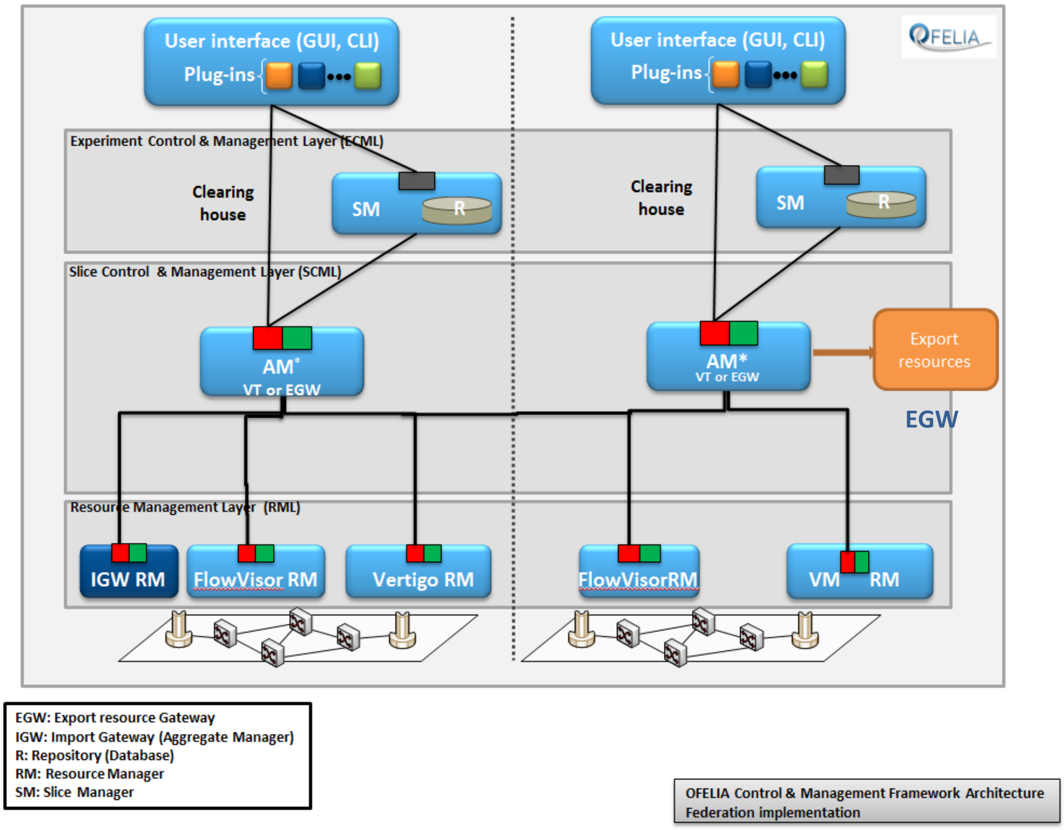
User interface (GUI, CLI)
Plug-ins
User interface (GUI, CLI)
Plug-ins
OFELIA
Experiment Control & Management Layer (ECML)
Clearing house
SM
R
Clearing house
SM
R
Slice Control & Management Layer (SCML)
AM*
VT or EGW
AM*
VT or EGW
Export resources
EGW
Resource Management Layer (RML)
IGW RM
FlowVisor RM
Vertigo RM
FlowVisorRM
VM RM
EGW: Export resource Gateway
IGW: Import Gateway (Aggregate Manager)
R: Repository (Database)
RM: Resource Manager
SM: Slice Manager
OFELIA Control & Management Framework Architecture
Federation implementation

Figure 1: OFELIA architecture for OCF v1.X

4.3.6.1 *Architectural building blocks*

The architecture is described in Figure 1, and is composed of the following entities: Slice Manager (SM), Repository (databases) (R), Clearing House (CH), Aggregate Managers (AMs), and Resource Managers (RMs).

Slice Manager

The slice manager provides a way for users to create and control their slices. The slice manager interacts with the repository (database) to retrieve user credentials and interacts with aggregate managers to create and control the slices.

Repositories (databases)

Repositories maintain the information about users (access and credentials) and projects.

Clearing House

The purpose of the Clearing House is to implement the logic specific to user projects and slices. The tasks of the Clearing House are to verify the user permissions, the operations performed on the slices, the slice policies, and the behaviour specification for each slice and between slices.

Aggregate Managers (AMs)

Aggregate Managers are entities with identical (or similar) interfaces, that can be put in a recursive architecture. This means, that these entities have an identical (or similar) interface. This applies to the north and south bound interface, so that they can be stacked. An Aggregate Manager serves resource booking and monitoring requests by either:

- Pre-processing/checking the request (e.g. against policies), breaking it up and delegating it further (down) or
- Managing the booking itself
  - If it does perform the booking itself it is called Resource Manager (RM)
- Same thing for monitoring requests.

This means an AM hides complexity of the lower AMs and collects information of the lower entities. Currently there are two kinds of AMs in production OCF: VT AM, for virtualized servers and Opt-in Manager for OF resources. The first one originally uses a custom XMLRPC API and RSpec which describes both the action to perform and the resources affected. The Opt-in uses also an XMLRPC API and GENIv1 OF RSpec. Nevertheless SFA APIs were added to these AMs, as also new AMs are about to be integrated in OCF in order to fulfil FIBRE Federation framework as it will be explained in following sections.

The architecture defines that AMs and RMs may form a hierarchical chain, also due to the fact they all speak the same APIs and have the same authorization and authentication frameworks. This will be achieved since all the different AM/RMs will be based on the same package which can be considered as the base toolset to build OFELIA AMs. It encapsulates common tasks which are performed by every AM/RM, such as Authentication & Authorization, interfaces such as the native OFELIA API, GENI API v2 and v3 and common AM/RM abstractions and mechanisms, like booking and monitoring logic.

The aggregation of AMs is something allowed by the architecture itself and may be implemented depending on the specific management requirements (resource aggregation). The architecture also defines that AM/RM component may implement policies locally according to the operational needs.

Figure 2 depicts the OCF architecture in a more general way, decoupled of a specific implementation.

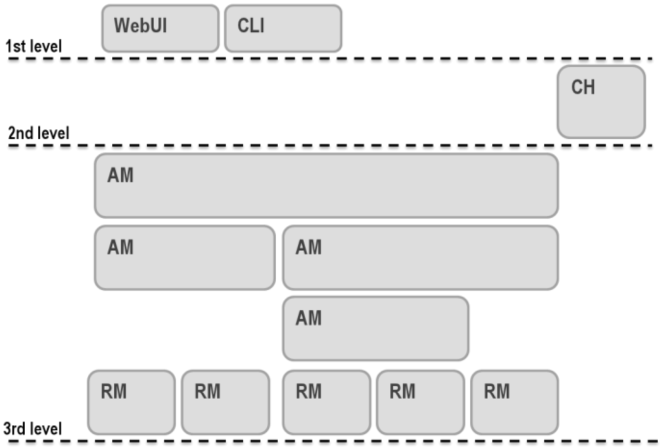

Figure 2: OFELIA architecture

In OFELIA, we divide federation into two: Inter-Federation and Intra-Federation. These two federation scenarios are discussed in the following sub-sections

4.3.6.2 *Intra-Federation*

Intra-Federation is defined as the federation of heterogeneous experimental facilities with a homogenous control framework. In intra-federation architecture, the UI will be a centralized entity which will talk to all the clearing houses in the lower layers. The Clearing house/Slice manager is responsible for communicating with all the aggregate managers (AMs) through its southbound interface to collect all information regarding the available resources and present it to the UI layer.

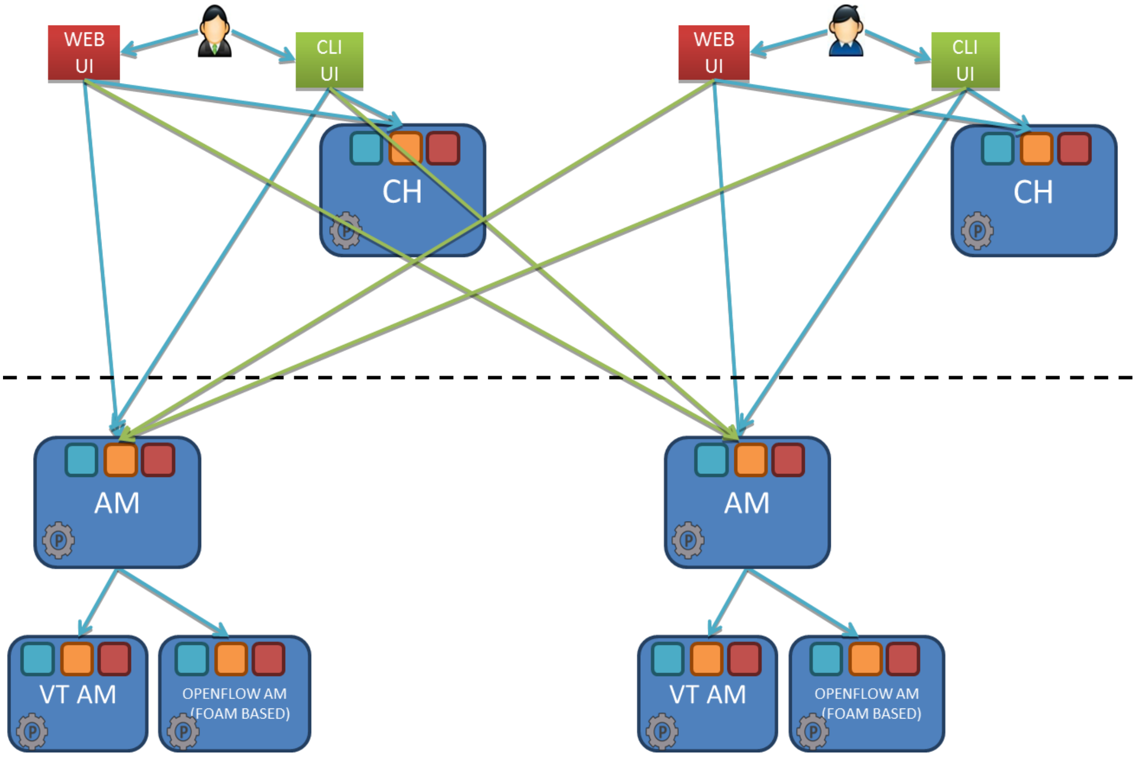

Figure 3: OFELIA intra-federation

For intra-federation (OCF in islands) it would be straightforward, i.e. discover AMs of other islands in respective OCF whereas for inter-federation (heterogeneous control framework) OFELIA will expose common interfaces to export resources to other testbeds.

4.3.6.3 *Inter-Federation*

Inter-Federation is defined as the federation of heterogeneous experimental facilities with heterogeneous control frameworks. When doing an Inter-federation, the resources will be made available to a different control framework through its association with the aggregated aggregate managers (AMs at the first level of the hierarchy). Aggregate AMs collect all information from the lower level AMs to present the available resources to the federation.

OCF's federation point is at the AMs, avoiding the fact of dealing with administrative issues of the other domain. The idea is that AMs are able to apply policies affecting the resources they manage, which are offered to an external testbed in certain conditions, but not caring about the specific users (even that would be possible if users' identifications are present in the requests). Island Managers, or administrators of those resources, will configure the certificates and policies for a specific federation.

The new architecture thought for OFELIA CF's AMs will count with a driver-based interface which will allow the AMs to support several APIs by developing their specific drivers. The first API identified is GENIv3, also supported by SFA. So that SFA appears as a feasible API to be used in FIBRE.

4.3.6.4 *OCF-SFA integration*

In order to make OCF resources available to the FIBRE Federation Framework as soon as possible, in addition to the internal OCF's roadmap in which FIBRE collaborates, an early integration of OCF and SFA was developed during the project. This integration has been done at the AM level. It is inspired on SFAWrap, although it was developed according to OCF particularities. The client part was kept as the accessible API for other testbeds and a driver has been added to every AM. This driver translates the API methods to the corresponding internal resource management. Figure 4 shows the SFA drivers inside the AMs that communicate with the SFA Client while the OCF own API driver is used to communicate with OCF GUI, maintaining the usual OCF functionalities.

Each AM has also a list of the trusted authorities. That means that when an AM receives a call from the SFA Client, this call includes the Client credentials, if those credentials are signed by one of the AMs trusted authorities, then the source is trusted and the method can be attended. As each AM has its own list of trusted authorities, which means that each AM can be federated independently, allowing certain clients for an AM and different ones for other AM.

More details about SFA implementation as well as the SFA API specification can be found on FIBRE deliverable milestone document MS9 [MS9].

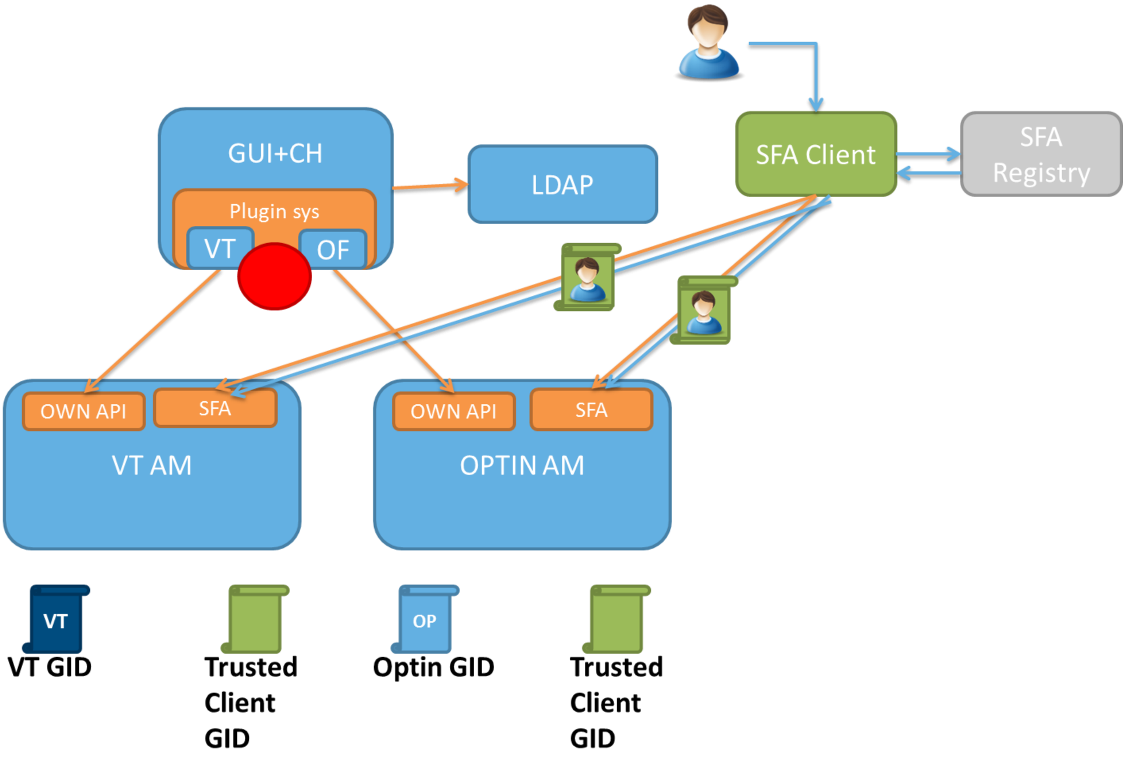

Figure 4: SFA drivers inside the AMs

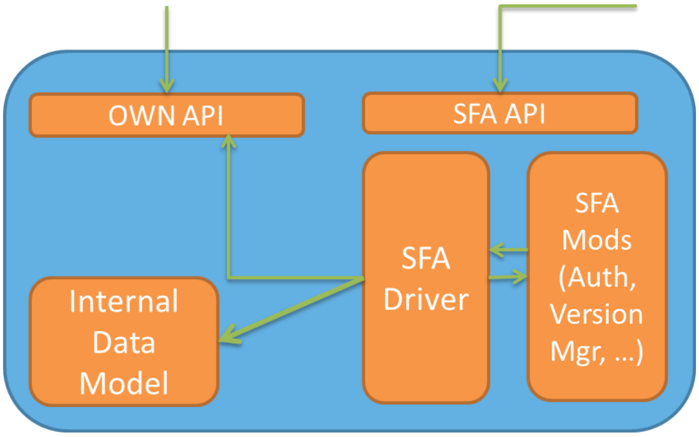

Figure 5: Internal structure of SFA federation

4.3.7 Discussion and supporting tools

4.3.7.1 *Towards a SFA Scheduler*

A large number of testbed today contain *reservable* nodes in some sense, whether they are homemade (e.g. PLE, NITOS) or more widely available. For instance, by bringing SensLab to SFA and thus providing an interface to OAR (a resource manager for large clusters), a ready-to-use scheduler is being proposed, which can be used by new platforms willing to adopt one. It will of course remain possible to bridge other tools similarly.

4.3.7.2 Handling user management

Currently, all testbeds are responsible for managing their users, from subscription to local authentication (possibly thanks to SFA).

Future work will focus towards allowing for an independent entity to manage users for a set of testbeds (we can think of the OneLab Experimental Facility offering such as service for example, that would correspond to an external registry for users, in SFA terms).

4.3.7.3 Federation policies

The Generic SFA wrapper comes with the *sfatables* utility [sfatables] that proposes a firewall-like administrative tool for federated systems. It allows to selectively expose resources to a third party while still controlling which are exposed and how they are used.

In addition, pyPElib [pypelib] is a small python library to help programmers to use abstractions provided by the library to build rule-based Policy Engine(s) within a certain scope of action. In a nutshell, pyPElib provides support for rule-based policy enforcement environments, with native support for flexible syntax definitions (mappings), rule-based action triggering, and logging. pyPElib could be used to develop a consistent engine to control the conditions in a certain federation schema (filtering the resources shown by one testbed to the other, create rules based on diverse criteria not limiting only to which is the testbed sending the request, etc.). It is being used in FIBRE within the developments in OCF.

4.3.7.4 User interfaces

It is of utmost importance for a solution to have a wide range of user interfaces allowing a convenient interaction with the testbed. We present here the major ones classified by category.

It is important that any singular testbed belonging to the federation can use its own user interfaces and still have access to the federated resources. This is not compatible to the possibility of having an extra UI shared by the federated testbeds.

Command line:

- sfi.py (in the Generic SFA wrapper codebase)
- omni: http://trac.gpolab.bbn.com/gcf/wiki/Omni

Libraries:

- sfaclientlib (in the Generic SFA wrapper codebase)
- java client library [no url]

Desktop GUI:

- sface: http://git.onelab.eu/?p=sface.git;a=summary

Web-based GUI:

- MySlice (recommended): http://www.myslice.info (*see Section 4.4)*
- OFELIA webUI (now integrated with the current clearinghouse implementation)

4.3.8 SFA Specifications

This implementation follows as much as possible the SFA specifications advertised at http://www.opensfa.info, which tries to reach a consensus among the involved partners, and can serve as a reference.

4.4 MySlice: a modular GUI for supporting users' experiments

One could see SFA as an agnostic container where resource descriptions (called RSpecs) are communicated between users and testbed providers. Then one would need sophisticated tools to interpret the different RSpecs coming from testbeds of different technologies and present them to the users in a meaningful way. There are various such tools under development today, such as omni (a command-line interface), sface (a python desktop application), and flack (a flash based web application).

MySlice is an ambitious project aiming to support researchers throughout the lifecycle of experiments that run on different testbeds spanning different administrative domains and networking technologies.

The coexistence of a highly heterogeneous set of experimentation practices and information coming from a wide variety of sources and administrative domains is a rather challenging task due to the different semantics and terminology, different usage models (best effort vs. reservations), different authorization policies, and more.

MySlice follows a bottom-up approach by starting simple and exposing the available through SFA resources to the users through a standard web interface and providing all the necessary information and the corresponding tools for annotating and filtering them "manually" (no automatic mapping between a high-level experiment description to substrate resources like in the case OMF).

The objective is through this process to learn from the users about the requirements of cross-technology future internet research and add complexity along the way. There is also an effort to engage testbed owners and developers into a collaborative open source development effort, which will enable developers with expertise on different testbed technologies and different experimental practices to work in parallel for optimizing the tools presented to the users allowing them for a wide range of choices according to their own requirements.

For this, the MySlice web framework provides today a modular implementation of independent plugins and a message passing interface shared between them. They are divided in three main categories: 1) query editing, the type of resources that the user is interested in; 2) data display, the visualization of resources that match the selected query; 3) and resource allocation, the selection and reservation (if needed) of resources. The information on which these tools operate come from a backend system which collects and represents in a standard format (i.e., a dictionary) the available SFA resources (MySlice API) and all available monitoring information about them (TopHat API).

These two APIs are completely independent of the web framework and are the providers of the raw information available globally using exactly the same XMLRPC APIs. Then a thin web framework is responsible for authentication and authorization issues, for presenting the above information to users, and enabling them to choose the most suitable resources for their experiment.

The interface between the plugins is the publish/subscribe system (based on jQuery) of the corresponding shared information as the user interacts with the system. For example, query editing plugins publish new versions of the active query (e.g., with more refined filters or more requested information fields). Data display publishes the user choices on the current set of results received based on the active query as reported by SFA and TopHat API (subscribers to query and publishers of results). We have implemented today different examples of such plugins, which can make it very easy for developers to extend them or provide alternative ones with minimal knowledge of the implementation details of the system.

The code is available at https://git.myslice.info and will hopefully enable different testbeds and portals to use and improve the codebase according to their needs contributing to a collective knowledge. Then, encouraging researchers to share the information produced by the usage of these platforms and provide explicit feedback for their personal needs can further contribute in the development of more sophisticated resource provision and experiment instrumentation tools.

Opening in this way the development of web-based user tools for experimentation and sharing effort and information can increase significantly the chances for the achievement of this challenging objective.

MySlice web framework

A web page in MySlice is conceived as the output of a query encoded in the URL. The query first defines the target object or table that is requested by the user (called the method). Today there are three main methods supported: slices, nodes, and agents. Then for each method the query defines the requested annotation information (the fields) and filtering parameters.

Upon the loading of a page, the corresponding query is dispatched asynchronously to the three main modules of the framework:

- The SFA module requests, through the standard SFA API, the RSpecs corresponding to the user and/or slice in question.
- The TopHat module fetches the annotation information corresponding to the selected fields and filters (e.g., node characteristics, geographic location, etc.).
- The rendering module initializes the selected set of plugins, which asynchronously wait for the results of the query and communicate between them the various user choices.

There are three main categories of plugins:

1. Query editing: one can imagine a wide variety of ways to help a user express his requirements in terms of resources and annotation information. From simple SQL-like free text queries, to modular editors, even graphic interfaces.
2. Display of results: given a selected query there are again various ways to visualize the results. From a simple table to sophisticated geographic maps or graphical representations.
3. Selection and reservation of resources.

At our development page (http://demo.myslice.info), you can see some draft implementation of such plugins.

The main information flow between plugins belonging to different categories is as follows:

Query → Display: Query editing plugins publish query updates (a new field requested or a filter applied), which are received by the plugins responsible for the data display.

Display → Allocation: Display plugins offer different options for users to select the resources that wish to attach to their slice (the simplest being the automatic selection of all results matching the currently active query).

Going back and forth through these different functionalities users can narrow down their search and select the resources that match most their requirements.

4.5 Interconnecting monitoring systems with TopHat and Manifold

The integration of monitoring systems in the lifecycle of an experiment is key to provide the experimenters information about the results of their experiments. This integration relieves the users from having to perform the deployment of such instrumentation. Deliverable document [D4.4], which focus on the monitoring and diagnosis tools to be used in the facility, offers a complete description of several available monitoring tools considered to be used in FIBRE and the ones finally chosen and the rationale.

It is often the case that a testbed has one or several monitoring and measurement systems, with heterogeneous data and interfaces. **TopHat** (http://www.top-hat.info) uses as MySlice proposes an interconnection framework (named Manifold) allowing the different platforms to expose their information in a consistent way to the federation, and it helps mapping the different measurements to the corresponding SFA resources identifiers.

The Manifold information framework is able to answer user queries by a combination of two or more platforms, and thus provide information that was not previously available through individual platforms. MySlice uses it heavily to expose monitoring information to its users, and to allow for RSpec annotation.

More specifically, Manifold proposes a simple an efficient **query language** to request several sources of data and simple processing across interconnected platform, with many similarities to the capabilities offered by SQL;

- an engine performing efficient **query dispatching and result aggregation**;
- a **standard interface** for platforms to expose measurement information (similar to tables in a relational database), alongside a metadata syntax to describe them;
- a set of **gateways** to support platforms not complying to this interface. Those gateways are responsible for handling authorization/authentication with the platforms, and interpreting the transport of data (e.g. XMLRPC), their format (e.g. XML), and the ontology used for representing information;
- a modular interface for the visualization (used by MySlice for example).

Our work towards the interconnection of measurement systems is tightly linked to the efforts in federating testbeds, and should allow for an ecosystem of measurement systems to emerge, and the various actors to seamlessly exchange information. This corresponds to the vision of a distributed federation of systems (as opposed to a single central entity), where the data can be accessed through different entry points (with their own strength and specificity) and still allowing access to the full range of available information.

5 The FIBRE context and the federation of its components

FIBRE is mostly composed of OpenFlow islands compliant to OCF, one Emulab testbed, and Wireless NITOS testbeds compliant to OMF.

The various steps to follow in order to achieve global federation of the FIBRE resources are as follows:

1. *Deploy similar basic testbeds in Brazil and in Europe.*

This process already started as OpenFlow islands already exist in Europe and will be extended further. In the Brazilian side, several OpenFlow islands following the OFELIA model will be deployed, making use of NetFPGAs to implement OpenFlow 1.0 enabled switches.

For the wireless testbeds, NITOS compliant testbeds will be deployed in some Brazilian nodes. Those will follow the NITOS specifications in order to avoid the burden of managing incompatible platforms. NITOS testbeds will also be extended in European locations such as UPMC.

2. *Interconnect the testbeds components on both sides.*

The European part of the FIBRE infrastructure will be aggregated at the NOC located and managed by University of Bristol. Alike for the Brazilian side where the NOC is in charge of RNP. University of Bristol NOC and RNP will connect physically the two aggregated components. This process is currently under construction.

The European and Brazilian sides will count with two connections, from Europe both ending at RNP backbone. On one hand, University of Bristol will connect physically to RNP via Internet2. This process is currently under construction. On the other hand, the same thing is in process between i2CAT and RNP, where a connection via RedClara and RedIRIS is being deployed.

3. *Federating the FIBRE infrastructure.*

As depicted earlier, federation is much more than interconnection. The FIBRE components are heterogeneous and reassemble two different communities. The one that builds on the OneLab facility (mostly constructed around the PLE “PlanetLab Europe” and NITOS) and that have been promoted SFA and OMF, and the one issued from OpenFlow and carried out thanks to many initiatives, among those OFELIA plays a prominent role in Europe. FIBRE offers the potential to bridge these communities and explore what should be a reasonable architecture and its related components to federate them.

The main components of the OneLab (PLE+NITOS) and OFELIA facilities are given in the following table:

Component	**PLE**	**NITOS**	**OFELIA (OpenFlow)**
Resource discovery	MyPLC or MySlice	NITOS specific	OCF
Resource requirements	MyPLC or MySlice	OEDL script	OpenFlow enabled resources, virtualization resources.
Resource reservation	MyPLC or MySlice	NITOS Scheduler	Hard reservations
Resource provisioning	SFA-like XMLRPC	Reservation, NITOS Scheduler	OCF (XMLRPC)
Experiment Control	Third-party such as OMF	OEDL script OMF experiment controller	OFELIA slicing mechanism Web interface to control OFELIA resources
Monitoring	TopHat & third-party tools	?	Zenoss for infrastructure
GUI	MySlice or XMLRPC	NITOS specific	Based on Expedient, WebUI

Indeed, these systems are not using similar components and besides, there have different organization, management and governance solutions that will need to be further explored.

6 FIBRE Federation architecture

Concerning the FIBRE Federation architecture, some decisions were taken related to the number of authorities, naming, user portals, control plane federation, etc. In the following, all these decisions are enumerated and explained.

It was agreed that the FIBRE testbed will have two top-domain authorities, one under the responsibility of Brazil and another under Europe responsibility. Then, these two authorities will be peered in order to achieve a federation. This peering implies that a SFA Register has to be deployed in each side and each authority has to sign the certificate issued by the other authority. It was also agreed that these two top-authorities will not have sub-authorities.

Figure 6 shows the peering of EU-BR authorities for FIBRE Federation. Concerning the name of these authorities, the Brazilians will soon decide on the name and submit it to the Europeans for approval. European side has decided to use the OpenLab authority.

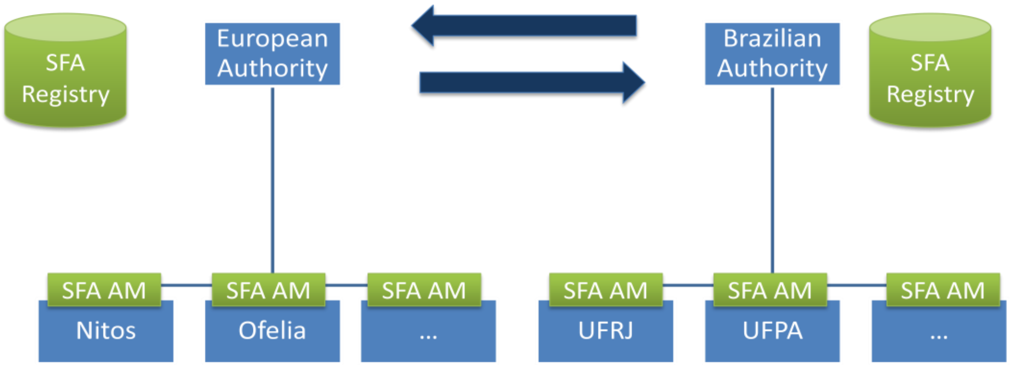

Figure 6: FIBRE top-authorities

Concerning the user portal, it was agreed that FIBRE will have at least one portal per top-authority, and the software component of choice was MySlice ().

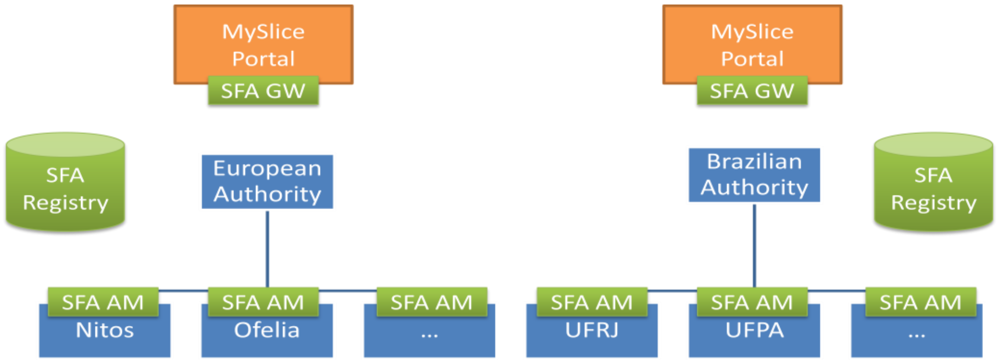

Figure 7: Fibre Federation Portal

Concerning the federation control plane, Figure 8 shows the interaction of MySlice portal with NITOS and OFELIA CMFs, i.e. OMF and OCF, respectively, through its SFA GW component, which includes the responsibility of Slice Manager. Since the Aggregate Managers (AMs) of OCF, i.e. Opt-in and VT-Manager, already implement the GENI v3 API, the SFA GW interacts directly with these two AMs. As OMF 5.4 still does not support SFA API, an SFA Wrapper (or SFA Driver) will be needed, which is already implemented by the Generic SFA Wrapper. In its new version, OMF 6.0 will natively support SFA through the omf_sfa component.

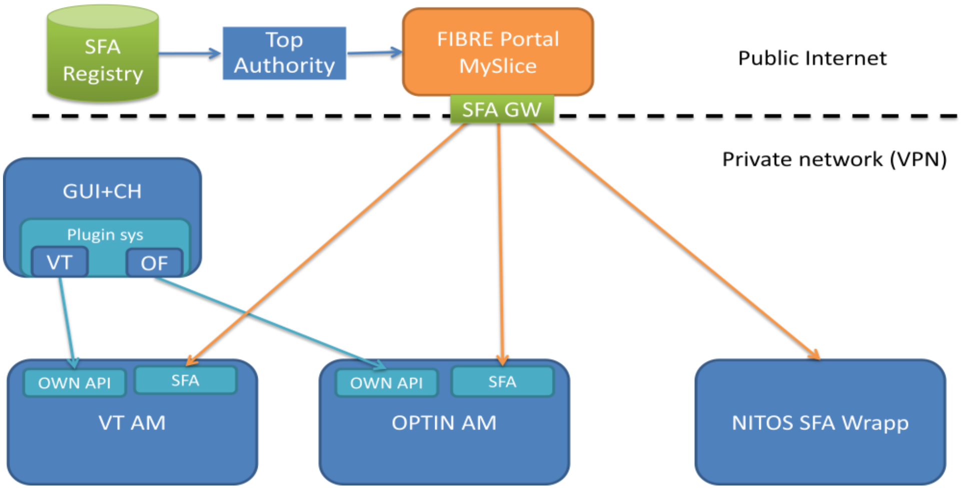

Figure 8: FIBRE Federation Control Plane

Concerning the OMF user management, a possible synchronization between LDAP and SFA Registry is under discussion. This will allow that LDAP user management, already deployed in OFELIA and Brazilian testbeds, be maintained. Figure 9 illustrates this synchronization.

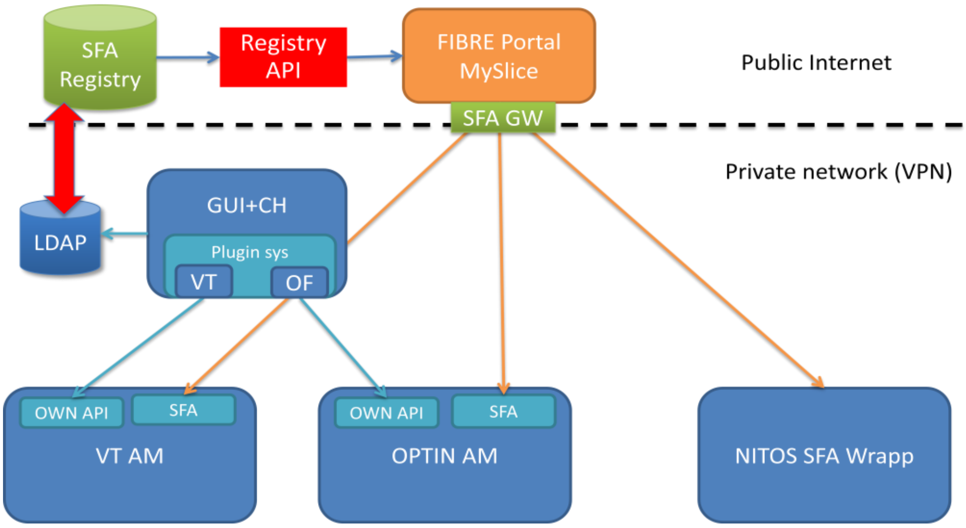

Figure 9: LDAP and SFA Registry Synchronization

Concerning the implementation of access policies, the component of software sfatables is a good candidate.

7 Conclusion

We consider the testbeds framework as a playground to explore the practicality of federation. We introduce some potential components like SFA, OMF, MySlice, and TopHat that could provide the foundation for such architecture. Work is in progress to achieve a maturity of the building blocks, to play with heterogeneous testbeds at scale. We will learn by addressing the many challenges that we are facing to address a world of diversity. An example is given by the complex nature of scheduling the local resources provided by the different testbed authorities, for instance in order to enforce resource reservation.

The FIBRE project offers the opportunity to bridge two different testbed communities to explore adequate federation architecture and its relevant components.

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

END OF DOCUMENT