Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

TITLE: Implementation of the Ofelia Control Framework (OCF) for Open Flow-based testbed facilities

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Oscar Moya Gomez

DIRECTOR: Salvatore Spadaro

DATE:

Overview

This master thesis focuses on the concept of Infrastructure as a Service (IaaS). It firstly highlights the necessity to build experimental facilities that allow researchers to run their own experiments on novel architectures and protocols; this way the validation of their proposals is performed in real traffic scenarios. Then, the master thesis discusses the capabilities offered by the OpenFlow protocol to efficient build such experimental facilities. The protocol itself is presented and analyzed .In such a context, the main contribution of the master thesis is the implementation of the OpenFlow-based Ofelia Control Framework in order to overcome its limitations in offering services such as testbeds federation. The software modules that have been implemented in the framework of this master thesis are part of the Ofelia testbed available at the Fundació I2CAT premises. Experimental validation of the implemented modules is also presented.

Index

List of Figures

List of Tables

List of Acronyms

AM – Aggregate Manager
BFS – Breath First Search
CH – ClearingHouse
CM – Component Manager
CPU – Central Processing Unit
CRUD – Create, Read, Update and Delete
CSS – Cascade Style Sheet
DB – DataBase
DFS – Depth First Search
DNS – Domain Name Server
FI – Future Internet
FIBRE – Future Internet testbeds experimentation between BRazil and Europe
FIRE – Future Internet Research and Experimentation
FIT – Future Internet Testbed
FS – FlowSpace
FV – FlowVisor
GENI – Global Environment for Network Innovation
GID – Global IDentifier
GUI – Graphical User Interface
HTML – Hyper Text Mark Language
HTTP – Hyper Text Transfer Protocol
HTTPS – Hyper Text Transfer Protocol over SSL
HRN – Human Readable Name
IaaS – Infrastructure as a Service
ID – Identity
IM – Island Manager
InP – Infrastructure Provider
IP – Internet Protocol
ISP –Internet Service Provider
IT – Information Technology
LAN – Local Area Network
MAC – Media Access Control
NGN – Next Generation Network
OCF – Ofelia Control Framework
OF – OpenFlow
OFAM – OpenFlow Aggregate Manager
OFELIA – OpenFlow in Europe Linking Infrastructures and Applications
OS – Operative System
PC – Personal Computer
PI – Principal Investigator
pyPElib – Python Policy Engine Library
QoS – Quality of Service
R – Registry
RAM – Random Access Memory
RSA – Rivest, Shamir and Adleman
RSpec – Resource Specification
SDN – Software Defined Network
SFA – Slice Based Federation Architecture

SM – Slice Manager
SP – Service Provider
SSH – Secure Shell
SSL – Secure Socket Layer
SU – Stanford University
TCP – Transmission Control Protocol
TLS – Transport Layer Security
UI – User Interface
UUID – Universal Unique Identity
VLAN - Virtual Local Area Network
VM – Virtual Machine
VN – Virtual Network
VPN – Virtual Private Network
VTAM – Virtualization Aggregate Manager
XML – eXtensible Markup Language
XMLRPC – eXtensible Markup Language Remote Process Call

1. Introduction

Internet has become a critical infrastructure due to its ossification caused mainly by the absence of changes in the core networks. The changes have not been produced basically for two reasons:

- The investments done by the Internet Service Providers (ISPs) for legacy networks and the required investments to make any change in the core network are too high.
- To change the core networks efficiently an agreement of most of ISPs is required; this is very complex task.

The high exponential growth of traffic demand over the last twenty years and the increase of both end-users and services, have pushed to perform these core changes since many years ago; however, only some “patches” have been deployed that made the internet just work [1].
In such an environment, the deployment of solutions that require architectural changes has gained more and more attention. Many solutions have been investigated but there is still a big problem: the testing. There are two main problems to test new protocols/algorithms over real networks:

- The required investment to test with real traffic a new protocol/architecture in a self-constructed physical topology is very high.
- Depending of the deployed protocols extra flexibility from the network equipment might be required. This means to have the vendor equipment open to implement programmable software platforms. Nevertheless, it is very hard to have the vendors disclosing (opening) their technological solutions and algorithms, due to the high investments they made to get such solutions.

A solution for the first problem is to deploy tests using programmable virtual networks. Network virtualization provides a logical network (virtualized and isolated) on a shared physical infrastructure, giving to the researchers the environment (flexible, programmable, reliable, etc.) to deploy their tests.
And a solution for the second problem is to use network solutions with the following flexibility degrees [2]:

- Amenability to high-performance and low-cost implementations.
- Capability of supporting a broad range of research experiments.
- Assurance to isolate experimental traffic from the production traffic.

In order to provide a real environment to carry out experimental activity for the researchers, multiple Future Internet Testbeds (FIT) have appeared, with the main aim to provide the virtualized network and the programmable network equipment. These testbeds are facilities deployed in networks where there is a lot of traffic, such university campuses.
In this project, it will be firstly described the importance of FIT projects for the Future Internet; then the concepts of Software-Defined Networks (SDNs) and federation capability are discussed. The facilities that use the OpenFlow protocol, such as the Ofelia Control Framework (OCF) are finally considered.

2. Objectives of the Thesis

The main goals of this master thesis are:

- Introduction of the concept of Software Defined Networks (SDNs) and enabling OpenFlow protocol.
- Description and concept of the Future Internet Projects (FIT) from the Future Internet Research and Experimentation (FIRE) initiative; special emphasis is devoted to those projects based on Ofelia Control Framework.
- To relate the importance of FIT projects, such as OCF implementing SDNs and OpenFlow, in order to allow to the researchers experiment with their own protocols.
- To emphasize the added value that the federation provides for FIT projects. Basically it is opening for researcher new ways to obtain different resources from other testbeds.
- To describe the work done for OCF in order to improve its initial features.
- To make a set of tests to demonstrate the functionality of the designed OCF as well as to highlight the utility of the modules developed.

The work done in this master thesis has been implemented into the Fundació i2CAT OCF Island, in the framework of the European project Future Internet testbeds experimentation between BRazil and Europe (FIBRE) [3].

3. Basic Concepts

In the introduction of this master thesis, it is explained the main inefficiency of the legacy networks and some possible solutions. Basically, new standards and protocols are required to overcome the “ossification” problem. To help the research community to deal with these problems, new network solutions are required. However, to offer the required tools to the researchers, high quantity of resources and investment is required, not only in resources but also in maintenance and deployment of facilities to offer these resources.
To minimize the investments, the concept of Infrastructures as a Service (IaaS) architectures is used. This approach allows to different facilities be interconnected using cloud computing; by this way, every facility can support only few types of resources. Therefore, with a relatively small amount of investment supplied by different facility owners, a heterogeneous multi-resource facility can be provided to the researchers for their experimental activities on novel algorithms, concepts, architectures, etc.

3.1 Infrastructures as a Service

Infrastructure as a Service (IaaS) concept is based on the abstraction of hardware components (servers, storage and network infrastructure) into a pool of computing, storage, and connectivity capabilities that are delivered as services. These services usually are provided using virtualization techniques.
The end users (for example researchers) take the responsibility for the configuration and operations of the guest Operating System (OS), software, and Database (DB). The consumer takes on the operational risk that exists above the infrastructure.
The essential characteristics of IaaS are [4]:

- **On-demand self-service.** A consumer can independently and unilaterally provision computing capabilities, such as network connectivity and storage, as needed automatically without requiring human interaction with each service’s provider.
- **Broad network access.** Capabilities are available over the network and accessed through standard mechanisms that promote the use by heterogeneous thin or thick client platforms.
- **Resource pooling.** The provider’s computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. The customer generally has no control or knowledge over the exact location of the provided resources, but may be able to specify location at a higher level of abstraction (for example, country, state, region, or datacenter). Examples of computing resources include storage, processing (compute), memory, network bandwidth, and virtual machines.
- **Rapid elasticity**. Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out, and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

- **Measured Service.** Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (for example, storage, compute, bandwidth, active user accounts, etc.). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

The Future Internet Research and Experimentation Initiative (FIRE) is an initiative to provide new ways to approach the Internet from the most fundamental level early experimenting and testing in large-scale facilities (IaaS), creating a multidisciplinary research environment for investigating and experimentally validate highly innovative ideas for new networking and service paradigms.

3.1.1 Future Internet Research and Experimentation Initiative

FIRE initiative is promoting the concept of experimentally-driven research, combining the academic research with the wide-scale testing and experimentation that is required for industry.
FIRE also works to create a dynamic, sustainable, large scale European Experimental Facility, which is constructed by gradually connecting and federating existing and upcoming testbeds for Future Internet technologies [2].
The FIRE based projects propose, as a solution for the “ossification” problems, the implantation of Software Defined Networks (SDN) along the FIRE facilities and Virtualization tools to offer to the research community the most flexible way to research and experiment with new protocols and architecture.

3.1.1.1 Software-Defined Networks

Software Defined Networks (SDN) is a network architecture where network control is decoupled from forwarding and is directly programmable. This migration of control, bounds the individual network devices into accessible computing devices, enabling the underlying infrastructure to be abstracted for applications and network services. The network is thus treated as a logical or virtual entity.
Network intelligence is logically centralized in software-based SDN controller, which maintains a global view of the network. The network appears to the applications as a single logical switch. With SDN, enterprises and carriers gain vendor-independent control over the entire network from a single logical point, which simplifies the network design and operation. The SDN devices also are simplified since they no longer need to understand and process several protocol standards but only accept instructions from the SDN controllers.
Network operators and administrators can programmatically configure a simplified network abstraction easily. In addition, leveraging the SDN controller’s centralized intelligence, it can alter network behavior in real-time and deploy new applications and network services faster. By centralizing network state in the control layer, SDN gives network managers the flexibility to configure, manage, secure, and optimize network resources via dynamic, automated SDN programs. Moreover, these programs can be customized by the programmers. SDN architectures also support a set of APIs that make possible to implement common network services, including routing, multicast, security, access control, bandwidth management, traffic

engineering, quality of service, processor and storage optimization, energy usage, and all forms of policy management, custom tailored to meet business objectives.
With open APIs between the SDN control and applications layers, business applications can operate on an abstraction of the network, leveraging network services and capabilities without being tied to the details of their implementation. As a result, computing, storage, and network resources can be optimized [4].

3.1.1.2 Virtualization

3.1.1.2.1 Hardware Virtualization

Hardware virtualization refers to the creation of a virtual machine that acts like a real computer with an operating system. Software executed on these virtual machines (VM) is separated from the underlying hardware resources.
In hardware virtualization, the host machine is the current machine on which the virtualization takes place, and the guest machine is the virtual machine. The software or firmware that creates a virtual machine on the host hardware is called a hypervisor [5] [6].

3.1.1.2.2 ***Network Virtualization***

The solution to eliminate the network ossification problems is to enable the coexistence of different virtual networks on the top of same physical infrastructure. This work requires split up with Internet Service Providers (ISPs) in infrastructure providers and service providers as next figure shows:

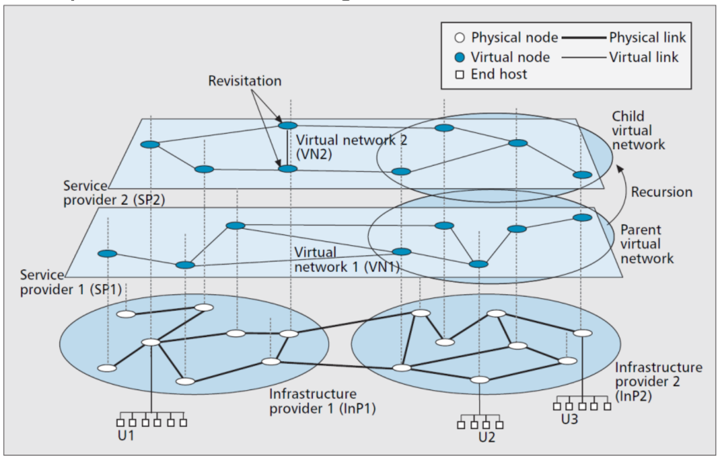


Figure 1. Network Virtualization Environment [7]

Infrastructure Providers (InPs): Managers of the physical infrastructure.
Service Providers (SPs): Virtual Networks (VNs) managers joining resources from different InPs and offering end-to-end services (to the users or even to other SPs).
The main features of network virtualization should be [8]:

- **Flexibility:** Every SP should have freedom to deploy any topology, routing protocol or other controlling mechanism from the InP resources.
- **Manageability**: The separation into InPs and SPs should provide complete end-to-end control of the VNs to the SPs.
- **Stability and Convergence:** Virtualization must ensure the stability of a NVE. In case of any problem the affected VN/s should be able to converge to their working states.

- **Programmability:** The SP should be able to implement customized controlling protocols on leased infrastructure.
- **Scalability:** The number of VN should not be the limiting factor affecting the performance of the system.
- **Isolation:** The operations of a VN should not affect other VNs.
- **Heterogeneity:** the technologies comprising physical infrastructure should not affect network virtualization process. There should be heterogeneity with respect to underlying technologies and heterogeneity with respect to VNs.
- **Legacy Support:** The current network should be supported in the VN environment.

4. OpenFlow

OpenFlow (OF) protocol is a standard designed to be applied to enable flexible solutions based on the SDN concept. On its current specification, it can be applied only in some types of network equipment (basically Ethernet switches). However, it is being properly extended in order to be applied to more network equipment (such as optical switches).

Common Ethernet switches are divided into data plane and a control plane. The data plane consists of a forwarding table, which is a set of entries that refers to a table in where the router looks up the destination address of the incoming packet and determines the path from the receiving element. The control plane is a set of actions executed on received Ethernet frames to decide their destination ports. This approach provides a fast execution but does not offer any flexibility in controlling frames.

In order to provide the flexibility required by the researcher to deal with the internet problems, OpenFlow switches separate the control and data planes and abstract the data plane using OpenFlow tables.

OpenFlow moves the control plane outside the switch in order to enable an external control of the data plane through a secure channel. Since Ethernet vendors' realizations of the data plane differ between each other's, OpenFlow implements a more general data plane abstraction, the flow tables representing the forwarding table of several switches.

In conclusion OpenFlow switches contain a flow table, a secure channel and are driven by a controller using the OpenFlow protocol [9].

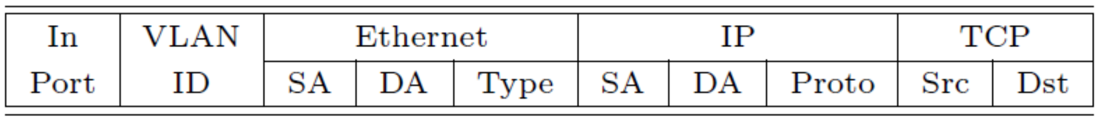
4.1 Flow Table

The Flow table is a set of entries intending to represent the forwarding tables of common Ethernet switches but in a more general form in order to provide as much flexibility as possible.

Each flow table entry contains header fields to match against packets, counters for update the matching packet and actions to apply to matching packets.

4.1.1 Header Fields

Every header of a flow that entries in the flow table are compared against the header fields of the switch. If a header is matched, there is the possibility of match another sub-header in order to improve the precision of a match.



In	VLAN	Ethernet			IP			TCP	
Port	ID	SA	DA	Type	SA	DA	Proto	Src	Dst

Table 1. Header Fields matched in OF switches [8]

4.1.2 Counter Fields

There are counters in the tables, the flows, the ports and the queues in order to have identified every match and monitor the network [10].

4.1.3 Actions

Each flow entry is associated with zero or more actions that determine how the switch handles matching packets. If no forward actions are present, the packet is dropped. Action lists for inserted flow entries are processed in the order specified.
There are two types of OpenFlow (OF) switches, the OpenFlow-enabled switches, that is, those switches that allow OpenFlow and Ethernet packets and the OpenFlow-dedicated switches, those switches that only allow OpenFlow packets.
The main difference of these switches is in the actions that they have to take. The OpenFlow-dedicated switches only have the required actions meanwhile the OpenFlow-enabled switches implement the required and other optional actions.
<u>List of Required Actions</u>
-Forward: packet forwarding to physical ports and the virtual ones

- **ALL**: Send the packet out all interfaces except the incoming interface.
- **CONTROLLER:** Send the packet to the controller.
- **LOCAL:** Send the packet to the local networking devices.
- **TABLE:** Perform actions in flow table. Only for packet-out messages.
- **IN PORT:** Send the packet out from the input port.

-Drop: Discard packets with no specified action in the flow-entry.

<u>List of Optional Actions</u>
-Forward: Add other forwarding directions to the existing ones.

- **NORMAL:** Process the packets using the traditional forwarding path supported by the switch (i.e., traditional L2, VLAN, and L3 processing.). If the switch does not support the normal forwarding, should be specified.
- **FLOOD:** Flood the packet along the minimum spanning tree except to the incoming interface.

-Queue: The queue action forwards a packet through a queue attached to a port. Forwarding behavior is dictated by the configuration of the queue and is used to provide basic Quality-of-Service (QoS) [10].

4.1.4 Packet Matching

Every entering packet in the switch is parsed and the values of the headers are compared to the entries in the header fields.

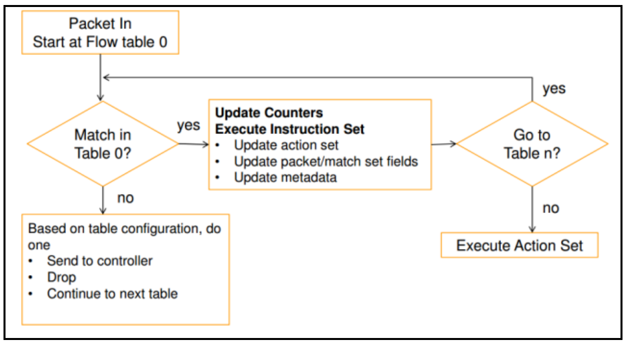


Figure 2. OpenFlow Packet Matching workflow [10]

If there is a match in any entry, then the action of the entry is executed. If there is not any match, the packet is forwarded to the controller or to other tables (depending on the switch configuration).
The next figure shows the workflow of the packet matching.

4.2 Secure Channel

The secure channel is the interface that connects the OpenFlow switch/s to the controller. Through this interface, the controller configures, manages and receives packets from the switches or sends out the managed packets to other devices.
The secure channel establishes and terminates the connection between OpenFlow Switch and the controller using Connection Setup and Connection Interruption procedures.
The secure Channel connection is a Transport Layer Security (TLS) connection. Switch and controller mutually authenticate by exchanging certificates signed by a site-specific private key [8].
The communication between the switches and the controller is defined by the OpenFlow protocol that allows three types of communication establishment:

- Controller-to-switch: Synchronous communication generated by the controller in order to get the state of the switch. These messages can require a response from the switch or not.
- Asynchronous: Asynchronous messages by the switches without the controller request. Switches send these messages to inform about a packet arrival, change of state or some kind of error.
- Symmetric: Symmetric messages sent without solicitation by the controller or the switch in order to get additional information or test the connection.

4.2.1.1 Controller-to-Switch Messages

Here, the messages between the controller and the switch are described:
Features: The controller sends a features request message to the switch. The switch must reply with the capabilities supported by the switch.
Configuration: The controller can set and query configuration parameters in the switch. The switch only responds to a query from the controller.
Modify-State: Add/delete or modify flows in the flow tables and to set switch port properties.
Read-State: Collect statistics from the switches flow-tables, ports and flow entries.
Send-Packet: Send packets out of a specified port on the switch.
Barrier: Ensure message dependencies have been met or to receive notifications for completed operations.

4.2.1.2 Asynchronous Messages

The possible asynchronous messages are:
Packet-in: For all packets that do not have a matching flow entry.
Flow-Removed: When a flow entry is added to the switch by a flow modify message. The flow modify message also specifies whether the switch should send a flow removed message to the controller when the flow expires.
Port-status: When the port configuration state changes.

Error: Notifies to the controller that an error has occurred.

4.2.1.3 Symmetric Messages

The symmetric messages can be:

Hello: Exchanged messages between the switch and the controller when the connection starts.

Echo: Messages that must return an echo reply used to calculate the latency, bandwidth, etc.

Vendor: To offer additional functionality within the OpenFlow message type space.

4.3 Controller

4.3.1 FlowVisor

FlowVisor is a controller that provides a network virtualization tool to allow coexistence of multiple and isolated logical networks on top of same physical infrastructure. This tool allows several researchers to run experiments simultaneously and independently of each other. FlowVisor can be seen as network proxy.

FlowVisor provides the following features:

- Flexibility: The resource allocation and sharing should be flexible to support the virtual network creation used to research. Allowing to setting up different parameters as the topology used, the bandwidth, type of packets, etc.
- Transparency: The controllers and the physical layer should not be aware about the virtualization. The controller should act as if the whole network was controlled by its and the network should act as if only one controller was connected.
- Isolation: Existence of several virtual networks controlled by different controllers without a performance reduction and without interfere other neighbor networks.
- Extensible Slice Definition: Since the resource allocation in slices is being changed and researched usually the slice definition should be flexible in order to support as much as users as possible.

4.3.1.1 FlowVisor Architecture

FlowVisor has three main modules, namely: the translation, the forwarding and the Resource Allocation Policy.

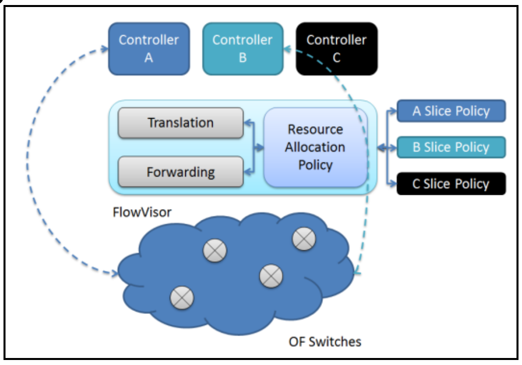


Figure 3. FlowVisor Architecture

In order to maintain different controllers and VN in the same physical infrastructure, FlowVisor applies a slicing policy. The slicing policy gives a set of MAC Ranges, IP ranges, Ports, etc. configured using the OpenFlow Protocol to every controller, in that way, each slice only contains one controller and the slice is isolated from the other slices.
Figure 3 shows how the FlowVisor Works. For every controller there is a slice policy in the FlowVisor. All the interactions between the controller and the physical topology are passed through the translation and forwarding modules.
The FlowVisor is able to forward the packets to the correct controller checking the slice policy. In this way, the FlowVisor can be seen as a proxy server.
Every packet in the physical network is supervised by the FlowVisor, in order to manage the different slices, the FlowVisor adapts the OF protocol to the slice policing parameters from the controller and gets sets the forwarding tables associated to each slice.

4.3.1.1.1 Messages from devices to OpenFlow controller

FlowVisor gets all the OpenFlow messages sent from devices and acts in consequence:

- Send the control plane messages to the Slice controller if the source device is in the slice topology.
- Rewrites OpenFlow negotiation messages in the way that the slice controller only sees the ports in its slice.
- Port up/down messages are only forwarded to affected slices.

4.3.1.1.2 Messages from OpenFlow controller to devices

FlowVisor also get the OpenFlow controller messages and applies the following actions.

- Rewrites the flow Insertion, deletion and modifications in order to do not break the slice definition.
- Expands the flow rules into multiple rules to fit the policy.
- Returns “action is invalid” error if the controller tries to control ports/packets flows outside of the slice.

4.3.1.2 FlowSpace

FlowSpace is the collection of packet headers that can be assigned to a slice and provides the flexibility for the researchers to request a “FlowVisor slice” that provides different environments for make tests with their own OpenFlow controller. The headers that can be request/assigned by the FlowVisor are:

- Source and Destination MAC Addresses
- VLAN ID
- Ethernet type packets
- IP Protocol
- Source/Destination IP Address
- Source/Destination Port Number

4.3.1.3 FlowVisor Performance

The existence of a FlowVisor in the networks affects the whole network performance because it adds an element that needs to process a lot of traffic in the networks and also has to apply its own work (policing slice definition). The performance can be translated in how the different slices are isolated, how much delay brings the of a FlowVisor in a network, how the bandwidth of the system is affected by the FlowVisor or in what point the system is over headed and stops to work properly.
The FlowVisor does not add overhead to the data path since, packets are forwarded at full line rate; nevertheless, FlowVisor adds overhead to the control plane: control-level calculations like route selection proceed at their un-virtualized rate. FlowVisor only adds overhead to actions that cross between the control and data path layers. [11]
As an example, in the next figure it is shown the cumulative distribute function of virtualization overhead for OpenFlow port status requests.

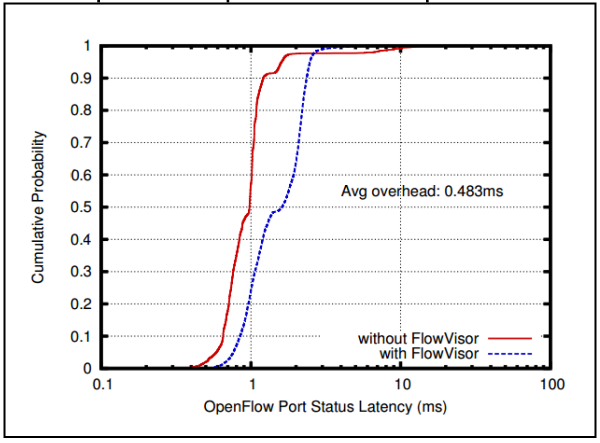


Figure 4. OpenFlow Port Status Latency with FlowVisor and without FlowVisor [11]

The figure above shows that the use of a FlowVisor in the network adds some latency due to the extra work done by the FlowVisor itself. The shapes of the graphs are maintained so the behavior is the same as working without FlowVisor but with a higher latency.
In summary, FlowVisor provides:

- FlowVisor defines a slice as a set of flows running on a topology of switches.
- FlowVisor sits beside between each OpenFlow controller and the switches, to make sure that a guest controller can only observe and control the switches it is supposed to.
- FlowVisor partitions the link bandwidth by assigning a minimum data rate to the set of flows that make up a slice.
- FlowVisor partitions the flow-table in each switch by keeping track of which flow-entries belong to each guest controller.
- Network packets with FlowVisor are forwarded at full line rate but overheads are addend in the control and data planes.

4.4 OpenFlow Workflow Summary

Once the all OpenFlow elements are presented, the lifecycle of a flow packet between a sender and a receiver can be explained in the next points:

- The sender sends the flows to the receiver.
- The flows arrive to an OpenFlow switch.
- The packet matching is made in the OpenFlow switch.
- If there are not entries in the flow table, a message to the controller is sent.
- The controller responds adding an entry to the switch flow table and to all the switches that are going to process the flows.
- All the flows are forwarded to the switches set up by the controller, and no messages are sent to the controller anymore.
- The flows arrive to de receiver.

5. Future Internet Testbeds

5.1 Overview

Future Internet Testbeds (FIT) have been implemented thought projects with the aim to provide a framework to the researchers in order to help them to test and experimentally validate new protocols or solutions for the future internet. These frameworks provide an environment where the researchers can flexibly use several programmable resources (Switches, Access points, Virtual Machines, etc.).

FIT projects, usually are deployed in networks with real traffic as campus networks; in order to provide the programmability required to research new protocols, FIT projects usually use Software Defined Networks (SDNs) and OpenFlow protocol, as explained in previous section of this master thesis. Such projects have been promoted in the framework of the FIRE initiative.

The aim of the FIT projects is to provide a framework to research but taking into account these requirements:

- Every researcher/group of researchers need a private environment to run their experiments isolated from the other, but with real traffic scenarios.
- FIT projects are platforms that try to be used as many researchers as possible; however, these platforms have limited resources, so a resource management is required.
- FIT projects need a well-defined permission and authentication engines in order to protect the experiments, the researchers and the whole platform functionality
- Since these frameworks provides a lot of flexibility and allows configuring a lot of parameters in a network, it is required to define a strict user admission in order to avoid malicious experiments.

5.2 Deployment of the FIT Projects

To achieve a good solution to the requirement mentioned above and to give to the users a good service, FIT projects usually applies the following concepts:

- In order to get an isolated environment for the researches, FIT projects should be able to provide a “piece” of the network where the users could allocate resources to be used exclusively for their experiment.
- A resource manager is required to control resources in different ways. The resource manager could assign directly the physical resources, virtualize different resources or even loan the resources with a certain expiration time and give them to the experimenters.
- To guarantee the privacy and offer a confident service to the users, a secure connection to the testbed and a reliable authentication/registration service is needed the use of pair keys (as Rivest, Shamir and Adleman (RSA)) and a good use of certificates (as X.509 certificates).
- Also FIT projects require a role based permissions in order to restrict and control the privileges of the users and establish a general manager authority

able to monitor the all the parameters of the project and take decisions depending of the state of them.

- And of course a user interface and a communication channel to interconnect the different modules and provide a way to the users to interact with the framework.
- In some cases, users with special privileges, users with more/restricted privileges compared with common users, could appear; in other cases, changes in the state of the resources are necessary, since most of the total resources could be busy and it can cause troubles to the experiments. To handle with these kind of situations, a good solution is to use a policy engine able to control, act and solve whenever possible, the problems that could appear daily and automatically.

5.3 Architecture

Most of the FIT projects are based on the architecture described in the figure below:

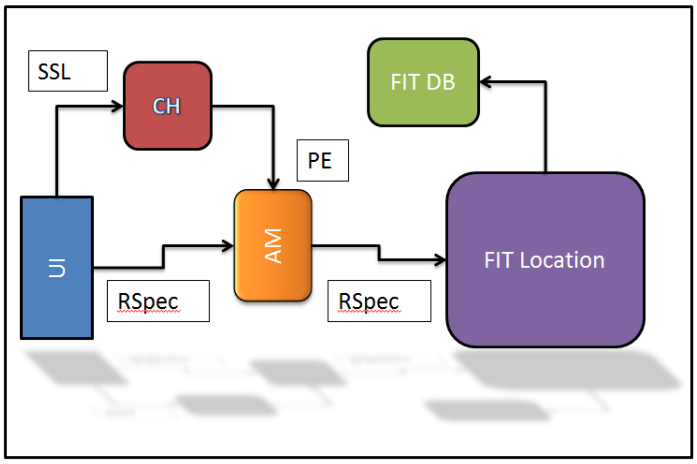


Figure 5. FIT Project Architecture

In the following, the main building blocks of the architecture are described in details.

5.3.1 User Interface

The user interface (UI) is where a user can login into the project, manage the account, request resources and make research with these resources. The user interface can be a graphical one using a web page or can be a simple command line.

5.3.2 Clearinghouse

The clearing house is the database where the users and their permissions are stored. When a user signs-up into the test, all the passwords, keys and credentials are saved in the clearinghouse. The clearing house is accessed via a secure channel, usually the connection User-Clearinghouse is made via HTTPS.

5.3.3 Aggregate Manager

The Aggregate Manager (AM) is the module which manages the physical resources and allows to the researchers to interact with them. There are different possibilities to interact with the resources (they can be used directly, generated by virtualization, loaned, etc.), so the AM should implement as many possibilities as possible taking into account the type of resource. For example, for an AM that offers switches could be a good solution to loan the switches for a period of time or offer only some ports of the devices. Since every AM has different type of resources that can be different from each other (in terms of implementation a VM provider AM is very different from another AM that offers Ethernet switches) AMs need their own communication language, the resource specification.

5.3.3.1 Slices

A slice is a “piece” of network of computing and communication resources capable of running an experiment or a wide-area network service. Slices are the primary abstraction for accounting and accountability. The resources that are acquired are consumed by slices, and external program behavior is traceable to a slice.
Slices contain a set of resources plus an associated set of users that are allowed to access those resources for the purpose of running an experiment.

5.3.3.2 Slivers

Slivers are the minimum “piece” the physical offered resources by an Aggregate Managers. Each AM must include hardware or software mechanisms that isolate slivers from each other, making it appropriate to view a sliver as a “resource container”.

5.3.4 Resource Specification

Resource Specifications (RSpecs) are XML files that describe the resources in an AM and to reserve these resources to the researchers. With RSpecs an AM can describe the available resources, list the created slices, list the allocated slivers, create slices and allocate slivers.
Since every FIT project is different due to the resources offered there is not a standard RSpec definition, so every project has their own RSpecs version that only can be understood by their own AMs.
There are three types of RSpecs:

- Advertisement: RSpecs that informs of the state and the availability of the AM resources.
- Request: RSpecs that allocate slivers in a certain slice
- Manifest: RSpecs that responds to the request RSpecs in order to inform what the resources were allocated in the slice.

5.3.5 Testbed Infrastructure

The testbed infrastructure is the physical layer where the testbed is located; here a database is required in order to store all the required data about the resources, slices, the actual state of the testbed, etc. Also, there are interfaces for the communication with the UI and the AM.

5.3.6 Policy Engine

The policy engine is a policy manager that contains a list of Accept/Deny rules that automatically manage user's requests without intervention of any person related with the project management. These policies can help the whole testbed to restrict the services offered in order to not collapse all the available resources with few users or absorb minor management tasks from the global manager of the Testbed.

5.3.7 Roles and Permissions

FIT projects have three different roles with different permissions to provide a different set of permissions:

- Island Manager: Is the responsible of the whole testbed; it manages all the resources, defines policies and accept or deny users requests. He has root permissions for any element of the testbed.
- Principal Investigator: User that research in the testbed, he can manage all the slices related with their research.
- Researcher: Are users that assist the principal investigators, they usually have less flexibility to request resources or use them.

The permissions are a set of rules that allows to a user have more flexibility to take decisions or to run experiments over the testbed. Some examples of permissions are to create slices, allocate slivers, access to a set of certain resources, etc.

5.4 FIT Project Lifecycle

As was said before, a FIT project is a framework to offer resources for researchers in order to experiment in future internet protocols or solutions. In order to make an experiment, a principal investigator should follow the next steps.

5.4.1 User Registration

In order to access to the framework and request resources, a user must be registered. The registration forms for new users require a lot of personal information and usually a top level accreditation (be in a research group/company, be in a certain university, for example.).
After a user is registered, he has not access immediately, first a message to the top level accreditation entity is sent in order to verify that the user is a proper user to use the framework. Then the Island Manager decides personally if the user can be registered in the project or not.
If the user registration is done without problems, a key-pair is generated or uploaded for him.

5.4.2 Creating/Managing Slices

Once the user is registered, he can start to use the testbed infrastructure. The first thing he should do is to create a slice:
Using the testbed interface he creates a slice giving to the slice a name and signing the slice with his private key.

5.4.3 Allocating Slivers

Now the user can request a list of the available resources in order to decide what resources can be allocated in his slice.
The request of the available resources is also signed with their private key and sent to the clearinghouse. The clearinghouse checks the origin of the user in its database and gets the permissions for that user. Then the request of the resources is sent to the AM, with the permissions and the AM returns an advertisement RSpec to the user.
Once the user receives the RSpec, he can write a Request RSpec in order to allocate resources to his slice. Again the clearinghouse receives the request to allocate slivers and checks the user origin and their permissions and forwards this to the AM. The AM respond to the user request with a Manifest RSpec, so the user can now use the resources granted by the manifest RSpec.

5.5 FIT projects Security

In FIT projects the security is a very delicate topic since the goal of these projects is to offer resources to researchers and let them to do whatever they want.
This is the main reason to restrict the user registration in these projects. Having a manual user administration and a third authority's reliability that ensures that the user that wants to be registered in the FIT project will do not do any malicious practice and in the case that the new user will do a malicious practice, the third authority will be the responsible one.
Other next problem, would be the identity impersonation of a researcher/PI solved by the use of key pairs and signing the requests to the AM or clearinghouse via a secure channel. The key pair is usually a RSA key of 2048/4096 bits and the secure channel is a SSL one.

5.6 Federation

Federation is the service offered to the users of a certain FIT project to get resources from other FIT project/s. For FIT projects, be federated means that there must be a complete understanding between the federation partners, they should share an API and "Speak the same connection language" in order to be connected and be able to share their resources and of course, these two partners mast trust each other.
Usually, the resources requested from users of another project, can have more restricted policies of use or can be more limited.
Federation is a powerful tool that allows to the researchers to get a lot of different kind of resources in a single slice in order to get a very homogeneous set of resources to improve the quality and the conclusions of their researches.
Every FIT Project has defined their methods, their secure channels, their RSpecs and their user's management since there is not any standard created that defines how to implement these mentioned points.
Due to the lack of the standards, federation is very difficult and there are few FIT projects able to offer federation, and even fewer ones that can accept the federation from other ones. Federation is only possible to those projects that have similar structure and of course similar RSpecs management, but in this case there is not any security between projects.

6. OFELIA Control Framework

OpenFlow in Europe: Linking Infrastructure and Applications (OFELIA) or Ofelia Control Framework (OCF) is a FIT project which its main objective is to automate, simplify and authorize users to create projects and slices and to allocate resources inside each slice in order to experiment with these resources. The main resources handled by OCF are OpenFlow resources and provision virtual machines.

Ofelia Control Framework is divided by several islands individually controlled but interconnected offering different kind of resources but with similar capabilities. The structure of every island is the same.

6.1 Architecture

An Ofelia control framework island has three main modules, Expedient which is the main resource management model, the OpenFlow Aggregate Manager (OFAM), the aggregate manager responsible to provide OpenFlow resources and finally the Virtualization Aggregate Manager (VTAM), responsible to create and provide virtual machines.
The whole island runs over the apache [12] server, using the Django Web Framework [13] and programmed using python language.

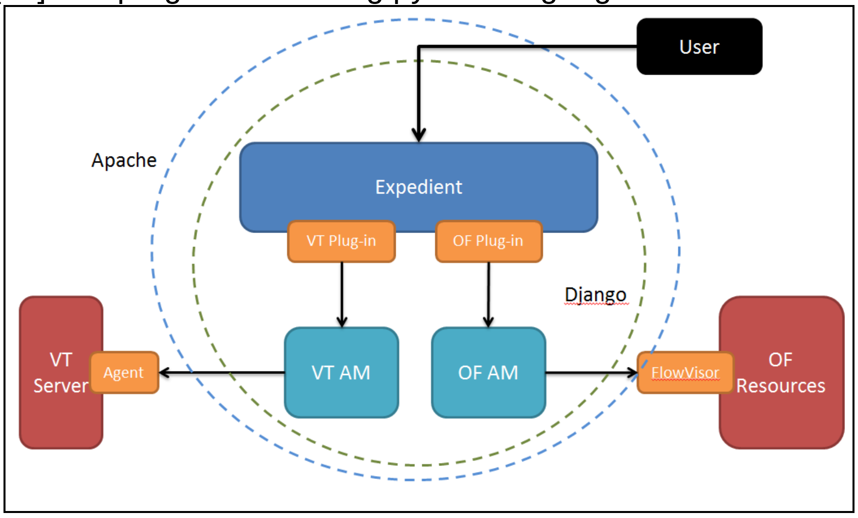


Figure 6. OCF Island overview

6.1.1 Expedient

Expedient is a pluggable centralized control framework that has been developed by Stanford University [14]. This module has been implemented as a Web application using Python programming language and Django Web Framework.
There are two main purposes to be completed by expedient. One is to provide the tools to easily build a plugin for their resource types and the other is to provide an environment where projects, slices and user can be managed.
By default, Expedient offers two types of plugins for OCF, the Virtualization and the OpenFlow plugins. These plugins are the ones which communicate with its respective aggregate manager.

6.1.1.1 User Management

Regarding user management and authentication, expedient has its own database (clearinghouse) to save authentication and persistent information. Users authenticate themselves against Expedient and Expedient authenticates itself against the Aggregates Managers databases. The communication between the user and the resources to be managed are communicated through XML-RPC protocol.
There are three main user roles:

- Island Manager: The root user of the control framework, he has permission to run every module without restrictions. The Island Manager has the task to control the state of the whole framework, add/delete expedient plug-ins and guarantee some user's special requests.
- Owner: The owners are users that can create projects and that has permission to do everything in their project. They also can give the permissions to everyone. In addition every time a slice is created, these users get full permissions over those slices.
- Researcher: the researchers can only create slices and delete slices they created having full permissions over their created slices.

6.1.1.2 Project Creation

A project is an environment managed by the users where they can create/allocate their slices and also other users. In a project the owner has to select different aggregates managers to be used among the slices, create slices and manage the slices.

6.1.1.3 Slice Creation

The OFELIA Control Framework slices follow the same concept explained in the FIT projects architecture but with a particularity: Since OCF can manage different types of AM, the user who has created a project and selected the aggregate managers he want to use, has to select the AMs that he wants to get resources for each slice that he creates.
This approach is done in order to not confuse the users providing them a lot of resources types when are searching for a specific type resource.

6.1.1.4 Federation

OCF provides an easy way to federate aggregate managers due to the pluggable capability of expedient. To add aggregate managers, there is only a simple form to be replenished where the URL has to be provided.
Since the communication between Expedient and the AMs is very specific, the possible AMs to be federated are the only ones that are based in OCF.

6.1.1.5 Virtualization plug-in

The virtualization plug-in is the one interacting with VTAM manager; it is constructed by containing a API able to understand and control the AM and connected with an XML-RPC communication interface.

6.1.1.6 OpenFlow plug-in

The OpenFlow plug-in is the one managing the OFAM, this plug-in shares API with this AM. This plug-in is also connected using an XML-RPC communication interface

6.1.2 Virtualization Aggregate Manager

VTAM is an Aggregate manager that provides virtual machines to the users and an interface to control them via expedient or SSH. This aggregate manager is able to create virtual machines with the parameters selected by user (RAM memory, CPU speed, operative system, etc.). VTAM has also a system to set up IPv4 interfaces and MACs to be included in the VMs in order to offer to the users a way to get into those VMs using the Secure Shell (SSH).
The VTAM is controlled by the expedient. The users can start, stop, delete, reboot and create virtual machines by a single click via the graphical user interface provided by a virtualization plug-in.
The virtual machines are created using para-virtualization from different servers located in every Island. These servers have a high amount of resources in order to support the creation and maintenance of several VMs.
The VTAM is not the one that creates the virtual machines, the VTAM is the responsible to offer, configure and provide the necessary tools to create virtual machines (Provisioning actions) and also to get in every moment the state of the existing virtual machines (Monitoring Actions). The one that makes possible the creation of virtual machines is the Agent.

6.1.2.1 Agent

The Agent is the module that handles the virtual machine creation. This module is controlled by the VTAM and serves as interface between VTAM and a virtualization server. One Agent only controls one virtualization server, but the VTAM can control as Agents as needed.
The Agent is able to create virtual machines thanks to the XEN [15] hypervisor and the libvirt [16] library and its python bindings. This library, allows create virtual machines from a configuration file in XML format. The communication between the VTAM is done using XMLRPC communication and the VMs are created using a particular RSpec configured by the VTAM and only understood by the Agent.

6.1.2.2 VM creation

When a user creates a VM, first he has to fill a form with some parameters of the VM (as RAM memory, Operative System, etc.) in the Expedient GUI. The system gets the form information, encodes this information into an XML file and sends it to the VTAM through the VTPlug-in. At this point the VTAM validate this information and configures an interface where the user could access to the VM when created, mounts an RSpec and sends it to the Agent. When the Agent receives the RSpec, first, validates it and sets up the configuration file required by the Libvirt library. If no errors occur within this process, the Agent sends an ongoing message to the VTAM and starts to create the VM. The ongoing message received by the VTAM is forwarded to the expedient and seen by the user as a loading icon. When the VM is finally created, then, a successful message is sent to the VTAM and forwarded again to the Expedient. This time the user sees a Start and a Delete links to control the VM.

Since the creation of a VM requires a lot of time (up to five or seven minutes) the connection between the Expedient, Virtualization Aggregate Manager and the Agent is an asynchronous connection.

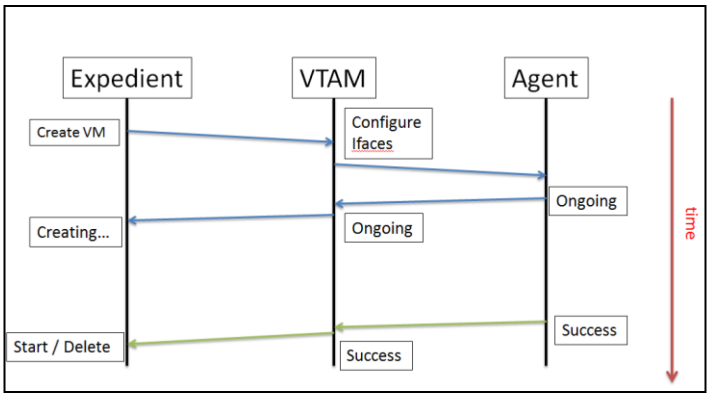


Figure 7. VM creation process

6.1.2.3 Provisioning Actions

The provisioning actions are defined as a set of commands that the VTAM and the Agent can understand in order to control a virtual machine. This type of actions is a human-readable kind of actions. The available actions to be done by the VTAM/Agent are:

- Create (VM): Creation of a VM. This action requires a set of parameters in order to configure the VM.
- Start (VM): Start a VM, equivalent to press the on button of a PC.
- Stop (VM): Stop a VM, equivalent to shut down the PC.
- Reboot (VM): Reboot a VM, equivalent to press the reset button of PC.

These actions are executed by the user who created the VM, or the ones who have permission to act in the slices where the VM is located.

6.1.2.4 Monitoring Actions

Monitoring actions are defined as a set of human-readable messages that shows the current state of a VM. The monitoring actions are not controlled by the users since its aim is only to provide information to the user if anything was wrong with the VM or there is not any problem. There are the following possible monitoring actions of a VM:

- Ongoing (creation): This message informs that the requested VM is being created.
- Created + Action (start/stop): This message informs that the VM is created and is already started or stopped.
- Failed (VM): This message informs that the (VM could not be created and usually provides more information as an exception of the code.
- Unknown (State): This state shows that the VM was working, but something happened with the connection between the VTAM or the Agent.

6.1.2.5 Virtualization Servers

The Virtualization Servers are the physical hardware that stores the virtual machines. They are also connected physically to several switches through their interfaces. Some of the interfaces of the VMs are bridged into the virtualization servers interfaces in order to offer to the user connectivity to the switches improving the possibilities of the researchers to prepare more easily experiments.
In order to work as VTAM Virtualization Servers, they need to run the Agent daemon provided by OCF that interact as an interface between the VTAM and the physical resources.

6.1.2.6 VTAM and Island Manager

Since the Island Manager is the responsible of the whole island, he can interact with the VTAM. The island Manager can create or delete Agents by adding the URL and the port of a virtualization server. The IM can also create, program, exclude or reserve IPv4 or MAC ranges provisioned to the VMs and control, locate and monitor any created VM by the VTAM.

6.1.3 OpenFlow Aggregate Manager

The OpenFlow Aggregate Manager (OFAM) is an AM that provides OpenFlow switches with physical topologies to the users. This AM allows OpenFlow experimentation to the researchers by allowing them to set a controller (via a created VM of VTAM for example) select a topology of switches and giving access to them.
This aggregate manager is controlled by the expedient. But the requests in this case are not as fast as in VTAM, since the Island Manager has to approve personally any resource and topology request. As is said before, this AM provides OpenFlow resources and in order to slice these type of resources for now only exist the Flow Visor as a high performance tool.

6.1.3.1 User Slicing

The only way to separate different OpenFlow network slices is using the FlowVisor controller, so the OpenFlow AM needs a three step resource offering form to the users.

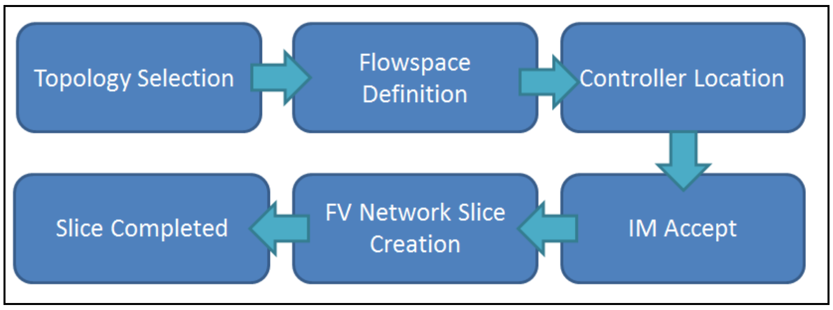


Figure 8. OFAM Slice Creation Steps

First, the users have to select the available OpenFlow switches and links from a topology engine. The second step is to select a FlowSpace where the user is going to run their experiments and finally provide the location (IP + port) of an OpenFlow controller.

When the user finishes the required steeps, the island manager receives an alert with the details of the FlowSpace; then after the review of the parameters set by the user, he grants or not the FlowSpace. It exist the possibility that the Island Manager modifies the requested FlowSpace in order to make possible the co-existence of different FlowSpaces. For example a user could request a similar FlowSpace demanding the IPs 10.0.0.1/8 and this FlowSpace already exists; in this case the island manager could provide a FlowSpace with the IPs 10.0.1.1/8.
Once, the FlowSpace is granted by the Island Manager, is introduced to the FlowVisor and the FlowVisor provides the network slice to be used by the user.

6.1.3.2 Slice Control

The OFAM slice control is managed by the expedient, once an slice is granted and created by the FlowVisor, expedient act as a trigger to start or stop the slice by clicking a button. Expedient offers also the possibility to update the slice, but this process rewrite the current slice in a new one and needs to follow the slice creation process again.

6.1.3.3 Physical Resources

The physical resources are the OpenFlow switches, these switches are connected to other switches forming a topology, the users can access to these switches thanks to the FlowVisor job. The possibility to the users to manage a switch starts from a single port to control the whole switch. It has to be noticed that the users that request smaller slices have more probability to have a granted FlowSpace since it give more flexibility to the Island Manager provides a similar FlowSpace that require less rules.
Some switches, also are connected to other switches from other OCF island allowing to create a bigger and multi-resourced island. The users that request slices using resources from different islands need the approval of the FlowSpace from every Island Manager.

6.1.4 Island Manager's work

The main task to be accomplished by the island manager is basically to grant FlowSpaces. Once a new slice is created by a user and pendant to approve, the Island manager receives an alert through email. At this point the Island manager is able to accept, deny or modify the requested FlowSpace. Other tasks to be done by the island manager are the FlowSpace slices management and monitoring. The IM can monitor every granted FlowSpaces around the island and view the state of the OpenFlow resources.

6.1.5 OCF Limitations

OCF is FIT testbed that has some limitations that must be corrected. One of the aims of this master thesis is to add features to framework or present solutions to correct these limitations and improve the general quality of the testbed for the users and for the Island Managers.
In the FIT projects section of this thesis, one of the architecture modules presented is the policy engine. Ofelia as FIT project does not have any policy engine to limit, restrict or extend the usage of the resources for the users in determinate conditions so there is not an easy way to manage the resources for the IMs. With a policy

Engine, the IM could control the resources in a better way, restricting or extending the resource usage depending on the actual testbed conditions.
The users can create and access to several VMs, but there is no monitoring instance that controls these VMs and reports the user when a VM has been stopped, rebooted or crashed without the user control. A monitoring action for the VMs would be a powerful tool for the users in order to know exactly what is happening with their VMs and to ensure that the VMs are run correctly.
Ofelia is a popular FIT architecture due to the pluggable capabilities of Expedient. There have been some projects around Ofelia based on the architecture but with a similar core. Since every FIT has its own styles, images, logos, etc. OCF needs a system to change the whole OCF look and feel easily, transparent to the core functions and without modifying lots of files.
Due to the several connections between every switch in OFAM, it is possible that a user selects a topology with loops making their tests useless. For this reason a loop detection procedure should be done after the topology selection by the OCF.
The Ofelia federation principle is very limited due to the OCF; in fact, it is only able to federate with Ofelia based testbeds. A new approach to extend this limitation is required in order to federate with any kind of testbed without limitations.

7. Deployed Solutions to Solve OCF Limitations

7.1 Pypelib and PolicyEngine

PyPElib is a small library to help programmers to use abstractions to build rule-based policies within a certain scope of action.
These policies are used in the VT Aggregate Manager to automatically manage user's requests without intervention of the Island Manager. Resources requests are evaluated against the defined rules and they are accepted or denied. PyPElib has also the possibility to trigger actions depending on the results of the policies evaluation.
The library allows the Island Managers to define policies for their islands based on the resources requested (maximum number of VMs, RAM memory, hard drive space, etc.) but also based on other parameters such as users origin (to restrict or not parameters to users from other islands or authorities) [17].

7.1.1 pyPElib structure

7.1.1.1 RuleTable

This is the main instance. It contains a set of rules with a common scope. The rule table has a default ACCEPT/DENY value when no rules to evaluate are found.
RuleTables are basically containers of rules with a certain value (to accept or deny the request) when all the rules are evaluated without a deterministic value.

7.1.1.2 Rule

This is the instance used to represent a specific policy. There are two types of rules, the terminal and non-terminal ones.

- Terminal rules: When the condition of this rules is matched, the main workflow is interrupted, returning the ACCEPT/DENY value of the rule.
- Non-terminal rules: These rules do not break workflow when are matched, but they can trigger the actions.

Rules are RuleTable entries in human readable format with the possibility to interrupt the main workflow (by denying or accepting automatically the users' requests) or trigger actions when conditions are matched.

7.1.1.3 Condition

The conditions are the instances to be evaluated of each rule. The conditions can be simple (*LeftOperand, SimpleOperator, RightOperand*) or complex (*LeftCondition, ComplexOperator, RightCondition).*
Conditions in principle are logical operations to be evaluated in every Rule. For example a typical condition could be A >= B, where A and B are single values or other conditions.

7.1.1.4 Mappings

Mappings are structure instances which relate the condition Left Operands or the actions with the location of the value to be able to compare with the Right Operands. The keys of the mappings are the conditions left operands and the value of the mappings is the path to the function to retrieve a value to compare.

7.1.1.5 Resolver

Resolver is a class used to obtain the value (string, integer, Boolean, etc.) or run a function (if the mapping key is an action) from the mappings.

7.1.2 PyPElib Persistence Engine and Parser Engine

One of the goals of pyPElib is to give flexibility to the users, to accomplish that, two backend engines were designed: the persistence engine and the parser engine.

- Persistence Engine: This class is able to handle different type of persistence drivers, these drivers are used to store different rules and rule tables.
- Parser Engine: This class handles the different type of parser drivers. These drivers are used to instantiate rules from different syntaxes or input types.

These two modules provide the flexibility for the pyPElib users to configure how to store the rules (types of databases, if any) and how to write these rules to be stored (human readable text, XML, etc.).

7.1.3 PyPElib Current Persistence drivers

In pyPElib-1.1, there are two persistence backends: Django Backend and the RAWFile Backend.

- Django Backend: Since pyPElib was developed as a tool for Ofelia Control Framework this was the first persistence driver done. This driver consists in two Django models one for the Rules and the other for the RuleTables.
- RAWFile Backend: This driver was developed for non OCF users. This driver consists in store RuleTables in files serializing the RuleTable instances.

7.1.4 PyPElib Current Parser drivers

The rules have a human readable structure, in order to help the much as possible the users to construct their own rules, the RegexParser Driver was developed.

- RegexParser Driver: Is a backend to parse rules from a string input with a certain structure required. This driver works matching groups inside the rules text using regular expressions.

The rules required structure to use the RegexParser is the following:

If *[condition]* **then** *[accept/deny]* *[nonterminal]* **do** *[action]* **denyMessage** *[Error Message] #[Description]*

Due to the simple structure of the rules, multiple parser drivers could be developed, for example an xml parser.

7.1.5 Pypelib Work Flow

The pyPElib workflow is can be separated in different steps:

- RuleTable creation: In this point an empty RuleTable is created, a name, the mappings and the parser and persistence drivers are required in order to properly construct this instance.
- Rules Addition: Some rules following the structure of the parser driver are added in the RuleTable.
- Evaluate Trigger: The point where the rules will be evaluated is setup with the RuleTable evaluate and the resolver over the evaluate method.
- Rule Evaluation: Pypelib checks the inputs in the evaluate trigger and raises exceptions for the denied rules.

7.1.6 PyPElib integration into OCF

Pypelib is an external library made for Ofelia Control Framework but ready to use for any application that requires policies. To integrate this library in OCF, the following works were done:

7.1.6.1 Policy Engine Logger

Module that prints the pyPElib exceptions, information about the rules and the debug messages to an external file where the pyPElib managers can monitor the state of the requests.

7.1.6.2 Controller Mappings Interface

Module to manage all the mappings required. This interface provides the main mappings to control all the VM parameters with their respective instances to be resolved. Controller mappings also provide some methods to achieve values for some parameters, as the number of interfaces of a VM or the project VMs number.

7.1.6.3 RuleTable Manager

Interface to manage pyPElib methods and the Controller Mappings. This class also provides extra methods to do Create, Read, Update and Delete (CRUD) functions as a function to edit rules (not provided by pyPElib).
This module is developed to simplify the interactions between Ofelia code and pyPElib in order to create more easily the Rules. For example, RuleTable Manager implements the function UpdateRule, done by deleting the old rule, and introducing a new rule with the updated parameters in the position of the older Rule.

7.1.6.4 PolicyEngine GUI

Using the VT Aggregate Manager graphical user interface, a new module was done providing an interface to interact as easy as possible with pyPElib. This GUI creates automatically a provisioning rule table with the objective to manage the virtual machines specifications selection from the users.
In this page, all the rules from the provisioning rule table are displayed, showing the rule a checkbox to enable or disable the rule and a button to delete the rule. Every rule can be clicked going to a page to edit any parameter of the rule. Also there is a

button to change the default policy type of the rule table and a button to create new rules.
In the rule creation page, there is a form to create the rules in two ways: The easy mode and the advanced mode.

7.1.6.4.1 Easy mode

The easy mode provides a form interface to put every part of the rule separated. The simple conditions are created separately in a fancybox [18] where only the available mappings and the possible operators can be selected. The right operands are typed by the user with input text types.
When the simple conditions are created, there is a drag and drop zone where the conditions can be grouped with the complex condition operands and construct complex conditions very easy. The drag and drop zone is programmed in javaScript using the Redips [19] library.
The other parameters required by a rule are set via input text types or select options types. There are multiple tooltips in several fields in order to help as much as possible to create well-formed rules

7.1.6.4.2 Advance mode

The advance mode provides a single input text type to write the rule using the Regex Parser Driver syntax.

7.1.6.4.3 Evaluation Point

The point where all the rules are evaluated and act if is necessary, is in the Provisioning Dispatcher, the class that handles the provisioning communication between the Agent and the VT AM, before mounting the VM RSpec and send it to the Agent.

7.2 Theme Manager

Due to the high popularity of OCF, new testbeds based on the OCF architecture has been deployed as the case of FIBRE-FP7 [3] or Géant [20]. Mainly, this new testbeds are Django based facilities with the three main modules (Expedient, OFAM and VTAM). Since this facilities require a different layout with respect to from Ofelia, but with small changes in the HTML pages, was required a tool to change the main look and feel of Ofelia but maintain its core functions.
Theme manager allows to use a set of custom static content files (images, CSSs, javaScript files...) and/or templates (HTMLs files) creating a custom theme or look for OCF without overwriting or substituting any file of OCF and taking advantage of what is already there. These custom files must be named equal than the OCF main files that are replacing.
This entire job is done by overwriting the template tag URL. A template tag is a python variable (Dictionary, Object, List, String, etc.) that can be used in a HTML template allowing flexibility, programmability and adaptability.

7.2.1 OCF Static Content and Templates Overview

7.2.1.1 Static Content

Ofelia Control Framework's static content is distributed in the three main modules (VTAM, Expedient and OFAM). Every module has its static content distributed in static paths configured in Django settings.
Over the static content directories, there is a folder called default. Here there are all the images, CSS and javaScripts main files of OCF. Usually the static content is organized in subdirectories separating the different static content files type.

7.2.1.2 Templates

The Django templates have a similar static content distribution, also separated in the three main modules and configured via Django settings.
In templates directory there is also a folder called default. Here there are all the templates, some distributed in different subdirectories and the others in the default directory.
Theme Manager takes advantage of static content and template directories disposition, organizing the files in media path or template path in a new folder named with the name of the theme with the same structure than default. The structure of the theme folders must be equal than default only for the customized files.

7.2.2 Working Principle

When a file (static Content file or template) is loaded, Theme Manager looks for the file in the custom theme directories, if the file is found, Theme Manager will serve this custom content; if not, Theme manager will serve this file from OCF default directories.

7.2.3 Theme Manager Usage

To setup a new theme, it is only required to follow this steps:

- Set the Django settings variable THEME equal to the name of the theme in every module.
- Create a directory named as theme variable in media path (static content location) and create the required subdirectories in order to maintain the structure of OCF.
- Create the directory or directories named as theme variable in template paths (HTML files location) and create the required subdirectories order to maintain the structure of OCF.
- Add the custom files to the theme directory respecting the structure and the name as in OCF.

7.2.4 Theme Manager Implementation

When the server (Apache) starts and Django loads all the internal modules using the settings, Theme Manager launches an initialization call, creating the static content URLs in Django URLs and adding the theme template directories into the Django template directories setting.
The functionality of the template tag URL is changed in the following way:

When this template tag tries to load a static content file, Theme Manager first checks the location of this file in the theme URLs, if the file is in the theme URLs, Theme Manager loads the file, in the negative case, Theme Manager loads the file from the default URL.

7.3 Libvirt Callbacks for VMs

The OCF users and the Island Manager required more monitoring functions of the VMs. The main function required is to know if a created or running VM crashes due to external facts (problems in the Agent, errors with XEN hypervisor, etc.), however, due to the communication between the VTAM and the Agent there is no simple way to get the information of all the VMs at real time. The best solution requires a callback system on the VM side.
The communication between the VT AM Manager and the Agent is an asynchronous connection. Due to this type of connections there are some limitations in order to have a complete Monitoring of the VMs in the Agent.
In order to get all the running VMs in the Agent, VT AM Manager sent periodically a List Active VMs call. After this call the Agent response with a RSpec listing all the VMs that were started. Comparing this RSpec with the VMs list that VT AM Manger had, the down VMs could be obtained and act in consequence.
This monitoring only could act over the started VMs and the information was not instantaneously due to the List Active VMs call is done with a certain periodicity.
In order to improve the communication between the Agent and VT AM Manager and the monitoring done over the VMs a Livbirt Monitoring Module was designed.

7.3.1 Libvirt Monitoring Module

Libvirt Monitoring Module uses the libvirt [16] library, starting a daemon that registers a callback function triggered by the VM lifecycle when a VM is created in the Agent. The aim of this module is to register a function in all the VMs allowing to all the users know the status of s VM in real time, improving the monitoring functions of OCF.
The callback function mounts an RSpec with the new status in the Agent and sends it to the VT AM Manager Monitoring Dispatcher, Module that handles the state messages in VTAM. At this point the status of the VM is updated and the RSpec is forwarded to the Expedient updating the VM state too.

7.4 Loop Detection Algorithms over OF Topologies

Developing new solutions for the future internet can be a hard work. The required development and experiment process to get good results is large and typically slow. Sometimes researchers develop algorithms to be tested in specific conditions, but when a researcher requests a FlowSpace to run his experiments in these conditions, the researcher himself could be confused and select an unwanted topology containing. Since the appearance of loops in a topology can result in bad results for the experiment, the topology selection of OCF requires an alert when a user selects a topology containing loops.
The available links that a user selects constructing a topology using an OF aggregate manager and the connection with a server can contain loops. The next step after the links selection is to request a FlowSpace with the topology selected.
The problem is that due to the extension and the flexibility of the OF aggregates the users can request a containing loops topology without notice that.

To solve this issue, a topology loop detection algorithm was done, showing an alert message when the topology requested contains loops, indicating to the user if he wants to continue or not.
The algorithm is based on a Depth First Search (DFS) [21] search and the Trajan's [22] algorithm adapting it to the needs of the OCF topologies. This algorithm, first constructs the requested topology in a structure instance creating a map. Then, a pointer follows the map and checks how many times will have this pointer to pass through every node. If the pointer arrives one time more than the supposed at the same node, a loop is detected.
This algorithm has to deal also with the possibility to have one or more sub-topologies in the input, and check the loops of all these sub-topologies.

7.5 Federation improvement for OCF

OCF federation is limited to only OCF based testbeds, using Django python language for its deployment. However federation is an important tool and one of the aims of FIRE initiative. Therefore, OCF needs to be extended to support the federation among available testbeds.
Federation basically means to trust other authorities and the users of said authorities. This implies to generate trust using certificates and establish secure connections between the authorities. This work requires a lot of elements, such certification authorities, certification creation and revocation and secure connections management. Also it has to be noticed that every authority has to be able to understand also the foreign RSpecs, and this work has to be implemented in every authority that wants federation.
Since the required work is huge, Slice-based Federation Architecture (SFA) presents a solution for federation, trying to standardize the federation procedure and automatically manage the RSpecs and the trust between authorities. Since SFA has been implemented in many projects as MySlice [23], Federica [24], Novi [25], Nitos [26], etc. it is the best solution for OCF.

7.5.1 Slice-Based Federation Architecture

7.5.1.1 Federation Problems

As it was commented before, federation is a powerful tool that allows to the researchers to get a lot of different kind of resources in a single slice in order to get a very homogeneous set of resources to improve the quality and the conclusions of their experiments.
Every FIT Project has defined its methods, its secure channels, its RSpecs and its users' management since there is not a standard created that defines how to do the points mentioned above.
Due to the lack of standards, federation is very difficult and there are few FIT projects able to offer federation, and even fewer ones that can accept the federation from the other ones. Federation is only possible to those projects that have similar structure and of course similar RSpecs management, but in this case there is not any security between projects.
To solve these problems, Slice-Based Federation Architecture (SFA) was designed. SFA is a minimal set of functionalities a testbed can implement in order to enter into a global and interoperable federation. An experimenter in an SFA-based environment

can transparently browse resources on any federated testbed, and allocate and reserve those resources.
Because of the potential for a very large number of testbeds, global federation architecture faces a serious scalability issue. SFA introduces a fully distributed solution in which each peer testbed serves as the authority of reference for the resources that it brings, and each user community, along with its experiments, is represented by an authority (possibly, but not necessarily identified with an individual testbed).

7.5.1.2 SFA Principles

Under the SFA architecture, there is a separation between what is generic and what testbed-specific is. Testbed-specific information is captured in a resource model, called a resource specification (RSpec), which is an XML transported by the SFA layer. SFA itself does not cover such aspects as resource model, policies, reservations or measurements. These functionalities are implemented by the FIT project.
SFA can be the structure of a whole testbed (adding the policies and the other requirements) for new FIT projects or can be used as plug-in for the existing ones.

7.5.1.3 SFA Architecture

The main structure provided by SFA is in the following figure:

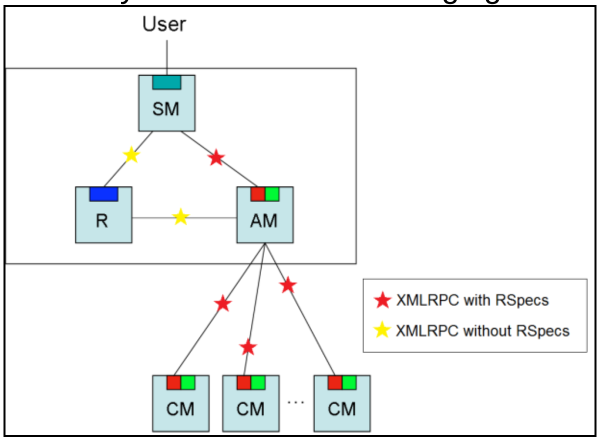


Figure 9. SFA Architecture [27]

SFA has defined four main entities to control the whole testbed: Registry, Slice Manager, Aggregate Manager and Component Manager

7.5.1.3.1 Registry

The registry is a typical clearinghouse with enhanced features: the registry is synchronized with registries belonging to federated testbeds.

7.5.1.3.2 Slice Manager

The Slice Manager (SM) is an entity connected to the registry and the interface that manages the slice creation and distributes the user requests to the Aggregate Managers (AM). In a SM several AMs can be connected, the AM belonging to the own testbed and the federated ones.

The SM and the AM share a common API and are connected via an XMLRPC interface. The SM is able to forward a RSpec to the AMs which can understand it.

7.5.1.3.3 Aggregate Manager

The Aggregate Manager (AM) is the interface that interacts with the testbed, this interface access to the resources directly or through a *component manager*. The AMs are connected to the Testbed and to the SM (sharing a common API)

7.5.1.3.4 Component Manager

Components Managers (CM) are a lower interface that are only able to handle one type of resource. For example, if a FIT project has an Aggregate Manager that provides different kind of switches, could be two CMs, one providing Ethernet switches and the other optical switches.

7.5.1.4 SFA Elements

In SFA there are a set of elements that should be defined in order to understand the whole SFA functionality.
Authority: Is the top level entity of the testbed, it is referred usually to the name of the testbed and has its own certificate.
Sub-authority: is the entity below an authority or another sub-authority, the sub-authorities containing a certificate chain until arrive to the authority.
The slices and the roles are the same as the FIT projects, an also have a certificate chain referenced to the authority.
The notation in SFA refers to the Human Readable Names (HRNs) with the certificate associated with the HRNs is called Global Identifier (GID) [27].

7.5.1.5 Authentication Certificate, Global Identifier

SFA currently bases its authentication mechanism on a public key infrastructure, where each object has a key pair (a public and a private key) and is associated with a signed certificate, called a Global Identifier (GID) that is stored in the registry. The certificate is used for authentication, following the same principles as user and website authentication on the web. It is a X.509 certification [28] that associates the object's HRN with its public key, and that is signed recursively by each parent authorities up to the root.

7.5.1.6 User Registration

A new user is supposed that possess a key pair. A bootstrap procedure (such as user registration) is necessary to that its home authority knows its public key, and is able to associate it to its HRN.
From its key pair, the new user can generate a self-signed certificate it will use for the SSL connection. This is enough for the authority to authenticate the user since it knows his user account. This allows the user to retrieve its GID by contacting the registry of its home authority.

7.5.1.7 Authorization Generalities

For the authentication there are many possible ways to perform authorization in a distributed environment. For instance, in A there is authenticated a user of testbed B and he might also want to get authenticated and authorized by testbed C, which might never have heard of A or B, but trusts them indirectly because C trusts their root authority.

7.5.1.8 Credentials

SFA use their permission management using credentials, which are a signed XML document that proves that an entity has a set of rights relating to another one, and states whether or not the first entity has the possibility to delegate those rights. Such credentials can be used to establish the various trust relationships necessary to run a federated platform. A credential stores the following information:

- Caller id: identified entity to which the credential has been issued, characterized by its HRN and GID. Most of the time, the caller is a user (or an authority).
- Object: identifies the object for which the credential holds. The type of the object determines the type of credential: user credential, slice credential or authority credential.
- Expires: a credential is issued for a limited lifetime.
- Privileges: a set of privileges that are assigned to the caller with respect to the object,
- Delegate: each privilege is annotated with a flag indicating whether it can be further delegated.

7.5.1.9 Credential Delegation

The delegation mechanism has been implemented in SFA in order that a user could perform actions for which it has been delegated the rights, on behalf of another user. A delegated credential has the same structure as a normal credential. It also encloses the original credential, proving that the original user has both the delegated privileges and the right to delegate them.

7.5.1.9.1 Credential verification

The credential verification is done to check the credential against the with an authority (GID included in the credential), check the connections, if the caller is connected via HTTPS connection, check if the credential was signed by a trusted cert and check if the credential is allowed to perform the specified operation.
Also is verified that all of the signatures are valid and that the issuers trace back to trusted roots, the XML matches the credential schema, the issuer of the credential is the authority in the target's urn in the case of a delegated credential, all of the GIDs in the credential are valid and the credential is not expired.

For Delegated credentials the privileges must be a subset of the parent credentials, the privileges must have be able to be delegated, the target GID must be the same between child and parents, the expiry time on the child must be no later than the parent and the signer of the child must be the owner of the parent.

7.5.1.10 SFA Federation

7.5.1.10.1 Federation Principle

As is said in other points of this work, to federate two or more authorities they need to satisfy the following requirements:

- Know the location of the peer AMs.
- Ensure the bidirectional trust between the different partners.
- Understand the different RSpecs from the partners.

SFA provides an easy way to complete these requirements. First of all SFA has different configuration files where the Island Manager can fill the aggregate location just typing the URL and the port provided by the partners. To ensure the bidirectional trust, The IM can only put the GIDs in a trusted root directory. If one of the partners removes the GID of the other partner, the two authorities will no longer be federated. In the case of the understanding of the RSpecs, SFA provides a version manager that handles different versions developed for each authority that can be installed separately and there is no need to develop any translator since the version manager does already this translation.
With the SFA version manager several types of RSpecs can be wrote in only one single file and without worrying about the destiny because the SM will do this job.

7.5.1.10.2 Federation Lifecycle

The use of federated resources is transparent to the user. The user will do the same processes as a normal FIT project. Once a slice is created, and the available resources should be shown to the user, is when the SFA Federation Lifecycle starts.
The authority has connected its Slice Manager to their Aggregate Manager and the federated Aggregate Managers. When the user checks the available resources the slice manager sends a broadcast list resources call to all the aggregate managers passing the user credentials (containing the authority GID, their role and their permissions). The AM checks that the authority is valid comparing their existing GIDs form the trusted roots directory with the one in the credentials, then resolves the permissions and returns an RSpec to the authority SM.
In the SM, when all the RSpecs are returned, they are merged and converted in one single SFAv1 RSpec and sent to the user.

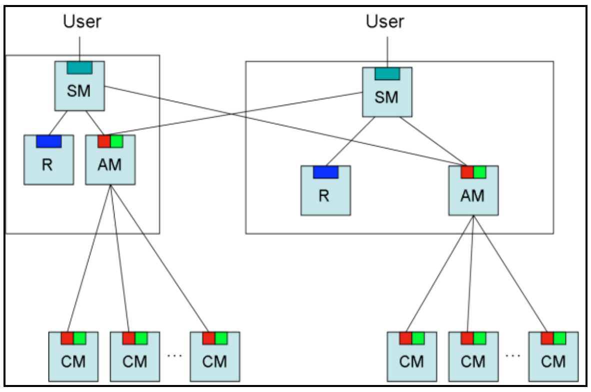


Figure 10. SFA Federation Architecture [27]

When a user allocates his slice using some resources form the available ones, he writes a SFAv1 RSpec that contains the resources that he wants to use and sent to the SM. This time the SM identifies every version of RSpecs and sent each part to the corresponding AM. In the AM, the process of allocation is done and responded with a manifest RSpec to the SM. The SM merges again the RSpec and sends it to the user.

7.5.1.11 SFA Integration in OCF

SFA is a very powerful tool for federation that not only provides security, also provides this features:

- Transparency: A user has not to perform extra work for federation resources and the whole test works in the same way with federation or non-federation.
- Simplicity: For Island Managers federate/un-federate authorities is that simple as to add/remove a single file in a directory.
- Featuring Improvement: SFA provides several different very useful tools: its RSpec version manager, the Registry and the SM/AM API and a very wise way to structure a FIT project.
- Flexibility: Is very easy to install as a plug-in for existing FIT projects or be the main structure for the new ones.

Due to these advantageous features, it was decided to integrate SFA as a plugin in every Ofelia Control Framework Module, mainly in Expedient, VTAM and OFAM. Since there is not a standard on how to deploy exactly FIT projects, SFA and OCF have some important differences on how they handle the different elements, it is required to implement some adaptations to OCF in order to support SFA. The main critical points of this integration are:

- OCF does not have the SFA hierarchy (authority, sub-authority).
- OCF allows to create a project where a user/users can allocate several slices meanwhile SFA does not provide any way to group slices.
- The concept of slice and the slice lifecycles differ from SFA to OCF but have some similarities.
- VTAM and OFAM have their own API, different from the implemented by the SFA.
- SFA have modules as its version manager, RSpec element management and its environment configuration that OCF requires to emulate in order to work properly.

Since OCF has already defined its structure and has its own workflow, the goal of implement SFA is to be able to only offer federation to other testbeds.

7.5.1.11.1 SFA API

SFA uses as API the GENI API version 2 [30] this API is shared between the Aggregate Manager and the Slice Manager and has only the following methods to execute:

- GetVersion
- ListResources
- CreateSliver
- DeleteSliver
- SliverStatus
- ListSlices
- Start/Stop/Reset Slice

7.5.1.11.2 Version Manager

The SFA Version manager is a module that handles the different versions of RSpecs used by the SFA Authorities. This module provides a way to merge the different versions in to one version (SFA RSpec Version), but also defines a set of methods and classes to get RSpec from simple dictionary structures. Version Manager requires three inputs to work properly:

- Loader: The location of all the different RSpec versions deployed.
- Element Library: A Simple library to translate the resources into namespaces. For example, VM objects into "<VM>" namespace.
- Base Class: The main class containing the methods to get RSpecs from dictionaries.

7.5.1.11.3 OCF Integration

For OCF the best way to offer SFA services for federation is to implement three different authorities around the three main modules. In the Virtualization and OpenFlow Aggregate Managers is required to deploy a SFA Aggregate Manager behavior in order to offer resources to foreign testbeds and for Expedient is required to deploy a SFA Slice Manager behavior in order to offer resources from other testbeds to Ofelia Control Framework users.
The VT AM and OF AM modules need to work independently, since for foreign users might not be a good idea to offer OF + virtualization resources in one call and it is not required that our SM and AMs to be connected. The main schema of the SFA implementation is in the following figure:

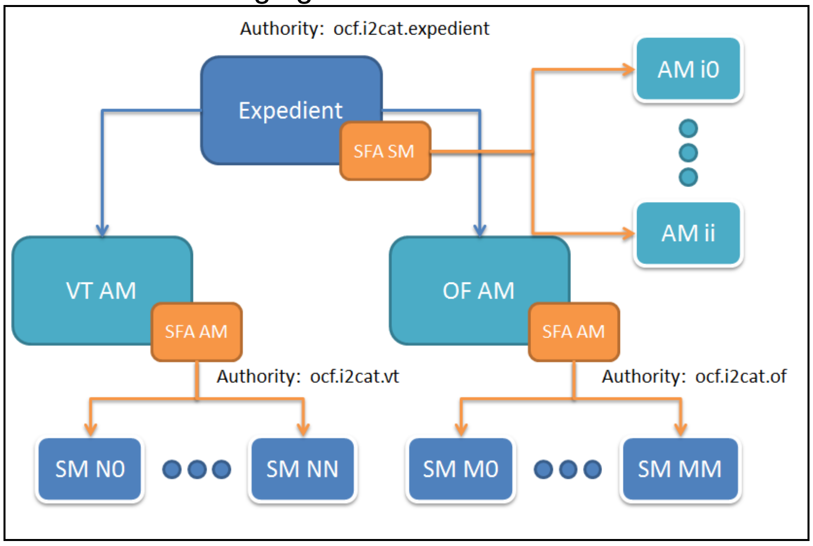


Figure 11. OCF SFA Integration Schema

The three main authorities are independent from each other and in the same level. Every authority is connected to several federated Aggregates or Slice managers but is not required to be connected to the same entities. In the figure, appear three sub-authorities ocf.i2cat.expedient, ocf.i2cat.vt and ocf.i2cat.of having three different GIDs and the certificate chain pointing to ocf.i2cat sub-authority GID.

7.5.1.11.4 Virtualization Aggregate Manager

To enable SFA communication into VTAM it is mandatory to enable the following modules:

- SFA Slice Manager/ Aggregate Manager API
- OCF virtualization RSpec version for SFA
- Redefine Advertisement, Request and Manifest RSpecs (for SFA)
- Define a VTAM driver in order to implement the task above and provide new ones.

7.5.1.11.4.1 SFA Slice Manager/Aggregate Manager API

Ofelia VT AM offers resources based on virtual machines, these VMs do not exist until the user creates them. The SFA plug-in for VTAM must consider this situation. All considered by “slice” in the Aggregate Manager API is considered by Virtual Machine/s in this plug-in. The calls of the API must have the following behavior.

API Call	**SFA VT AM plug-in behavior**
ListResources	Must return a RSpec showing the available Virtualization Servers with their remain resources (HDD, RAM…), their IP and MAC Ranges and their physical connections to the switches if any.
CreateSliver	The user has to send a request RSpec with the parameters of a VM or several VMs. The plug-in must validate the RSpec and if it has no errors, start to create the VM(s) and send to the user a manifest RSpec informing that the VMs are being created getting the “ongoing” state.
SliverStatus	The plug-in must return a dictionary structure showing the state of the VMs belonging to the user that makes the call. The VMs that are being created will be shown as “ongoing”, the others will be shown as “stopped” or “started”.
DeleteSliver	The plug-in must delete the VM(s) that the user wants from the slice.
StartSlice	This call must start all the VMs belonging to the Slice that the user is starting.
StopSlice	This call must stop all the VMs belonging to the Slice that the user is stopping.
ResetSlice	This call must reboot all the VMs belonging to the Slice that the user is rebooting.

Table 2. Work done by VT AM though SFA API

7.5.1.11.4.2 OCF Virtualization SFA RSpec Version

The RSpec version requires new type of element to handle VMs. First of all is required an element describing a Virtualization Server with its own features, then an element describing a VM with the possible characteristics to be set up by the users and finally a generic element range to describe the MAC and the IP ranges.
For the base class, it is only required a class that can traduce slices, slivers and virtualization servers from dictionaries or vice versa.

Since VT AM plug-in for SFA only offer resources from VTAM, and so is not required to have installed the other authority versions of RSpecs.

7.5.1.11.4.3 OCF Virtualization SFA RSpecs schemas

There are three types of RSpecs handled by VTAM plug-in for SFA, the same explained in earlier points of this work: Advertisement, Request and Manifest Rspecs.
The Advertisement RSpecs must contain the main information of the virtualization server, indicating how many resources are still available as the HD and RAM memory, the CPU available, etc. Also it is important to inform what kind of hypervisor is used. Finally this RSpec must show the available IPv4 and MAC ranges and the links between the virtualization servers and the OpenFlow switches.
In the Request RSpec the user has to specify how many VM to create, the features of those VMs (HD, RAM, OS…), how many interfaces the user want in every VM and select the IP (from the IPv4 Ranges) for every interface.
Finally in response of the Request RSpec, the Manifest RSpec must contain enough information to notice the user if something was wrong during the creation of the VM(s). The Manifest must show the name of the VM and the state. The possible states after a request call can only be “ongoing” or “failed”. If the VM creation state is failed, the manifest RSpec shows the exception.

7.5.1.11.4.4 VTAM driver

The VTAM driver has to manage the VMs that are being created using SFA. This means to deploy a driver that handles the normal VM operations (Create, Update, Reboot and Delete) as well as store the SFA created VMs in the database. Since there are two possible interfaces to manage VMs (Expedient calls and SFA calls), the database tables have to store enough information to differentiate this two possible ways to create VMs. Also, it has to be noticed, that the Island Manager has to be able to control this VMs and the VM lifecycle through SFA must not reach the Expedient in any case.
The VTAM driver implementation is made transparently to the normal operation. All the RSpecs and parameters got from the SFA input are translated or adapted in the way that the Agent and the intermediate modules previous to the Agent do not know that they are working for a SFA user.

7.5.1.11.5 OpenFlow Aggregate Manager

The idea to implement SFA to this AM is the same as in VTAM, but since the architecture and the nature of the resources is totally different from VTAM needs to redefine every module applied to VTAM. To enable SFA communication into OFAM is required to enable the following modules:

- SFA Slice Manager/ Aggregate Manager API
- OCF OpenFlow RSpec version for SFA
- Redefine Advertisement, Request and Manifest RSpecs (for SFA)
- Define a OFAM driver in order to implement the task above and provide new ones.

7.5.1.11.5.1 SFA Slice Manager/Aggregate Manager API

The architecture of OF AM is totally different from VT AM, but to accomplish the SFA requirements for federation, the API must be equal. OFAM offers resources OpenFlow based resources, based mainly in OpenFlow Switches and links between this switches joining them into a FlowSpace. The SFA plug-in for OFAM must consider by “sliver” the FlowSpaces creation taking into account the number of switches and the physical links between them. The calls of the API must have the following behavior.

API Call	SFA VT AM plug-in behavior
ListResources	Must return a RSpec showing the available switches with the number of ports. Also in this type of RSpec all the links between the switches must be specified showing the source and destination ports.
CreateSliver	The user has to send a Request RSpec to the AM, this kind of request must create a FlowVisor entry. There is required that a user set up a FlowSpace using a RSpec.
SliverStatus	The plug-in must return a dictionary structure showing the state of the FlowSpaces managed by a user. Since the IM could modify the FlowSpace, this call will return the granted FlowSpaces showing the whole FlowSpace and the pending to approval ones, only with a pending to approval state.
DeleteSliver	The plug-in must delete the FlowSpace(s) that the user wants from the slice.
StartSlice	This call must start all the FlowSpaces belonging to the Slice that the user is starting.
StopSlice	This call must stop all the FlowSpaces belonging to the Slice that the user is stopping.
ResetSlice	This call must reboot all the FlowSpaces belonging to the Slice that the user is rebooting.

Table 3. Work done by OF AM through SFA API.

7.5.1.11.5.2 OCF OpenFlow SFA RSpecs schemas

This aggregate require from the users to create a FlowSpace and this task need to be done using a RSpec. Due to the possible complexity of FlowSpace the RSpecs must be simple and clear as possible. Fortunately OpenFlow is a very used standard and many frameworks are researching with this protocol. In Fact, ProtoGENI [31] has developed a standard for OpenFlow RSpecs. The version used by OF AM is the ProtoGENI OpenFlowV3 [32]. This version provides a way to write the OpenFlow Slivers based on Matches (FlowSpaces). On the other hand, the RSpec version used does not provide a way to write the advertisement and manifest RSpecs. This is not critical due to these two types of RSpecs are only used to inform the users and do not required a standardized form.
The Advertisement RSpecs are based on the OpenFlowV3 RSpec version and must show all the available switches with their numbered ports using the OpenFlowV3 notation and all the links between the switches detailing the source and destination ports.

The Request RSpecs must follow the OpenFlowV3 RSpec version. The information contained in these RSpecs must be the enough information to set a FlowSpace: All these RSpecs must contain the controller URL, the used switches and ports and the MAC ranges, VLAN_IDs, packet types or IP Ranges to be able to the FlowVisor create a slice.
The Manifest RSpecs must contain some user parameters and the state of the requested FlowSpace. If the creation of the FlowSpace was correct, the shown state will be “pending to approval” (waiting to be granted by the Island Manager). If the requested FlowSpace had some error, the shown state will be “failed” and the rose exception messaged will be described.

7.5.1.11.5.3 OCF OpenFlow SFA RSpec Version

This RSpec version does not require new elements, since this version is used as a Standard.
For the base class, it is only required a class that can traduce slices, slivers and OpenFlow switches and links from the OpenFlowV3 RSpec version to dictionary structures. This task is critical due to the flexibility provided by OpenFlowV3 RSpec version and the different ways to set up a FlowSpace.
As the VT AM plug-in SFA, OF AM plug-in will only offer resources from OFAM, and so is not required to have installed the other authority versions of RSpecs.

7.5.1.11.5.4 OFAM Driver

The OFAM driver has to manage basically the FlowSpace request from SFA. Basically, this driver has be able to handle the FlowSpace operations (Create, Update, Reboot and Delete) as in VTAM Driver the databases have to store enough information to be able to differentiate Ofelia Control Framework clients from SFA clients. It has to be ensured that the requested FlowSpaces can be controlled by the island manager (in order to grant or modify them).
The most critical part of the driver is the FlowSpace translation from the RSpecs. The GeniV3 OpenFlow Version Reference [32] is very flexible. There are several ways to describe the FlowSpace and some parameters (as lists, ranges or single value). The FlowSpace Translator must be consistent and well-defined in order to construct properly the FlowSpace wanted by the user and raise exceptions when some errors are found in the client description instead of causing this exception by an inconsistent code.
The OFAM driver implementation is made transparently to the normal operation. All the RSpecs and parameters got from the SFA input are translated or adapted in the way that the FlowVisor and the intermediate modules previous to the FlowVisor do not know that they are working for a SFA client.

7.5.1.11.6 Expedient Integration

Since Expedient is not an aggregate manager the best way to offer resources from other testbeds to Ofelia users is to locate a Slice Manager in this module. The methods to be deployed in Expedient will be the same methods as VTAM or OFAM due to the API between AM and SM is shared. It has to be taken into account that the SFA user management and the OCF user management are different. To be fully compatible with SFA, an adaption from OCF users to SFA users should be implemented.

It is also clear that the deployed Slices Manager has to be transparent to OCF, the clients do not have to do any extra work to request resources from SFA authorities and the received resources must be compatible with Ofelia Slice Structure.
The main tasks to be deployed in expedient are:

- Create a Registry to setup a module to provide GIDs and credentials supported by SFA.
- Make an OCF user Wrapper that converts OCF users into SFA users.
- Install a Version Manager for all the federated authorities.
- Develop an environment to manage the authority federation/un-federation
- Implement the SM API

7.5.1.11.6.1 Registry

The Registry has to be able to get any Registry instance in SFA format. It has to provide a way to get any slice, permission from OCF databases or clearinghouse and adapt it to have a SFA instance. One of the main works of this Registry would be to assign GIDs to the Slices. Of course the OCF Registry must have a way to generate new GIDs, sign these GIDs and create a GID chain (certificate chain) and provide a way to generate signed credentials with Rights and permissions depending of the associated element (Slice, Authority, User, Island Manager, etc.).

7.5.1.11.6.2 User Wrapper

The OCF users are authenticated to OCF via username and password. There is an LDAP that stores the main type of users (Researcher, Island Manager or Owner) and their associate permissions.
SFA requires three main parameters to authenticate or sing in a user: A key pair, a credential and the GID. The required parameters are not meet by OCF registration process, so make authentication in SFA federated AMs with the current OCF parameters is no possible. In order to act with as much transparency as possible with the current OCF users, the best way is to deploy an internal wrapper that converts automatically OCF users to SFA compatible users. The wrapper must meet the following characteristics:

- It must provide a key pair to the user and store it to the LDAP. This is a way to work a transparent as possible with OCF users. It is understood that the privacy that a pair key provides to user is not meet with this approach. Another way to make this handle with the user SFA conversion is to offer the possibility to upload a public key instead of generates it.
- The OCF permissions do not match with SFA rights and also no credential procedure is done. First of all, the user permissions need to be translated SFA Rights and then the must be embedded in a credential as well as the certificate chain from the user to the top OCF authority.
- With the user's public key, a certificate signing request can be generated. Signing this request with the main OCF authority (or sub-authority) the user's final GID can be generated and validated.

7.5.1.11.6.3 Version Manager

Since the Slice Manager will receive RSpecs from different authorities is required and is critical to have available the versions of federated authorities RSpecs versions with their corresponding required elements. Since SFA RSpec versions are organized into the SFA package is a good idea to check for new or updated RSpec versions from the SFA stable repository.

7.5.1.11.6.4 Slice Manager API

This API is the same API used by OFAM and VTAM, but in this case, the required job for every call is to set up a secure server using as certificate the expedient GID and make the corresponding call to all the federated AMs. In case of the calls that as a return have a RSpec, the SM has to merge all the different RSpecs in one using the version manager and then return the merged RSpec to the user.

8. Experimental Results

In this part of the thesis the experimental results validating the different modules deployed are presented; the main aim is to improve the OCF capability to provide testbeds federation and improve the quality of the services of OCF.

8.1 Experimental Environments

Since OCF is a big facility being developed by different partners and teams, the main code is divided in Git branches [33]. Once different modules are completed, the branches are merged improving OCF and obtaining a new version.
Since the work developed by this master thesis was implemented along the project duration and there are modules bigger than others, the experiments are deployed using different OCF versions/branches.

To test the policy engine, Theme Manager, Loop-Detection algorithm and the libvirt monitoring, the test environment is the following:

OCF release version	0.5
Code Branch	ofelia.development
Involved Modules	VT AM, Expedient, OFAM

Table 4. Development Experimental Environment

For the Libvirt monitoring is required to use the agent to run correctly the experiment, in this case the Agent has the following conditions:

OCF release version	0.5
Code Branch	ofelia.stable
Involved Modules	Agent

Table 5. Agent Experimental Environment

Due to the SFA integration is not yet deployed, the experiments are run in a parallel branch with other branch.

OCF release version	0.43
Code Branch	ocf.sfa
Involved Modules	VTAM, OFAM

Table 6. OCF SFA Integration Environment

Finally to test properly SFA federation is required a SFA client. This client does not run any OCF branch, runs a SFAwrap client version.

OCF release version	2.1-24
Code Branch	-
Involved (OCF) Modules	OFAM

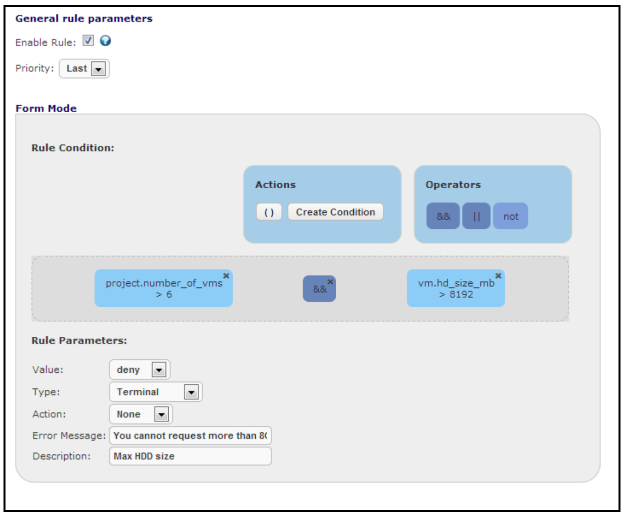
Table 7. SFA Client Environment

8.2 Policy Engine

To show how pyPElib works, first of all, the rule creation procedure is going to be done, showing the different ways to set rule in the OCF VTAM GUI. Then a simple

RuleTable is going to be created with only a Rule that denies every request from those users that try to create a VM with more than 2048 Mbytes of RAM.

8.2.1.1 Rule Creation Using Simple Mode



General rule parameters

Enable Rule: ☑

Priority: Last

Form Mode

Rule Condition:

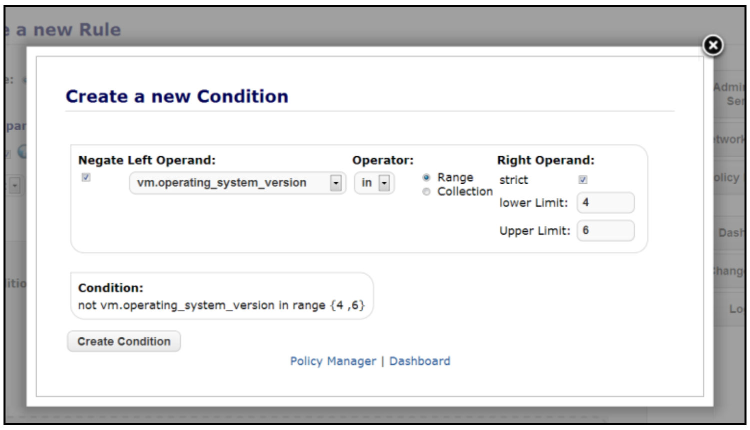
project.number_of_vms > 6 && vm.hd_size_mb > 8192

Rule Parameters:

Value:	deny
Type:	Terminal
Action:	None
Error Message:	You cannot request more than 8(
Description:	Max HDD size

Figure 12. pyPElib GUI in Form Mode

The figure above presents how to create a rule with the form mode. The user has only to drag the conditions to the drop zone and release the mouse and do the same with the operands for complex conditions. Then press the button add rule after setup the rule parameters. As it can be seen in the picture, the rule created contains a complex condition with a deny value without actions being a terminal rule. If a user tries to request more than 6 VM with a hard drive with capacity higher than 8GB, the VM will not be created and the error message will appear to the Expedient message center.



Create a new Condition

Negate Left Operand: ☑ vm.operating_system_version

Operator: in

● Range ○ Collection

Right Operand: strict ☑

lower Limit: 4

Upper Limit: 6

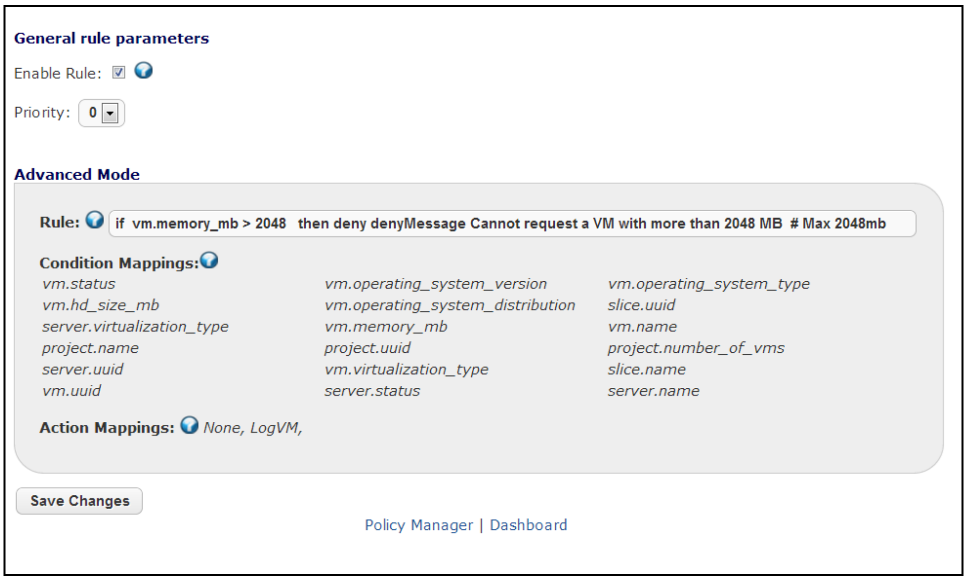
Condition:
not vm.operating_system_version in range {4 ,6}

Figure 13. pyPElib FancyBox create Condition GUI page.

If the user needs to create a condition, he only has to press the create condition button and fill the required fields depending on the condition to be created. In this example the condition created is a negated Range condition limiting the VM operating system version between four and six (these two values included). This condition is going to be discarded in order to simplify the experiment.

Now can it be demonstrated how the Island Manager can setup a rule using the advanced mode:

This time the page is simpler than in the form mode case, the only help that the Island Manager has, consist on the available mappings and some tooltips that informs about the rule structure according to the Regex Parser Driver.


General rule parameters

Enable Rule: ☑

Priority: 0

Advanced Mode

Rule: if vm.memory_mb > 2048 then deny denyMessage Cannot request a VM with more than 2048 MB # Max 2048mb

Condition Mappings:

vm.status	*vm.operating_system_version*	*vm.operating_system_type*
vm.hd_size_mb	*vm.operating_system_distribution*	*slice.uuid*
server.virtualization_type	*vm.memory_mb*	*vm.name*
project.name	*project.uuid*	*project.number_of_vms*
server.uuid	*vm.virtualization_type*	*slice.name*
vm.uuid	*server.status*	*server.name*

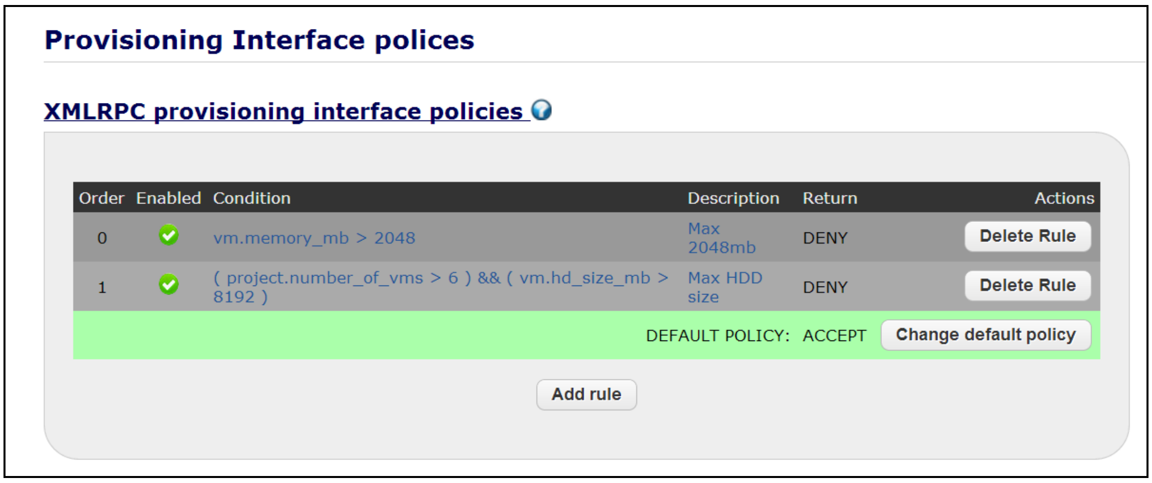
Action Mappings: *None, LogVM,*

Figure 14. pyPElib Advanced Mode Rule GUI

This time the condition is also make implicitly with the rule sentence. As it can be seen the rule this time is a terminal rule that denies every VM user request with more than 2GB of RAM memory.

8.2.1.2 RuleTable View

After creating these two rules the RuleTable has this appearance:


Provisioning Interface polices

XMLRPC provisioning interface policies

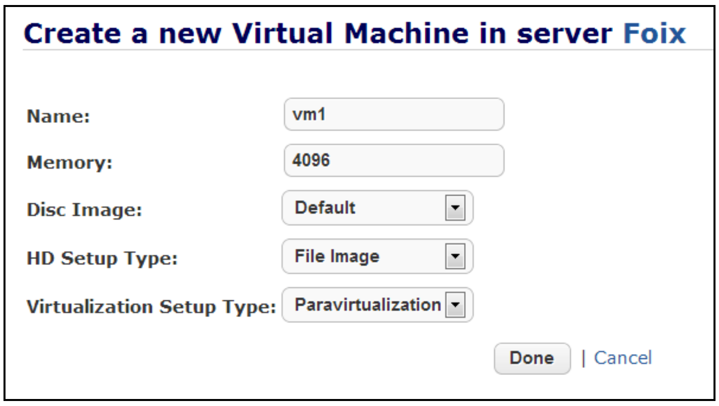
Order	Enabled	Condition	Description	Return	Actions
0	✓	vm.memory_mb > 2048	Max 2048mb	DENY	Delete Rule
1	✓	(project.number_of_vms > 6) && (vm.hd_size_mb > 8192)	Max HDD size	DENY	Delete Rule
			DEFAULT POLICY:	ACCEPT	Change default policy

Figure 15. pyPElib RuleTable Management

As it can be seen the two rules created appear in order. At this point, the rules can be deleted, modified or disabled. The default policy of the RuleTable is Accept, so if no rules are matched, the user request is going to be accepted.

8.2.1.3 Testing the Created Rules from Expedient

To test the PolicyEngine, a VM with the following parameters is going to be created:

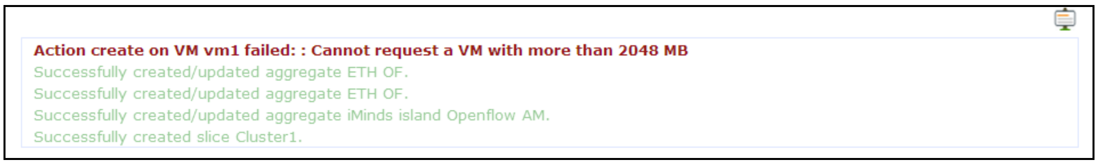


Create a new Virtual Machine in server Foix

Name:	vm1
Memory:	4096
Disc Image:	Default
HD Setup Type:	File Image
Virtualization Setup Type:	Paravirtualization

Figure 16. Trying to Request a VM with more than RAM 2GB

The requested VM has more than 2GB of RAM, so presumably this request is not going to be accepted by the PolicyEngine and the Raised Exception by the rule error message is going to appear in the Expedient Message Center.



```
Action create on VM vm1 failed: : Cannot request a VM with more than 2048 MB
Successfully created/updated aggregate ETH OF.
Successfully created/updated aggregate ETH OF.
Successfully created/updated aggregate iMinds island Openflow AM.
Successfully created slice Cluster1.
```

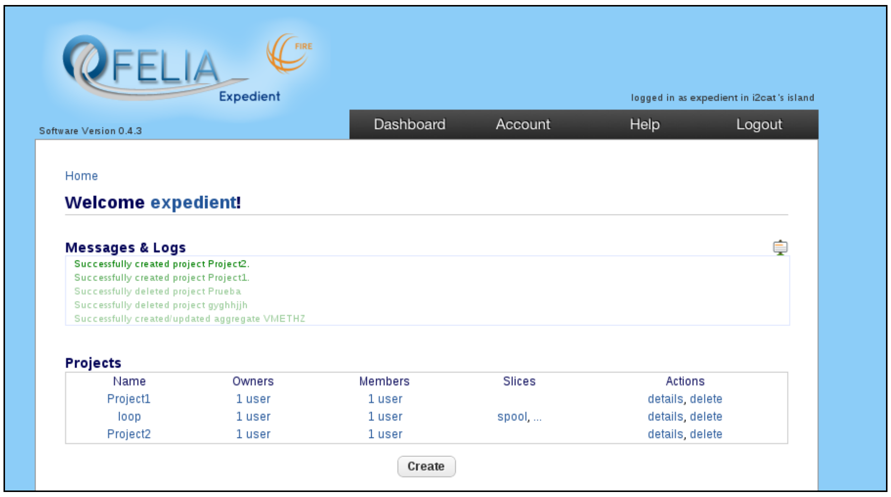
Figure 17. pyPElib Exception Raised in expedient when requesting more than 2GB RAM

The exception appeared in the message center is warning to the user what is doing wrong. The main pyPElib purpose as PolicyEngine is accomplished.

8.3 Theme Manager

For this test is going to be presented a theoretically theme for a UPC OCF based testbed. First of all is going to be shown the main appearance of the Expedient web page, and then the main OCF logo will be substituted by the UPC logo, the main HTML page will contain a different welcome title and finally the background color is going to be changed in the CSS, with this approach the whole appearance of the framework is going to change by only modifying 3 files and some settings.

8.3.1.1 Expedient Appearance



logged in as expedient in i2cat's island

Welcome expedient!

Messages & Logs

Successfully created project Project2.
Successfully created project Project1.
Successfully deleted project Prueba
Successfully deleted project gyghhjjh
Successfully created/updated aggregate VMETHZ

Projects

Name	Owners	Members	Slices	Actions
Project1	1 user	1 user		details, delete
loop	1 user	1 user	spool, ...	details, delete
Project2	1 user	1 user		details, delete

Figure 18. OCF Expedient Appearance

In this figure appears the main page of Expedient Module, the Ofelia logo can be seen, the main welcome phrase is "Welcome expedient!" and the background color is blue.

8.3.1.2 Setting up the UPC Appearance

First of all, the UPC directory must be created in the static content and templates directories.

```
OfeliaSDKr1:/opt/ofelia/expedient/src/templates# cd
default/  fibre/   upc/
```

Figure 19. Theme Manager Template Folders Creation

```
OfeliaSDKr1:/opt/ofelia/expedient/src/static/expedient/clearinghouse/media# cd
default/          fibre/            upc/
```

Figure 20. Theme Manager Static Content Folders Creation

The logo image maintaining its name but containing the UPC logo must be placed in the static content directory. As it can be seen, is only required to add the files that are different, the other files, if remain equal can be stored in default directory.

```
OfeliaSDKr1:/opt/ofelia/expedient/src/static/expedient/clearinghouse/media/upc/img# cd ../../default/img/
account.png            favicon.ico            notification_16x16.png   switch-5406zl.png        users-tiny.png
active.png             genilogo.png           ofelia.png               switch-cat6500.png       warning.png
aggregate-tiny.png     groupin.png            question_mark_15x15.png  switch-juniper.png       white_arrow_big.png
black_arrow_big.png    help.png               question_mark.png        switch-nec-ip8800.png    white_arrow.png
black_arrow.png        host-tiny.png          server-tiny.png          switch-netfpga.png       white_big.png
black_big.png          inactive.png           slice.png                switch-nf2.png           white.png
black.png              loading.gif            slice-tiny.png           switch.png               zoomin.png
button.pxm             logo_expedient.png     status-question.jpg      switch-tiny.png          zoomout.png
dashboard.png          logout.png             status-question-small.jpg tiny_arrow.png          zoom.png
OfeliaSDKr1:/opt/ofelia/expedient/src/static/expedient/clearinghouse/media/upc/img# ls
logo_expedient.png
```

Figure 21. Comparison between Default Static Content and UPC Static content

The main template with the new welcome phrase must be placed in the correct template directory. As in this test is only modifying one single template, there is no need to copy the other templates in the UPC directory.

```
OfeliaSDKr1:/opt/ofelia/expedient/src/templates/upc/expedient/clearinghouse# cd ../../../default/expedient/clearinghouse/
404.html                 base.html                help/                    messagecenter/           registration/            under_construction.html
500.html                 div_base.html            iframebase.html          permissionmgmt/          roles/                   users/
aggregate/               fixformtable.js          index.html               project/                 slice/
OfeliaSDKr1:/opt/ofelia/expedient/src/templates/upc/expedient/clearinghouse# ls
index.html
```

Figure 22. Comparison between Default Templates and UPC Templates

The new CSS file must be placed in CSSs static content directory. Again is just required to change the file that is going to be different.

```
OfeliaSDKr1:/opt/ofelia/expedient/src/static/expedient/clearinghouse/media/upc/css# cd ../../default/css/
egeni.css            images/              ocf.css              themes/              tooltip.topology.css
egeni-iframe.css     jquery-ui.css        ocf-iframe.css       tooltip.css
OfeliaSDKr1:/opt/ofelia/expedient/src/static/expedient/clearinghouse/media/upc/css# ls
ocf.css
```

Figure 23. Comparison Between Default CSS content and UPC CSS content

And finally, the main theme must be set as “upc” in the expedient settings.

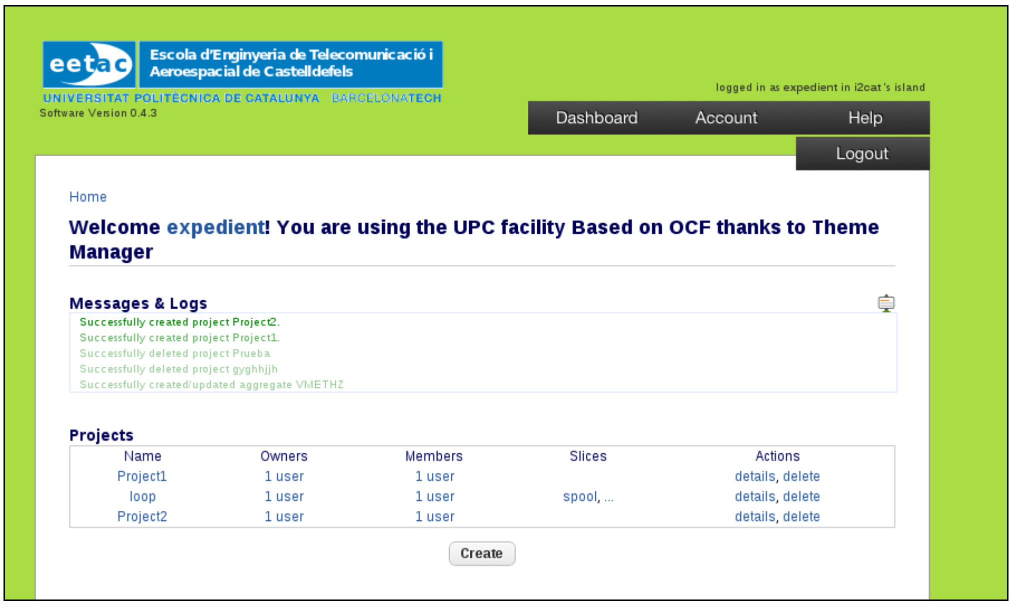
```
#In days
SLICE_DEFAULT_EXPIRATION_TIME = 30

ENABLE_LDAP_BACKEND = True

THEME = "upc"
```

Figure 24. Setting the Main Theme as UPC

The UPC FIT testbed based on OCF is ready to share resources.


eetac
Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH
Dashboard
Account
Help
Logout
Create

Software Version 0.4.3

logged in as expedient in i2cat's island

Home

Welcome expedient! You are using the UPC facility Based on OCF thanks to Theme Manager

Messages & Logs

Successfully created project Project2.
Successfully created project Project1.
Successfully deleted project Prueba
Successfully deleted project gyghhjjh
Successfully created/updated aggregate VMETHZ

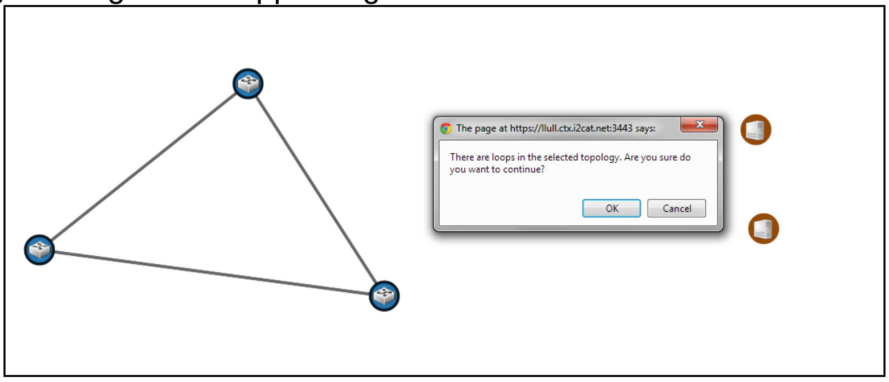
Projects

Name	Owners	Members	Slices	Actions
Project1	1 user	1 user		details, delete
loop	1 user	1 user	spool, ...	details, delete
Project2	1 user	1 user		details, delete

Figure 25. UPC Theme main Appearance

8.4 *Loop Detection Algorithm*

To test the loop detection algorithm basically is going to done the test by requesting a simple containing loops topology to check the warning message and then is going to request over the same conditions but a topology without loops to check that this time the warning message is not appearing.



The page at https://llull.ctx.i2cat.net:3443 says:

There are loops in the selected topology. Are you sure do you want to continue?

Figure 26. Topology with loops request.

The topology tested is a star topology containing three switches. When a user wants to continue over this form and select the FlowSpace, he gets a warning message, informing that the selected topology contains loops. The user now, can change topology or ignore this message and request a FlowSpace.
In the case that the user do not selects a loop containing topology, it not appears any warning message at the selection of the FlowSpace.

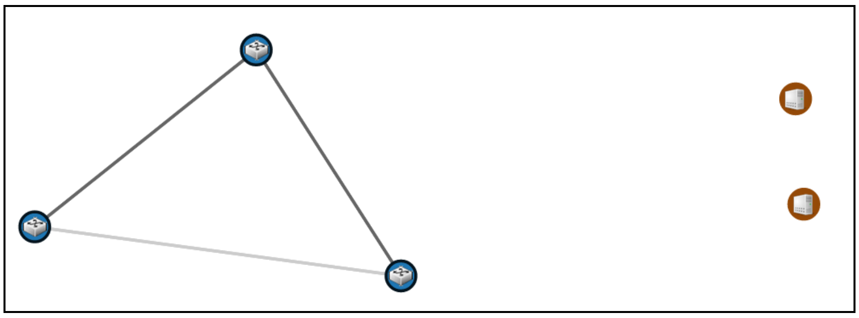


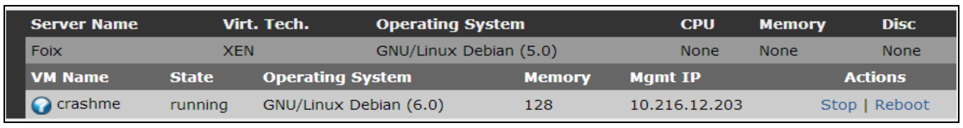
Figure 27. Topology without loops detection.

8.5 *Libvirt Monitoring*

Make a VM crash is a difficult task, to make the simulation of a VM crash, the test will consist in the creation of a VM, the start of that VM and directly in the Agent this VM is going to be stopped using the XEN command line for VM management.

8.5.1.1 Creation of a VM

The name of the created VM is “crashme”:

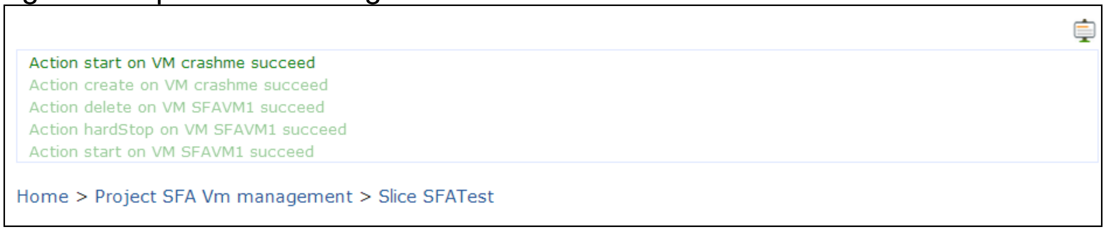


Server Name	Virt. Tech.	Operating System	CPU	Memory	Disc
Foix	XEN	GNU/Linux Debian (5.0)	None	None	None

VM Name	State	Operating System	Memory	Mgmt IP	Actions
crashme	running	GNU/Linux Debian (6.0)	128	10.216.12.203	Stop | Reboot

Figure 28. "Crashme" VM created and Started

After the creation and the start of the VM, the users can see the actions applied through the Expedient Message Center.



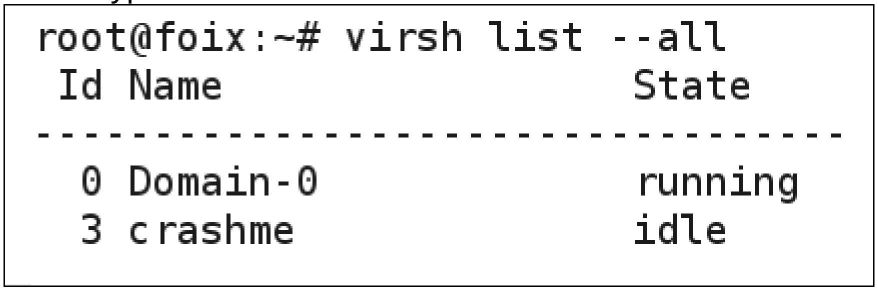
Action start on VM crashme succeed
Action create on VM crashme succeed
Action delete on VM SFAVM1 succeed
Action hardStop on VM SFAVM1 succeed
Action start on VM SFAVM1 succeed

Home > Project SFA Vm management > Slice SFATest

Figure 29. Message Center last five user actions

8.5.1.2 Agent VM Stop

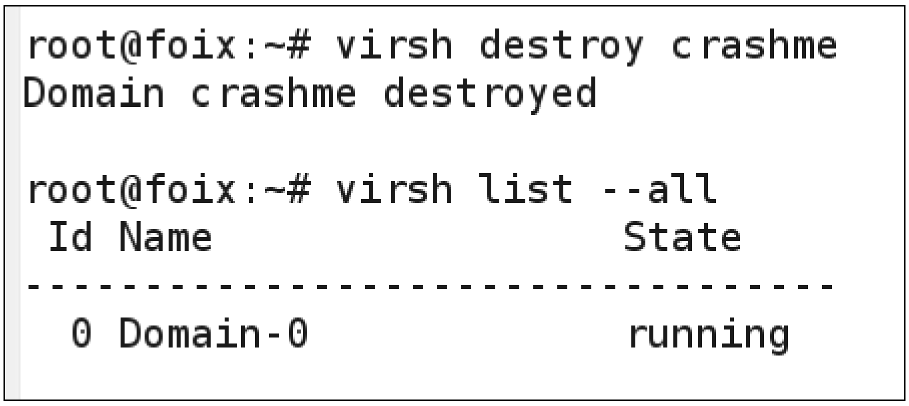
As it can be seen in the following figure, the running VMs in the Agent are "crashme" and the domain-0 hypervisor:



```
root@foix:~# virsh list --all
 Id Name                 State
----------------------------------
  0 Domain-0              running
  3 crashme              idle
```

Figure 30. Agent Running VMs

Using the following command line, a VM can be stopped by name, in this case "crashme".

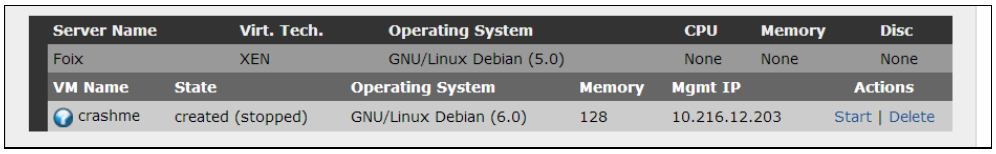


```
root@foix:~# virsh destroy crashme
Domain crashme destroyed

root@foix:~# virsh list --all
 Id Name                 State
----------------------------------
  0 Domain-0              running
```

Figure 31. "crashme" VM crash simulation

Once the VM is stopped in the Agent (crash simulation), the user in the Expedient views its VM stopped without command it.

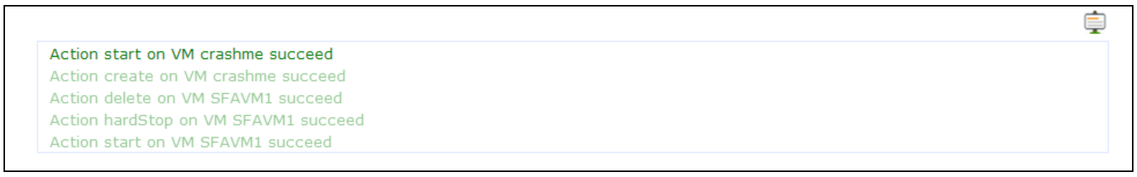


Server Name	Virt. Tech.	Operating System	CPU	Memory	Disc
Foix	XEN	GNU/Linux Debian (5.0)	None	None	None

VM Name	State	Operating System	Memory	Mgmt IP	Actions
crashme	created (stopped)	GNU/Linux Debian (6.0)	128	10.216.12.203	Start | Delete

Figure 32. "crashme" VM State After Crash Simulation

If the message center is checked, is seeing that not from the user is taken, so the stop procedure is being made automatically.



```
Action start on VM crashme succeed
Action create on VM crashme succeed
Action delete on VM SFAVM1 succeed
Action hardStop on VM SFAVM1 succeed
Action start on VM SFAVM1 succeed
```

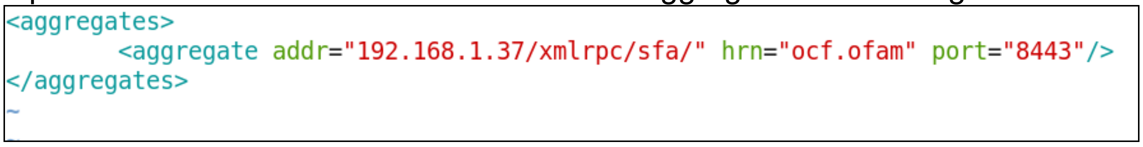
Figure 33. Message Center State after VM Crash Simulation

8.6 SFA integration

To test the OCF SFA implementation, OFAM is going to be accessed via SFA using the command line interface provided by SFA. First of all, the configuration before the federation is going to be shown. Then, the List Resources and the Create Sliver calls are going to be executed in order to view the available OpenFlow resources of OFAM and the FlowSpace creation.

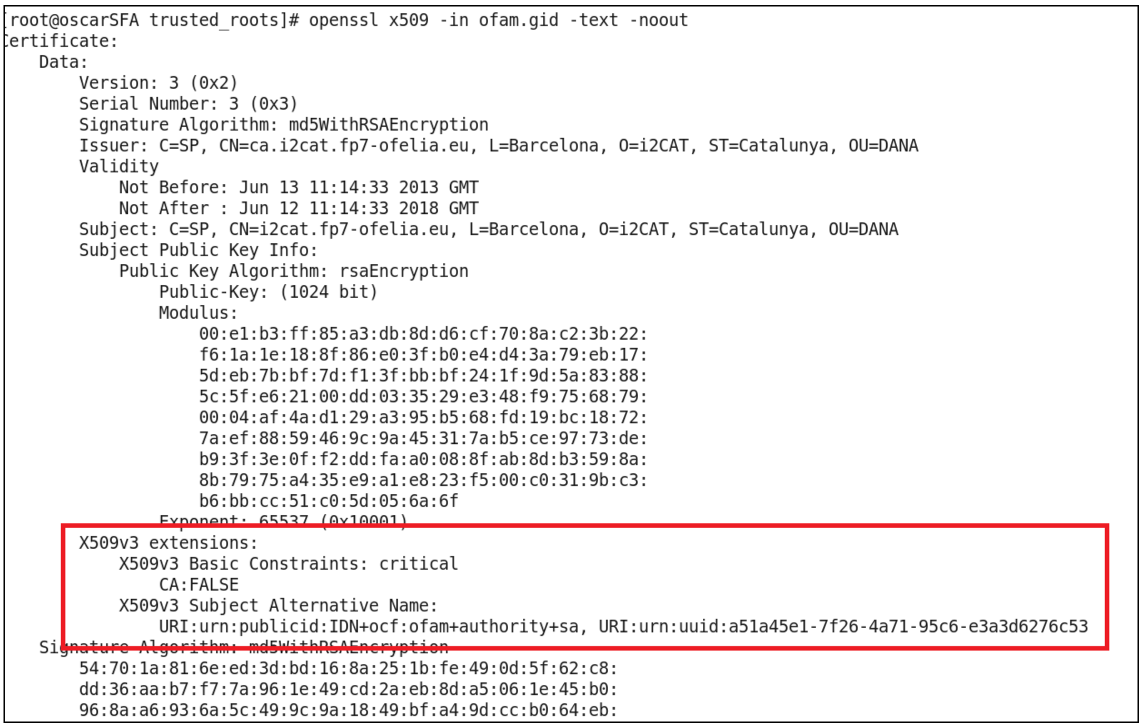
8.6.1.1Configuring SFA Side

To be able to federate the OFAM from Ofelia, the OFAM GID and the Aggregate URL and port and the HRN must be entered in the “aggregates.xml” configuration file.



```
<aggregates>
        <aggregate addr="192.168.1.37/xmlrpc/sfa/" hrn="ocf.ofam" port="8443"/>
</aggregates>
~
```

Figure 34. Setting the Federate AM URL, HRN and PORT



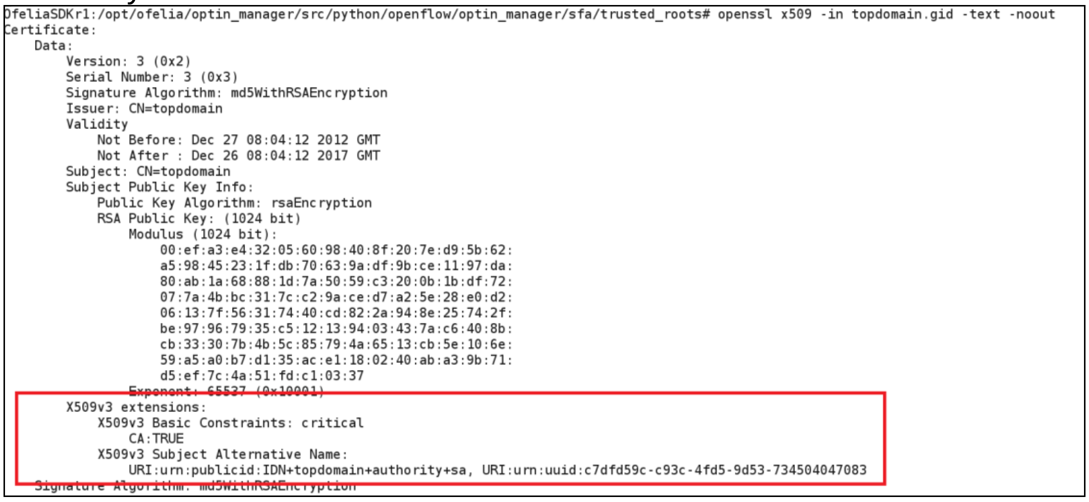
```
[root@oscarSFA trusted_roots]# openssl x509 -in ofam.gid -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3 (0x3)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=SP, CN=ca.i2cat.fp7-ofelia.eu, L=Barcelona, O=i2CAT, ST=Catalunya, OU=DANA
        Validity
            Not Before: Jun 13 11:14:33 2013 GMT
            Not After : Jun 12 11:14:33 2018 GMT
        Subject: C=SP, CN=i2cat.fp7-ofelia.eu, L=Barcelona, O=i2CAT, ST=Catalunya, OU=DANA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:e1:b3:ff:85:a3:db:8d:d6:cf:70:8a:c2:3b:22:
                    f6:1a:1e:18:8f:86:e0:3f:b0:e4:d4:3a:79:eb:17:
                    5d:eb:7b:bf:7d:f1:3f:bb:bf:24:1f:9d:5a:83:88:
                    5c:5f:e6:21:00:dd:03:35:29:e3:48:f9:75:68:79:
                    00:04:af:4a:d1:29:a3:95:b5:68:fd:19:bc:18:72:
                    7a:ef:88:59:46:9c:9a:45:31:7a:b5:ce:97:73:de:
                    b9:3f:3e:0f:f2:dd:fa:a0:08:8f:ab:8d:b3:59:8a:
                    8b:79:75:a4:35:e9:a1:e8:23:f5:00:c0:31:9b:c3:
                    b6:bb:cc:51:c0:5d:05:6a:6f
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Subject Alternative Name:
                URI:urn:publicid:IDN+ocf:ofam+authority+sa, URI:urn:uuid:a51a45e1-7f26-4a71-95c6-e3a3d6276c53
    Signature Algorithm: md5WithRSAEncryption
        54:70:1a:81:6e:ed:3d:bd:16:8a:25:1b:fe:49:0d:5f:62:c8:
        dd:36:aa:b7:f7:7a:96:1e:49:cd:2a:eb:8d:a5:06:1e:45:b0:
        96:8a:a6:93:6a:5c:49:9c:9a:18:49:bf:a4:9d:cc:b0:64:eb:
```

Figure 35. OFAM GID certificate characteristics

As it can be seen in the figure above, the OFAM GID is placed in the SFA trusted roots directory. The most important parts of this certificate are the URN of OFAM and the Basic Constrains CA Flag used by SFA that distinguish a GID from a common X509 certificate.

8.6.1.2 Configuring OCF Side

The HRN of the SFA client is *topdomain*. The next step to configure SFA is make OCF trust the *topdomain* certificate. Since the OCF SFA approach to federate OFAM is unidirectional (SFA to OFAM) there is no need to OFAM knows the *topdomain* URL. So to federate topdomain it is only required to put its GID in OCF SFA trusted roots directory.



```
OfeliaSDKr1:/opt/ofelia/optin_manager/src/python/openflow/optin_manager/sfa/trusted_roots# openssl x509 -in topdomain.gid -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3 (0x3)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: CN=topdomain
        Validity
            Not Before: Dec 27 08:04:12 2012 GMT
            Not After : Dec 26 08:04:12 2017 GMT
        Subject: CN=topdomain
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:ef:a3:e4:32:05:60:98:40:8f:20:7e:d9:5b:62:
                    a5:98:45:23:1f:db:70:63:9a:df:9b:ce:11:97:da:
                    80:ab:1a:68:88:1d:7a:50:59:c3:20:0b:1b:df:72:
                    07:7a:4b:bc:31:7c:c2:9a:ce:d7:a2:5e:28:e0:d2:
                    06:13:7f:56:31:74:40:cd:82:2a:94:8e:25:74:2f:
                    be:97:96:79:35:c5:12:13:94:03:43:7a:c6:40:8b:
                    cb:33:30:7b:4b:5c:85:79:4a:65:13:cb:5e:10:6e:
                    59:a5:a0:b7:d1:35:ac:e1:18:02:40:ab:a3:9b:71:
                    d5:ef:7c:4a:51:fd:c1:03:37
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Subject Alternative Name:
                URI:urn:publicid:IDN+topdomain+authority+sa, URI:urn:uuid:c7dfd59c-c93c-4fd5-9d53-734504047083
    Signature Algorithm: md5WithRSAEncryption
```

Figure 36. Topdomain GID certificate main characteristics

Again, can be seen the X509 extensions composing a GID certificate.

8.6.1.3 List Resources Call

To test the List Resources call it is only required to run the command “sfi resources”: Annex 1 shows the response from the ListResources. It can be differentiated the topdomain resources and the OpenFlow resources get from OFAM response.

8.6.1.4 Create Sliver Call

To test the create sliver call, this time the input needs a OpenFlow RSpec and a created Slice. The RSpec used to test this call is located in the Annex 2 and the name of the Slice used is Slice1. To create a sliver once a slice and a RSpec are created is “sfi create <slice HRN> <RSpec file>”. The output obtained by the creation of the sliver is shown in figure 37.

As it can be seen, OFAM responded with a pending to approval for the requested RSpec. The user has to wait until the requested RSpec is approved. The pending state can be checked by sending “SliverStatus” calls.

```xml
<?xml version="1.0"?>
<rspec xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" type="manifest"
xsi:schemaLocation="http://www.protogeni.net/resources/rspec/2 https://github.com/fp7-
ofelia/ocf/blob/ocf.rspecs/openflow/schemas/manifest.xsd
http://www.protogeni.net/resources/rspec/2/ad.xsd https://github.com/fp7-
ofelia/ocf/blob/ocf.rspecs/openflow/schemas/network_schema.xsd" expires="2013-06-
19T17:27:44Z" generated="2013-06-19T16:27:44Z">
 <statistics call="CreateSliver">
  <aggregate status="success" name="topdomain" elapsed="0.228933095932"/>
  <aggregate status="success" name="ocf.ofam" elapsed="0.774904966354"/>
 </statistics>
 <network name="ocf_of">
  <sliver slice="slice1" description="OF request example" email="a@b.com" status="Pending
to approval"/>
 </network>
</rspec>
```

Figure 37. OFAM Manifest RSpec Response

9. Conclusions

OpenFlow as well as other SDN-based paradigms can help to improve the future internet thanks to their flexibility. Moreover, they can help in the provisioning of experimental facilities to allow researchers to test their novel protocols at large scale with real traffic.

Ofelia Control Framework gives an easy way to experiment with OpenFlow providing the two required resources, the VM where the controller can be configured and the OpenFlow switches. All the experiments are supervised by the FlowVisor, allowing to isolate each experiment, but adding some extra delay due to the work carried in this isolation process.

The main problem of the FlowVisor is that is only compatible with OpenFlow 1.0. This limitation inhibits the possibility to upgrade the version of OF used by OCF. To solve this problem a new OpenFlow network “slicer” compatible with OpenFlow upgrades is required. And the FlowVisor interaction must be changed in order to fit with this new controller. This controller should be compatible with any OF versions to avoid adapting the slice interface with the new controller every time that the OpenFlow version is upgraded.

Ofelia Control Framework has also very powerful tools to help the users and the Island Manager to control the Island. The policy Engine Tool helps to the Island Manager set simple rules to administrate the resources as well as get information that he needs (by programming its own mapping). The VM callbacks generated by libvirt informs the users if a VM has crashed due to some kind of errors in the server or caused by an user script, without the need to use an issue tracker to find out what happened with the crashed VM or why the user cannot access to that VM.

Ofelia main federation, by adding Expedient plugins is a good tool for OCF based testbeds but is not enough since it requires that the federated aggregate manager to be implemented using python and the Django web framework. It is not a good solution as a main federation tool since is not reasonable to require a python binding and the Django framework to be running for every testbed that wants to federate with Ofelia.

SFA has the keys of federation with any kind of testbed, since this library do not require to use an specific language or run any framework, this tool only needs a driver able to communicate with their AM/SM API. The SFA Module implemented in this master thesis, works achieving the main SFA objectives: transparency, simplicity and flexibility.

The Policy Engine module, as is demonstrated in the experimental part, is a tool that simplifies and can carry out a lot of the Island Manager’s work. Moreover, this library is independent from OCF and it can be applied in different software modules as other FIT projects, firewalls (as IP tables), automatic access control solutions, etc.

The Libvirt Monitoring module cannot be exported, but the contribution to OCF grants confidence to the users to trust OCF services. Knowing that the requested VMs are being checked and if something fails, the user will be aware of it.

The Loop Detection Algorithm is a module that adds value to OCF against other OF Testbeds. Despite seems a trivial task, these kinds of algorithms need lot of time to plan the discover strategies and to perform the bugs. As presented in the

Experimental Part section, the selection of a containing loops topology can make fail an experiment which does not expect loops.
The Theme Manager is a key tool for OCF-based testbeds developers. Thanks to its flexibility and design Theme manager allow to change the content that is going to be different from OCF default theme. Saving hours for the developers and avoiding to replicate code and files.
Every module deployed in this thesis has been experimentally validated; they work as expected, erasing some of the OCF limitations and expanding OCF features in three ways, the users, the Island Manager and the Developers:

- The users have a more reliable and consistent service thanks to the Libvirt Monitoring and the Loop Detection algorithm.
- The Island Manager work has been reduced thanks to the Policy Engine.
- The OCF-based Testbeds developers have more facilities to deploy the GUI thanks to the Theme Manager.

10. References

[1] M. Handley, “Why the Internet only just works”, BT Technology Journal, vol. 24, no. 3, July 2006
[2] Future Internet Research and Experimentation (FIRE) Cordis Europe Union Department. Web Site: http://cordis.europa.eu/fp7/ict/fire/home_en.html7
[3] Future Internet testbed experimentation between Brazil and Europe (FIBRE) Web Site: http://www.fibre-ict.eu/
[4] Y. Demchenko, J. van der Ham, R. Strijkers, M. Ghijsen, C. Ngo, M. Cristea “Generic Architecture for Cloud Infrastructure as a Service (IaaS) Provisioning Model” April 2011
[5] Open Networking Foundation white paper, “Software-Defined Networking: The New Norm for Networks” April 2012
[6] IBM Global Education White Paper, “Virtualization in education” October 2009
[7] Hardware Virtualization: Brief Description of virtualization types. Web Site: http://en.wikipedia.org/wiki/Partial_virtualization#Partial_virtualization
[8] N. M. Chowdhury and R. Boutaba, “Network Virtualization: State of Art and Research Challenges”, IEEE Communications Magazine, Vol. 47, Issue: 7, pp. 20-26, July 2009
[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turn “OpenFlow: Enabling Innovation in Campus Networks”, March 2008
[10] OpenFlow Switch Specification v1.0. Web site: http://www.OpenFlowswitch.org
[11] R. Sherwood, “FlowVisor: A Network Virtualization Layer”, October 2009
[12] Apache Software Fundation. Web site: http://www.apache.org/
[13] Django project foundation. Web site: https://www.djangoproject.com/
[14] Expedient: A Pluggable Platform for GENI Control Frameworks. Web site: http://yuba.stanford.edu/~jnaous/expedient/
[15] Xen HyperVisor Project. Web Sitehttp://www.xenproject.org/
[16] Libvirt, the virtualization API. Web Site: http://libvirt.org/
[17] pyPElib wiki. Web site: https://code.google.com/p/pypelib/wiki/Overview?tm=6
[18] Fancybox, fancy jQery lightbox alternative. Web Site: http://fancybox.net/
[19] Redips: Drag and Drop JavaScript Library. Web site: http://www.redips.net/
[20] Geant Project. Web Site: http://www.geant.net
[21] Depth First Search and Breath First Search Overview: Web Site: http://homepages.ius.edu/rwisman/C455/html/notes/Chapter22/DFS.htm
[22] K. Nagatou, Y. Ishii, “Validated computation tool for the Perron-Frobenius eigenvalues”. http://www.math.kyushuu.ac.jp/files/publication/file/08e5e855bd5e78da9fd3e253d3f65b60.pdf
[23] MySlice SFA based project. Web Site: http://myslice.info/
[24] Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures (FEDERICA) Project. Web Site: http://www.fp7-federica.eu/
[25] Networking innovations Over Virtualized Infrastructures (NOVI). Web Site: www.fp7-novi.eu/
[26] Network Implementation Testbed using Open Source code (NITOS). Web Site: https://www.onelab.eu/index.php/testbeds/onelab-testbeds/nitos.html

[27] L. Peterson, S. Sevinc, S. Baker, T. Mack, R. Moran, F. Ahmed: “PlanetLab Implementation of the Slice-Based Facility Architecture”. Web site: http://www.cs.princeton.edu/~llp/geniwrapper.pdf, June 2009
[28] X509 Certificates RFC. Web Site: http://www.ietf.org/rfc/rfc3280.txt
[29] GENI API v2 Reference. Web site: http://groups.geni.net/geni/wiki/GAPI_AM_API_V2
[30] ProtoGENI RSpec documentation. Web Site: http://www.protogeni.net/
[31] ProtoGENI OpenFlow v3 RSpec Reference. Web Site: http://groups.geni.net/geni/wiki/HowTo/WriteOFv3Rspecs
[32] Open SFA Documentation. Web Site: http://opensfa.org
[33] Git code versioner project. Web site: http://git-scm.com/

11. Annexes

11.1 Annex 1

11.1.1 List Resources call response:

```
<?xml version="1.0"?>
<RSpec type="SFA" expires="2013-06-19T17:15:56Z" generated="2013-06-19T16:15:56Z">
  <statistics call="ListResources">
    <aggregate status="success" name="topdomain" elapsed="0.143084049225"/>
    <aggregate status="success" name="ocf.ofam" elapsed="0.813298940659"/>
  </statistics>
  <network name="topdomain">
    <node component_manager_id="urn:publicid:IDN+topdomain+authority+cm"
component_id="urn:publicid:IDN+topdomain:dummy+node+node1.dummy-testbed.org"
component_name="node1.dummy-testbed.org"
site_id="urn:publicid:IDN+topdomain:dummy+authority+sa">
      <hostname>node1.dummy-testbed.org</hostname>
      <location country="unknown" longitude="123456" latitude="654321"/>
      <exclusive>FALSE</exclusive>
    </node>
    <node component_manager_id="urn:publicid:IDN+topdomain+authority+cm"
component_id="urn:publicid:IDN+topdomain:dummy+node+node2.dummy-testbed.org"
component_name="node2.dummy-testbed.org"
site_id="urn:publicid:IDN+topdomain:dummy+authority+sa">
      <hostname>node2.dummy-testbed.org</hostname>
      <location country="unknown" longitude="123456" latitude="654321"/>
      <exclusive>FALSE</exclusive>
    </node>
    <node component_manager_id="urn:publicid:IDN+topdomain+authority+cm"
component_id="urn:publicid:IDN+topdomain:dummy+node+node3.dummy-testbed.org"
component_name="node3.dummy-testbed.org"
site_id="urn:publicid:IDN+topdomain:dummy+authority+sa">
      <hostname>node3.dummy-testbed.org</hostname>
      <location country="unknown" longitude="123456" latitude="654321"/>
      <exclusive>FALSE</exclusive>
    </node>
    <node component_manager_id="urn:publicid:IDN+topdomain+authority+cm"
component_id="urn:publicid:IDN+topdomain:dummy+node+node4.dummy-testbed.org"
component_name="node4.dummy-testbed.org"
site_id="urn:publicid:IDN+topdomain:dummy+authority+sa">
      <hostname>node4.dummy-testbed.org</hostname>
      <location country="unknown" longitude="123456" latitude="654321"/>
      <exclusive>FALSE</exclusive>
    </node>
    <node component_manager_id="urn:publicid:IDN+topdomain+authority+cm"
component_id="urn:publicid:IDN+topdomain:dummy+node+node5.dummy-testbed.org"
component_name="node5.dummy-testbed.org"
site_id="urn:publicid:IDN+topdomain:dummy+authority+sa">
      <hostname>node5.dummy-testbed.org</hostname>
      <location country="unknown" longitude="123456" latitude="654321"/>
      <exclusive>FALSE</exclusive>
    </node>
    <node component_manager_id="urn:publicid:IDN+topdomain+authority+cm"
component_id="urn:publicid:IDN+topdomain:dummy+node+node6.dummy-testbed.org"
```

```
component_name="node6.dummy-testbed.org"
site_id="urn:publicid:IDN+topdomain:dummy+authority+sa">
      <hostname>node6.dummy-testbed.org</hostname>
      <location country="unknown" longitude="123456" latitude="654321"/>
      <exclusive>FALSE</exclusive>
    </node>
    <node component_manager_id="urn:publicid:IDN+topdomain+authority+cm"
component_id="urn:publicid:IDN+topdomain:dummy+node+node7.dummy-testbed.org"
component_name="node7.dummy-testbed.org"
site_id="urn:publicid:IDN+topdomain:dummy+authority+sa">
      <hostname>node7.dummy-testbed.org</hostname>
      <location country="unknown" longitude="123456" latitude="654321"/>
      <exclusive>FALSE</exclusive>
    </node>
    <node component_manager_id="urn:publicid:IDN+topdomain+authority+cm"
component_id="urn:publicid:IDN+topdomain:dummy+node+node8.dummy-testbed.org"
component_name="node8.dummy-testbed.org"
site_id="urn:publicid:IDN+topdomain:dummy+authority+sa">
      <hostname>node8.dummy-testbed.org</hostname>
      <location country="unknown" longitude="123456" latitude="654321"/>
      <exclusive>FALSE</exclusive>
    </node>
    <node component_manager_id="urn:publicid:IDN+topdomain+authority+cm"
component_id="urn:publicid:IDN+topdomain:dummy+node+node9.dummy-testbed.org"
component_name="node9.dummy-testbed.org"
site_id="urn:publicid:IDN+topdomain:dummy+authority+sa">
      <hostname>node9.dummy-testbed.org</hostname>
      <location country="unknown" longitude="123456" latitude="654321"/>
      <exclusive>FALSE</exclusive>
    </node>
    <node component_manager_id="urn:publicid:IDN+topdomain+authority+cm"
component_id="urn:publicid:IDN+topdomain:dummy+node+node10.dummy-testbed.org"
component_name="node10.dummy-testbed.org"
site_id="urn:publicid:IDN+topdomain:dummy+authority+sa">
      <hostname>node10.dummy-testbed.org</hostname>
      <location country="unknown" longitude="123456" latitude="654321"/>
      <exclusive>FALSE</exclusive>
    </node>
  </network>
  <network xmlns:openflow="https://github.com/fp7-ofelia/ocf/blob/ocf.rspecs/openflow/schemas"
name="ocf_of">
    <openflow:datapath
component_id="urn:publicid:IDN+openflow:optin_manager:ocf_of+datapath+00:00:00:00:00:00:00:09"
component_manager_id="urn:publicid:IDN+openflow:optin_manager:i2cat.ocf.of+authority+am"
dpid="00:00:00:00:00:00:00:09">
      <openflow:port num="65534" name="dp0"/>
      <openflow:port num="1" name="s9-eth1"/>
      <openflow:port num="2" name="s9-eth2"/>
    </openflow:datapath>
    <openflow:datapath
component_id="urn:publicid:IDN+openflow:optin_manager:ocf_of+datapath+00:00:00:00:00:00:00:0a"
component_manager_id="urn:publicid:IDN+openflow:optin_manager:i2cat.ocf.of+authority+am"
dpid="00:00:00:00:00:00:00:0a">
      <openflow:port num="3" name="s10-eth3"/>
      <openflow:port num="2" name="s10-eth2"/>
      <openflow:port num="65534" name="dp1"/>
      <openflow:port num="1" name="s10-eth1"/>
    </openflow:datapath>
```

```
   <openflow:datapath
component_id="urn:publicid:IDN+openflow:optin_manager:ocf_of+datapath+00:00:00:00:00:00:00:0d"
component_manager_id="urn:publicid:IDN+openflow:optin_manager:i2cat.ocf.of+authority+am"
dpid="00:00:00:00:00:00:00:0d">
      <openflow:port num="3" name="s13-eth3"/>
      <openflow:port num="2" name="s13-eth2"/>
      <openflow:port num="65534" name="dp4"/>
      <openflow:port num="1" name="s13-eth1"/>
   </openflow:datapath>
   <openflow:datapath
component_id="urn:publicid:IDN+openflow:optin_manager:ocf_of+datapath+00:00:00:00:00:00:00:0e"
component_manager_id="urn:publicid:IDN+openflow:optin_manager:i2cat.ocf.of+authority+am"
dpid="00:00:00:00:00:00:00:0e">
      <openflow:port num="3" name="s14-eth3"/>
      <openflow:port num="2" name="s14-eth2"/>
      <openflow:port num="65534" name="dp5"/>
      <openflow:port num="1" name="s14-eth1"/>
   </openflow:datapath>
   <openflow:datapath
component_id="urn:publicid:IDN+openflow:optin_manager:ocf_of+datapath+00:00:00:00:00:00:00:0f"
component_manager_id="urn:publicid:IDN+openflow:optin_manager:i2cat.ocf.of+authority+am"
dpid="00:00:00:00:00:00:00:0f">
      <openflow:port num="3" name="s15-eth3"/>
      <openflow:port num="2" name="s15-eth2"/>
      <openflow:port num="65534" name="dp6"/>
      <openflow:port num="1" name="s15-eth1"/>
   </openflow:datapath>
   <openflow:datapath
component_id="urn:publicid:IDN+openflow:optin_manager:ocf_of+datapath+00:00:00:00:00:00:00:0b"
component_manager_id="urn:publicid:IDN+openflow:optin_manager:i2cat.ocf.of+authority+am"
dpid="00:00:00:00:00:00:00:0b">
      <openflow:port num="3" name="s11-eth3"/>
      <openflow:port num="2" name="s11-eth2"/>
      <openflow:port num="65534" name="dp2"/>
      <openflow:port num="1" name="s11-eth1"/>
   </openflow:datapath>
   <openflow:datapath
component_id="urn:publicid:IDN+openflow:optin_manager:ocf_of+datapath+00:00:00:00:00:00:00:0c"
component_manager_id="urn:publicid:IDN+openflow:optin_manager:i2cat.ocf.of+authority+am"
dpid="00:00:00:00:00:00:00:0c">
      <openflow:port num="3" name="s12-eth3"/>
      <openflow:port num="2" name="s12-eth2"/>
      <openflow:port num="65534" name="dp3"/>
      <openflow:port num="1" name="s12-eth1"/>
   </openflow:datapath>
   <openflow:link srcDPID="00:00:00:00:00:00:00:0d" srcPort="3" dstDPID="00:00:00:00:00:00:00:09"
dstPort="2"/>
   <openflow:link srcDPID="00:00:00:00:00:00:00:0a" srcPort="2" dstDPID="00:00:00:00:00:00:00:0c"
dstPort="3"/>
   <openflow:link srcDPID="00:00:00:00:00:00:00:0f" srcPort="3" dstDPID="00:00:00:00:00:00:00:0d"
dstPort="2"/>
   <openflow:link srcDPID="00:00:00:00:00:00:00:0a" srcPort="3" dstDPID="00:00:00:00:00:00:00:09"
dstPort="1"/>
   <openflow:link srcDPID="00:00:00:00:00:00:00:0a" srcPort="1" dstDPID="00:00:00:00:00:00:00:0b"
dstPort="3"/>
   <openflow:link srcDPID="00:00:00:00:00:00:00:0d" srcPort="2" dstDPID="00:00:00:00:00:00:00:0f"
dstPort="3"/>
```

```
   <openflow:link srcDPID="00:00:00:00:00:00:00:0d" srcPort="1" dstDPID="00:00:00:00:00:00:00:0e"
dstPort="3"/>
   <openflow:link srcDPID="00:00:00:00:00:00:00:09" srcPort="1" dstDPID="00:00:00:00:00:00:00:0a"
dstPort="3"/>
   <openflow:link srcDPID="00:00:00:00:00:00:00:09" srcPort="2" dstDPID="00:00:00:00:00:00:00:0d"
dstPort="3"/>
   <openflow:link srcDPID="00:00:00:00:00:00:00:0c" srcPort="3" dstDPID="00:00:00:00:00:00:00:0a"
dstPort="2"/>
   <openflow:link srcDPID="00:00:00:00:00:00:00:0b" srcPort="3" dstDPID="00:00:00:00:00:00:00:0a"
dstPort="1"/>
   <openflow:link srcDPID="00:00:00:00:00:00:00:0e" srcPort="3" dstDPID="00:00:00:00:00:00:00:0d"
dstPort="1"/>
  </network>
</RSpec>
```

11.2 Annex 2

11.2.1 Request RSpec to create a FlowSpace:

```
<rspec xmlns="https://github.com/fp7-ofelia/ocf/blob/ocf.rspecs/openflow/schemas/request.xsd"
      xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:openflow="http://www.geni.net/resources/rspec/ext/openflow/3"
      xs:schemaLocation="http://www.geni.net/resources/rspec/3
           https://github.com/fp7-ofelia/ocf/blob/ocf.rspecs/openflow/schemas/request.xsd
           http://www.geni.net/resources/rspec/3/request.xsd
           http://www.geni.net/resources/rspec/ext/openflow/3
           http://www.geni.net/resources/rspec/ext/openflow/3/of-resv.xsd"
   type="request">
  <openflow:sliver email="a@b.com" description="OF request example">
    <openflow:controller url="tcp:192.168.1.1" type="primary"/>
     <openflow:group name="fs1">
       <openflow:datapath
component_id="urn:publicid:IDN+openflow:optin:i2cat.of_ocf+datapath:06:a4:00:12:e2:b8:a5:d0"

component_manager_id="urn:publicid:IDN+openflow:optin:i2cat.of_optin+authority+am"
                      dpid="06:a4:00:12:e2:b8:a5:d0">
          <openflow:port name="ETH0" num="0"/>
          <openflow:port name="ETH1" num="1"/>
        </openflow:datapath>
        <openflow:datapath
component_id="urn:publicid:IDN+openflow:optin:i2cat.of_ocf+datapath:06:af:00:24:a8:c4:b9:00"

component_manager_id="urn:publicid:IDN+openflow:optin:i2cat.of_optin+authority+am"
                      dpid="06:af:00:24:a8:c4:b9:00">
          <openflow:port name="ETH3" num="3"/>
          <openflow:port name="ETH4" num="4"/>
        </openflow:datapath>
     </openflow:group>
     <openflow:match>
        <openflow:use-group name="fs1" />
        <openflow:packet>
          <openflow:dl_type value="0x800" />
          <openflow:nw_src value="10.1.1.0/24" />
          <openflow:nw_proto value="6, 17" />
          <openflow:tp_src value="80" />
        </openflow:packet>
     </openflow:match>
```

```
      <openflow:match>
        <openflow:use-group name="fs1" />
        <openflow:packet>
          <openflow:dl_type value="0x800" />
             <openflow:dl_vlan value= "2"/>
          <openflow:nw_dst value="10.1.1.0/24" />
          <openflow:nw_proto value="6, 17" />
          <openflow:tp_dst value="80" />
        </openflow:packet>
        </openflow:match>
   </openflow:sliver>
 </rspec>
```