FUTURE INTERNET TESTBEDS
EXPERIMENTATION BETWEEN
BRAZIL AND EUROPE

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

FIBRE-EU

Future Internet testbeds/experimentation between Brazil and Europe – EU

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

D5.1 Report on detailed design and development of technology pilots

Author: WP5
Revised by:
Iara Machado (RNP)
Due date of the Deliverable: Month 22
Actual submission date: 24/05/2013
Start date of project: June 1st 2011 Duration: 34 months
Version: v.1.0

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

FP7 Grant Agreement No.	288356
CNPq Grant Agreement No.:	590022/2011-3
Project Name	Future Internet testbeds/experimentation between Brazil and Europe – EU
Document Name	FIBRE-D5.1-v1.0
Document Title	D5.1: Report on detailed design and development of technology pilots
Workpackage	WP5
Authors	Mayur Channegowda (UNIVBRIS) Giacomo Bernini (NXW) Alessandro Canessa (NXW) Roberto Monno (NXW) Natalia Castro Fernandes (UFF) Luiz Claudio ScharaMagalhães (UFF) Francisco Manuel Carrami (UFF) Marcos Rogerio Salvador (CPqD) Fernando Redigolo (USP) Marco AntônioRoja (USP)
Editor	Mayur Channegowda (UNIVBRIS)
Reviewers	Iara Machado (RNP)
Delivery Date	24/05/2013
Version	V1.0

Abstract

The aim of the work package is to develop local and federated technology pilots and validate the developed local and federated FIBRE facilities through showcases, which have been pre-selected from a number of use cases. This document outlines the task 5.1 outcomes which describes the design and development of the technology pilots. The development activities follow a common process based on the requirements analysis of all the pre-defined use cases and the detailed design of the local and federated facilities.

This document defines the three pre-selected FIBRE use cases with respect to the use case description, the use requirements, the building blocks and the required connectivity. It outlines the various components of the involved building blocks and the interfacing between them. The purpose of this document is to give more details about the use cases' building block design and their implementation. It also details the design of the building blocks and OpenFlow related enhancements that are required to implement the technology pilots.

TABLE OF CONTENTS

List of Figures

List of Tables

1 Acronyms

AM	Aggregate Manager
API	Application Programming Interface
BoD	Bandwidth on Demand
CDN	Content Delivery Network
CDS	Content Delivery Server
CLI	Command Line Interface
CORBA	Common Object Request Broker Architecture
CRUD	Create, Read, Update and Delete
CP	Control Plane
DPID	DataPath IDentifier
DPS	Discovery Packet Switches
E-OFC	Extended-OpenFlow Controller
FIBRE	Future Internet testbeds / experimentation between Brazil and Europe
F-PCE	Flow-aware Path Computation Element
FPGA	Field-Programmable Gate Array
FV	FlowVisor
GMPLS	Generalized Multi-Protocol Label Switching
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
IM	Island Manager
LLDP	Link Layer Discovery Protocol
LSP	Label Switched Path
MS	Milestone
NE	Network Element
NMI	Network Management Interface
NOX	OpenFlow Controller
OCF	OFELIA Control Framework
OF	OpenFlow
OFELIA	OpenFlow in Europe: Linking Infrastructure and Applications
OFV	Optical FlowVisor
OMF	cOntrol, Management and Measurement Framework
OML	ORBIT Measurement Library
OSPF-TE	Open Shortest Path First - Traffic Engineering
OSS	Open Source Software
p.	Page
QoS	Quality of Service
ROADM	Reconfigurable Optical Add-Drop Multiplexer
RSVP-TE	Resource Reservation Protocol - Traffic Engineering
SDN	Software Defined Networking

SFA	Slice-based Federation Architecture
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
TED	Traffic Engineering Database
UDP	User Datagram Protocol
UNI	User Network Interface
VLAN	Virtual Local Area Network
VM	Virtual Machine
VON	Virtual Optical Network
VT	Virtualization Technology
WP1	Project Management
WP2	Building and operating the Brazilian facility
WP3	Building and operating the European facility
WP4	Federation of facilities
WP5	Development of technology pilots and showcases
WP6	Dissemination and collaboration

2 Scope

This deliverable presents the FIBRE use cases building blocks that include Seamless mobility, High Definition content delivery across different sites and Bandwidth on Demand through OpenFlow GMPLS.

3 Reference Documents

1. [FIBRE] http://www.fibre-ict.eu/
2. [OFELIA] http://www.fp7-ofelia.eu/
3. [NOX] http://www.noxrepo.org/
4. [NOX REPO] https://github.com/noxrepo/nox-classic
5. [NOX WIKI] https://github.com/noxrepo/nox-classic/wiki
6. OpenFlow Switch Specification version 1.3.0 - https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf
7. OpenFlow Switch Specification version 1.0.1 - https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.1.pdf
8. Extensions to the OpenFlow Protocol in support of Circuit Switching - http://www.openflow.org/wk/images/8/81/OpenFlow_Circuit_Switch_Specification_v0.3.pdf
9. FP6 PHOSPHORUS – Deliverable D2.3 "Grid-GMPLS high-level system design"
10. [IETF RFC 4655] – "A Path Computation Element (PCE)-Based Architecture", Informational RFC, Aug. 2006
11. Milestone MS5.1: Report on technology building blocks
12. M. Channegowda, P. Kostecki, N. Efstathiou, S. Azodolmolky, R. Nejabati, P. Kaczmarek, A. Autenrieth, J.P. Elbers, D. and Simeonidou, "Experimental Evaluation of Extended OpenFlow Deployment for High-Performance Optical Networks," in Proceedings of ECOC, Tu.1.D.2 (2012).
13. Okung-Dike Ntofon, Mayur P. Channegowda, Nikolaos Efstathiou, Mehdi Rashidi Fard, Reza Nejabati, David K. Hunter, and Dimitra Simeonidou, "Experimental demonstration of OpenFlow-enabled media ecosystem architecture for high-end applications over metro and core networks," Opt. Express 21, 4811-4816 (2013).
14. [WireShark] http://www.wireshark.org/

4 FIBRE Technology Pilots

4.1 Technology Pilot 1: Seamless Mobility

Table 1: Seamless Mobility

Technology Pilot	
Scenario/Use Case name	Seamless Mobility
Organisation Involved	UFF, UFG, UFPA, UFRJ, UFSCar
Summary Scenario (storyline)	The recent increase in the use of mobile devices is a reality worldwide, impacting on all social spheres. Therefore, techniques to provide a seamless mobility with efficiency, high availability, and security become increasingly necessary. This research intends to leverage this scenario, offering the infrastructure for shared network experimentation. The central purpose is to develop and analyse techniques for providing a seamless experience on network connectivity. The ways for doing this is improving mesh models, enhancing handoff latency, and using multiple network interfaces simultaneously.
Goal	To analyse and utilize the capabilities of wireless networks to augment the seamless handoffs experience on networks formed by mobile users (people using smart phones, tablets, netbooks, and notebooks). In particular, to exploit programmability to perform multipath communication to support various levels of mobility. Moreover, to study, deploy and extend OMF to share large scale distributed mobile networks to be used by various network/wireless researchers. Also, create an environmental monitoring tool to support the analysis of the test results.
User community	At the outset, Brazilian researchers from the organizations involved in the project will be the main users of this facility, but the testbeds will be available to the worldwide members after federation.
Figure to visualise the Use case scenario	See Figure 1 below showing the Seamless Mobility testbed and the experiment setup.
Features of the Use Case	This use case will show the use of FIBRE wireless testbed, which can be used to experiment with different kinds of wireless networks, such as infrastructured networks, mesh networks, ad hoc networks, DTNs, wireless OpenFlow networks, etc. We will develop experiments for analysing new techniques

	to improve wireless network performance. These experiments include subjects such as vertical and horizontal handoff, wireless monitoring, and spectrum sharing models. Features such as load balancing, reliability, and QoS will be considered in these experiments. This pilot will also build an extensive wireless monitoring framework. The main idea is to provide to researchers a full understanding of the impact of the environment over the performed test. Moreover, by storing and making this data available, we will also contribute with initiatives for artificially recreating wireless scenarios and for understanding and predicting the interference in wireless scenarios.
Virtualization layer and involved network/IT resources	• The wireless network infrastructure available in islands • The monitoring framework • Applications to be run FIBRE
Hardware Requirements	• Orbit nodes, • Mobile devices (model trains equipped with wireless devices), • Bluetooth cards, • WiMax base station, • WiMax cards, • 3G cards, • WiFi cards, • Monitoring server • Control framework server • Virtual machine server • USB GPS devices • Spectrum analysers
Access method to the testbed (FIBRE Islands)	Accessed within FIBRE islands by User or Researchers
Access Method to the Infrastructure Slice	Accessed through the OMF scheduler
OpenFlow Controller Requirements	Does not apply. The use of OpenFlow is not mandatory.
Monitoring Requirements	Statistic regarding: • geo-location, • bandwidth, • delay, • inter-time between mobile users encounters, • forwarding tables,

	• antennas signal noise ratio • spectrum usage • packets • losses • user activity • topology
Security	The access to the testbed will be performed according to FIBRE polices. The user data collected by the monitoring system will be treated to guarantee user anonymity. The use of cryptographic methods during tests will be recommended, but will not be mandatory.
Life-cycle of Service (Phases from start to end)	Phase 1 - Controlled and instrumented testbed. In this phase, we will install the equipment, deploy the control and monitoring framework, develop the interface for controlling the mobile devices, and provide the access to the tests through web. Phase 2 – Development of application software. In this phase, we will develop the software for performing the seamless mobility and for monitoring nodes.
Preconditions	Individualized operating environment & profile.
Post conditions	
Other comments	

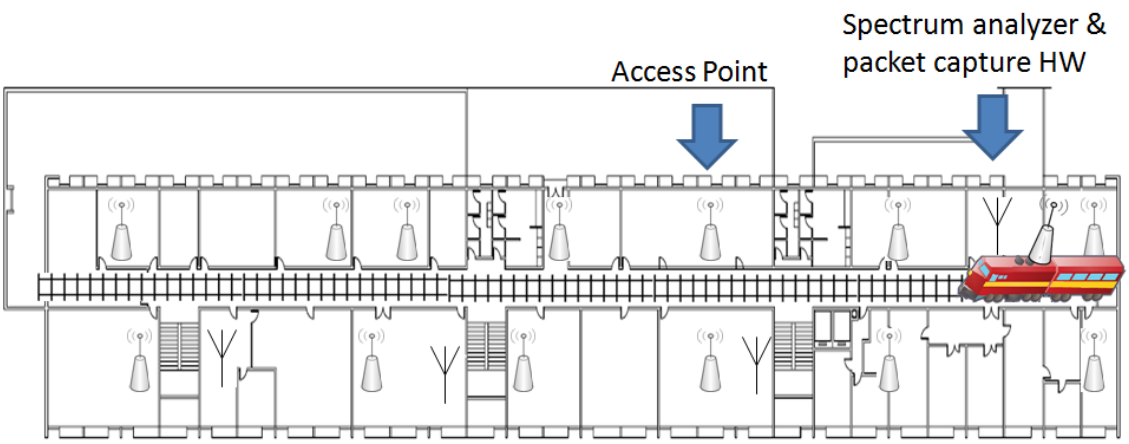

Figure 1: Seamless Mobility testbed

4.2 Technology Pilot 2: High definition content delivery across different sites

Table 2: High definition content delivery across different sites

Technology Pilot	
Scenario/Use Case name	High definition content delivery across different sites
Organisation Involved	UNIVBRIS, NXW, CPqD, RNP, USP
Summary Scenario (storyline)	An OpenFlow-based application (i.e. a NOX application) can be interfaced to one or more Content Delivery Servers (CDSs) that form a Content Delivery Network (CDN). The NOX application [3] will be able to monitor the CDS performance by retrieving the related status, load and failures. When certain thresholds are exceeded (e.g. the load on CDS or its energy consumption), NOX application will re-route one or more clients to another CDS located in another site. The re-routing will be performed and facilitated by NOX application which will change the flow tables of the OF switches under its control.
Goal	This use-case takes advantage of the functionalities provided by the OpenFlow Control Plane. In particular, it exploits the flexibility provided by "OpenFlow" and "Flow Routing" to properly distribute the traffic and load balance requests from clients/users of a high-definition video streaming service between delivery sites (CDSs located in different sites).
User community	In the pilot/lab scale experiments, only the partners involved in this use-case, but later it could be used by the other members of FIBRE.
Figure to visualise the Use case scenario	See Figure 2 below showing re-direction of clients by NOX controller.
Features of the Use Case	In this setup the UK site is responsible for delivering high quality video (e.g., 4K streaming) and the site in Brazil receives a copy of the content (cache). Assuming that equipment (packet and optical) in both sites are OF-enabled, the European and Brazilian CDSs can be interfaced with a NOX application capable to monitor their performance. Therefore, in a scenario where football matches are streamed during the World CUP 2014, clients in UK (or even Europe) will be served by the UK site. When a certain server usage threshold is exceeded (e.g. the load on the UK-CDS), NOX application will re-route some of the content-delivery traffic to another (less crowded, but not necessarily closer) CDS (e.g. Brazil site). In order to take proper re-routing decisions, NOX

	applications require to retrieve a detailed description of the network topology and the updated information regarding the performance and failures of the controlled CDS. The considered thresholds and their values will be specific and configurable parameter of the NOX application. So depending on the criteria of interest (load balancing, energy savings, cheaper electricity bills, QoS, failures), NOX application will take the decision and trigger the appropriate actions. It is worth noting that a close interworking of NOX and monitoring applications is also required (e.g. for migrating some users from their current site to another one 'less-loaded').
Virtualization layer and involved network/IT resources	• The OF enabled network infrastructure in UK and Brazil site • The High Definition/High Quality media streaming client/server (codec) • CDS in UK and in Brazil site • NOX application capable to monitor CDS
Hardware Requirements	• OpenFlow enabled islands in UK and Brazil • 4K streaming or high quality streaming servers for ordinary users • 4K Projection facility or ordinary clients (stream player) • OF Controller Requirements
Access method to the testbed (FIBRE Islands)	Accessed within FIBRE islands by User or Researchers
Access Method to the Infrastructure Slice	Individual access to OF enabled CDS
OpenFlow Controller Requirements	OpenFlow controller (e.g. NOX): • Enhanced functionalities to support also optical circuit switching NEs [8]. • NOX application (e.g. a Topology Manager) capable to retrieve detailed network topology information • NOX application (e.g. a Monitoring manager) capable to monitor CDS performance (status, load and failures) • NOX application (e.g. a Replanning Manager) to trigger re-routing when certain thresholds (e.g. OF resources load) are exceeded or when a link failure is detected • NOX application (e.g. Provisioning Manager) to handle the addition/deletion/update of the flow entries.

European Commission Information Society and Media · COOPERATION · CNPq 60 ANOS · Ministério da Ciência, Tecnologia e Inovação · GOVERNO FEDERAL BRASIL PAÍS RICO É PAÍS SEM POBREZA

Monitoring Requirements	Network Monitoring need to provide user-to-server statistics regarding: • Available bandwidth • One-way delay • Link utilization • Packet loss Content Delivery Server Monitoring need to provide pure server statistics regarding: • Status (memory usage, CPU, etc.) • Load (e.g. number of active flows) • Failures
Security	Testbed authenticated users
Life-cycle of Service (Phases from start to end)	2013 -2014
Preconditions	• Contents available and/or cached in UK-CDS and BR-CDS • One of the two CDSs is overloaded
Post conditions	• Part of the content-delivery traffic is re-routed towards the ‘less-loaded’ CDS • Re-routed content-delivery traffic flows through a new network path (new/updated flow entries in the OF resources, traffic log)
Other comments	

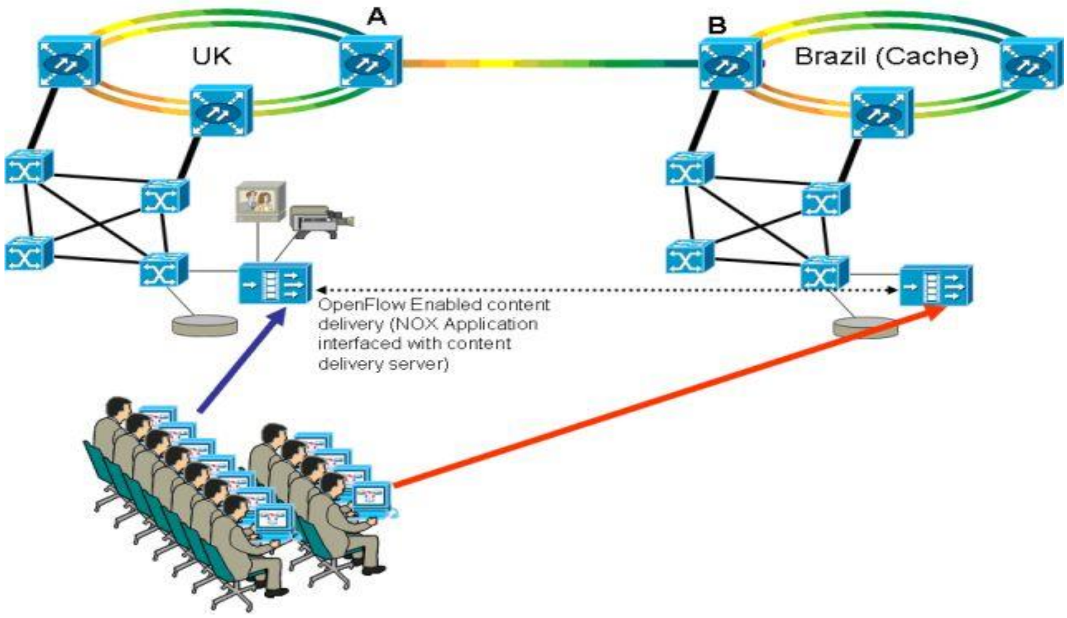

Figure 2: High-definition content delivery

4.3 Technology Pilot 3: Bandwidth on Demand through OpenFlow GMPLS in the FIBRE facility

Table 3: Bandwidth on Demand through OpenFlow GMPLS in the FIBRE facility

Technology Pilot	
Scenario/Use Case name	Bandwidth on Demand through OpenFlow GMPLS in the FIBRE facility
Organisation Involved	UNIVBRIS, NXW, CPqD, RNP, UFPA
Summary Scenario (storyline)	The OF protocol and the NOX control platform allow to generalize the control of the switching resources of a network node and to implement a flow-processing paradigm in the traffic switching. Moreover, the FlowVisor (FV) controller acts as a transparent proxy between physical OF switches and one or more OF controllers. FV allows the creation of a virtualized topology, in the form of multiple network resources slices, by delegating the control of each slice to a different controller. The functionalities provided by the OF CP may be integrated with the effectiveness of GMPLS signalling and routing protocols (e.g. RSVP-TE and OSPF-TE) in the distributed set-up and operation of on-demand network connectivity services.
Goal	This use-case exploits the flexibility of the OF, FV and GMPLS protocols to implement an open and generalized Bandwidth on Demand (BoD) service for OF controllers on virtualized networks.
User community	In the pilot/lab scale experiments, only the partners involved in this use-case, but later it could be used by the other members of FIBRE.
Figures to visualise the Use case scenario	Two potential deployment scenarios have been individuated: • OpenFlow overlay on GMPLS (see Figure 3) • GMPLS overlay on OpenFlow (see Figure 4)
Features of the Use Case	The BoD User Network Interface (UNI) will allow the user to request for dynamic flow reservations across OF-enabled optical layer. Two potential scenarios have been identified for the deployment of this Pilot by highlighting the related advantages and potential disadvantages. In the Scenario#1 (see Figure 3), the BoD requests coming from the UNI are processed by the OpenFlow Control Plane. A set of extended OF controllers will be deployed to control the network slices provided by the extended FV controller (e.g. Optical FlowVisor), which will operate on top of a heterogeneous physical infrastructure. In

	particular, the extended FV controller will be capable to interface with both OF and GMPLS enabled domains, to provide both virtualization/slicing functionalities and on-demand network connectivity services. The GMPLS control plane will be capable to abstract the topology under its ownership for proper extended FV controller usage, e.g. modelling the whole domain as an abstract OF node. Moreover, GMPLS will be capable to translate network connectivity requests coming from the extended FV controller into GMPLS proper actions (e.g. LSP signalling). In the Scenario#2 (see Figure 4), the GMPLS control plane will process the BoD requests from the UNI. A set of GMPLS controllers will be deployed to control the extended OF controllers. These extended OF controllers will be responsible for managing the network slices provided by the extended FV. With respect to the Scenario#1, the whole heterogeneous packet/optical physical infrastructure will be OF enabled. The GMPLS controllers will provide BoD network connectivity services by implementing enhanced routing and signalling procedures, and by interfacing with the underlying extended OF controller to configure the flow tables in the OF switches. For both scenarios, an extended Path/Flow Computation Engine (Flow-aware PCE) will be deployed to offer path and flow computation functionalities to both OF and GMPLS control plane.
Virtualization layer and involved network/IT resources	• The OF enabled network infrastructure in UK and Brazil site, including optical and MPLS switching hardware • The OSS GMPLS stack developed by the PHOSPHORUS project • A high-bandwidth application (e.g. streaming as per Pilot 2)
Hardware Requirements	• At least 3 OpenFlow-enabled (or open firmware) ROADMs per islands, with one island in Bristol and one island in CPqD
Access method to the testbed (FIBRE Islands)	Accessed within FIBRE islands by User or Researchers
Access Method to the Infrastructure Slice	The slices access will be guaranteed to the BoD users by using a proper setup/configuration of the extended-FVs in order to reserve the required resources for the Technology

	Pilot.
OpenFlow Controller Requirements	The functionalities of the common OF Controllers available in the state of the art (e.g. NOX) need to be enhanced to support also circuit switching [8]. Further functionalities are required according to the deployment scenario used (ref. Figure 3 and Figure 4). Requirements for Scenario #1: • Enhanced OF Control Plane Southbound Interface for Label Switched Path (LSP) setup in the GMPLS domain • Modelling of the GMPLS domain as an OF abstract node, and mapping installed LSPs into appropriate flow entries Requirements for Scenario #2: • Extensions to the OF Control Plane Northbound interface for interaction with GMPLS • Full knowledge of the whole heterogeneous topology (OF switches and OF-enabled optical NEs), controlled only via OF protocol (e.g. through a Topology Manager functionality)
Monitoring Requirements	For both potential identified scenarios, the retrieval of the packet/optical network resources status is required. According to the used scenario, some specific requirements can be identified. For Scenario#1: • Mechanisms for mapping optical NEs status into appropriate OF messages/statistics related to the aforementioned "abstract" node seen by OF controller For Scenario#2: • Status notification of the OF resources (packet and optical) for routing and recovery purposes.
Security	Testbed authenticated users
Life-cycle of Service (Phases from start to end)	2012 - 2013
Preconditions	• One of the two identified scenarios has been deployed
Post conditions	• The setup of the requested on-demand network connectivity services has been performed successfully (new/updated flow entries in the OF resources, traffic log, etc.)
Other comments	

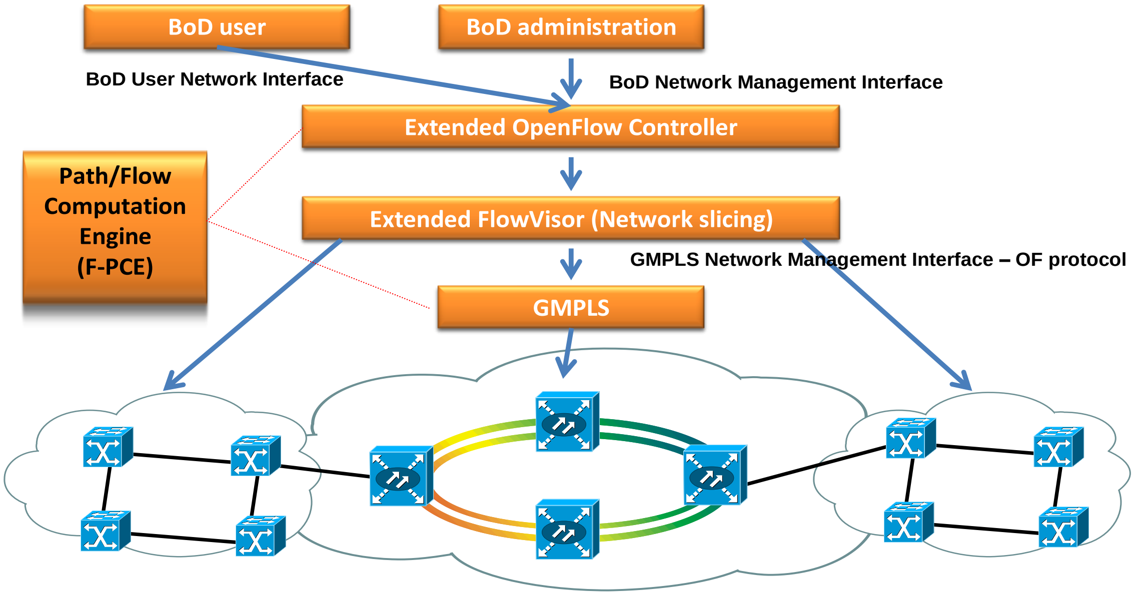

Figure 3: OpenFlow overlay on GMPLS BoD

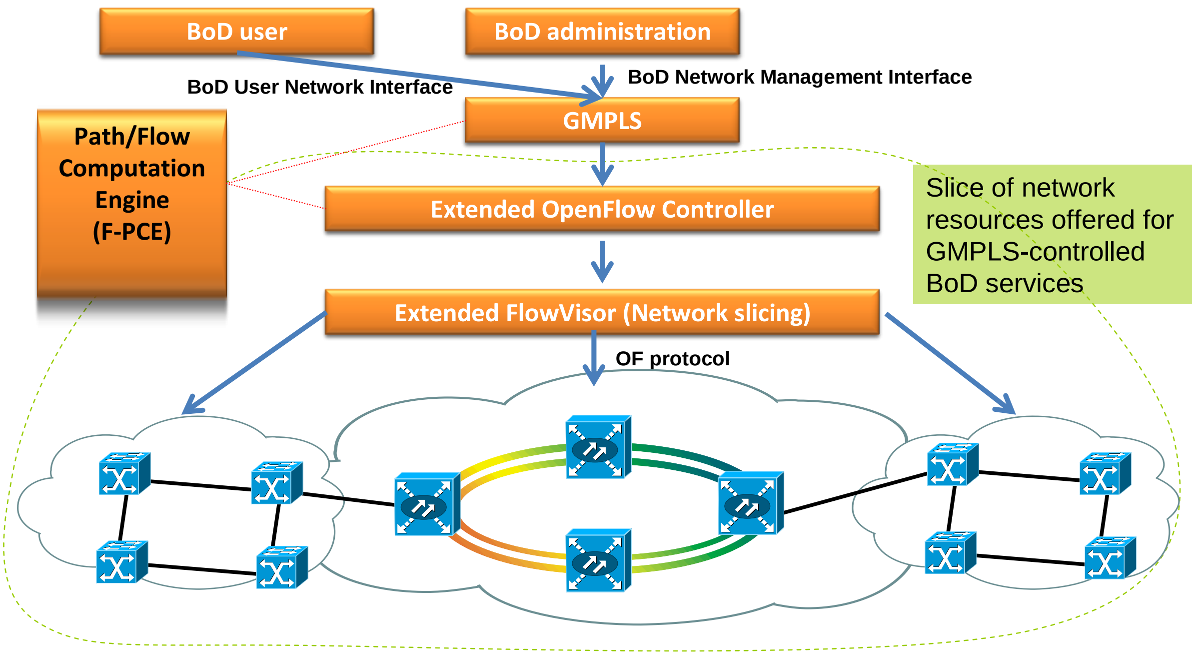

Figure 4: GMPLS overlay on OpenFlow

5 Technology pilot analysis

This section analyses the technology pilots and classifies their requirements. Furthermore, it maps the classified technology pilot requirements into different FIBRE high-level (Heterogeneous physical resources requirements, virtualization requirements, Slice and Management requirements, Experiment control and management) requirements.

There are two types of requirements:

- Functional requirements: are requirements relevant to technological capability and functionality of the technology pilots in FIBRE test-bed.
- Non-functional requirements: are a set of constraints (also known as quality requirements) that define performance expectation and quality of service (QoS) of the technology pilots in FIBRE test-bed.

These enhanced requirements help identify the necessary use case building blocks and provide a base for the software development.

The FIBRE technology pilots' requirements (both functional and non-functional) are classified and grouped into following categories mentioned below.

5.1 Heterogeneous Physical Resources Requirements (PR)

This category deals with the physical resources (non-virtualized) such as the network nodes, server nodes, traffic source which can be virtualized based on the requirements. Below the requirements are listed in terms of physical resources.

Table 4: OpenFlow L2 infrastructure

Requirement ID	PR01
Name	OpenFlow L2 infrastructure
Description	Requirements for using a network topology comprised of OpenFlow-enabled L2 nodes
Type	Functional
Related use case	Tech. Pilot #2, Tech. Pilot #3
Additional note	

European Commission Information Society and Media
COOPERATION
CNPq 60 ANOS
Ministério da Ciência, Tecnologia e Inovação
GOVERNO FEDERAL BRASIL PAÍS RICO É PAÍS SEM POBREZA

Table 5: OpenFlow-enabled ROADMs

Requirement ID	PR02
Name	OpenFlow-enabled ROADMs
Description	Requirements for using a network topology comprised of OpenFlow-enabled L1 nodes
Type	Functional
Related use case	Tech. Pilot #2, Tech. Pilot #3
Additional note	• 3 OpenFlow-enabled ROADMs per island

Table 6: Content Delivery Servers

Requirement ID	PR03
Name	Content Delivery Servers
Description	Requirements for using Content Delivery Servers
Type	Functional
Related use case	Tech. Pilot #2
Additional note	• 1 x 4K compression boards per Server

Table 7: Wireless access points

Requirement ID	PR04
Name	WiFi/Bluetooth/WiMax access points
Description	Requirements for using WiFi/Bluetooth/WiMax access points
Type	Functional
Related use case	Tech. Pilot #1
Additional note	

Table 8: Controlled mobile devices on model trains

Requirement ID	PR05
Name	Controlled mobile devices on model trains
Description	Requirements for using mobile devices
Type	Functional
Related use case	Tech. Pilot #1

Additional note	Model trains equipped with devices containing WiFi, Bluetooth, WiMax , and 3G interfaces

Table 9: Experimental storage space

Requirement ID	**PR06**
Name	Experimental storage space
Description	Requirements for using storage space to store 4K video/ monitoring data.
Type	Functional
Related use case	Tech. Pilot #1, Tech. Pilot #2, Tech. Pilot #3
Additional note	• ~5TB storage space required

Table 10: High Definition media streaming server/client

Requirement ID	**PR07**
Name	High Definition media streaming server/client
Description	Requirements for using HD media streaming server/client
Type	Functional
Related use case	Tech. Pilot #2, Tech. Pilot #3
Additional note	

Table 11: GMPLS controllers

Requirement ID	**PR08**
Name	GMPLS controllers
Description	Requirements for using a set of GMPLS controllers (e.g.: the GMPLS Edge, Core and Border Controllers) by installing and configuring an OSS GMPLS stack [9]
Type	Functional
Related use case	Tech. Pilot #3
Additional note	

Table 12: High bandwidth application

Requirement ID	**PR09**
Name	High bandwidth application

Description	Requirements for using High bandwidth application
Type	Functional
Related use case	Tech. Pilot #2, Tech. Pilot #3
Additional note	

Table 13: OpenFlow controller

Requirement ID	**PR10**
Name	OpenFlow controller
Description	Requirements for using OpenFlow controller
Type	Functional
Related use case	Tech. Pilot #2, Tech. Pilot #3
Additional note	

Table 14: Monitoring Manager

Requirement ID	**PR11**
Name	Monitoring Manager
Description	Requirements for using NOX application capable to retrieve the CDS status/load
Type	Functional
Related use case	Tech. Pilot #2
Additional note	

Table 15: Flow-aware Path Computation Element network application

Requirement ID	**PR12**
Name	Flow-aware Path Computation Element (F-PCE) network application
Description	Requirements for using F-PCE application for the path and flow computation in a multi-domain network with multiple switching regions (e.g. packets and optical)
Type	Functional
Related use case	Tech. Pilot #2, Tech. Pilot #3
Additional note	

Table 16: Topology Manager

Requirement ID	**PR13**

Name	Topology Manager
Description	Requirements for using Topology Manager functional entity to retrieve the topology info for the OpenFlow regions
Type	Functional
Related use case	Tech. Pilot #2, Tech. Pilot #3
Additional note	For the Scenario#1 ("OF overlay on GMPLS") of the Tech. Pilot #3, Topology Manager also retrieves the ingress/egress nodes and ports (Transport Network Assigned address) of the GMPLS domains.

Table 17: Provisioning Manager

Requirement ID	**PR14**
Name	Provisioning Manager
Description	Requirements for using Provisioning Manager functional entity to handle the flow entries addition/deletion/update in the flow table of the OF switches
Type	Functional
Related use case	Tech. Pilot #2, Tech. Pilot #3
Additional note	For the Scenario#1 ("OF overlay on GMPLS") of the Tech. Pilot #3, Provisioning Manager may be also interfaced to the GMPLS control plane for LSP setup.

Table 18: Load Balancing application

Requirement ID	**PR15**
Name	Load Balancing application
Description	Requirements for media client Load Balancing based on user traffic and forward traffic to appropriate content delivery servers
Type	Functional
Related use case	Tech. Pilot #1, Tech. Pilot #2, Tech. Pilot #3
Additional note	

Table 19: Replanning Manager

Requirement ID	**PR16**
Name	Replanning Manager
Description	Requirements for using Replanning Manager functional entity to trigger re-routing when certain thresholds are exceeded or when a link failure is detected

Type	Functional
Related use case	Tech. Pilot #2
Additional note	

Table 20: Railway track

Requirement ID	PR17
Name	Railway track
Description	Plan and install the railway track for the model train
Type	Functional
Related use case	Tech. Pilot #1
Additional note	

Table 21: Measurement points

Requirement ID	PR18
Name	Measurement points
Description	Plan and install measurement points to monitor the wireless environment
Type	Functional
Related use case	Tech. Pilot #1
Additional note	

Table 22:Location of access points

Requirement ID	PR19
Name	Location of access points
Description	Plan the proper location and install the access points
Type	Functional
Related use case	Tech. Pilot #1
Additional note	

Table 23: Hardware for model train speed control

Requirement ID	PR20
Name	Hardware for model train speed control
Description	Develop and install the hardware for model train speed control
Type	Functional
Related use case	Tech. Pilot #1

Additional note	

Table 24: Deployment of wireless interfaces in the model train

Requirement ID	PR21
Name	Deployment of wireless interfaces in the model train
Description	Project and deploy the support for wireless interfaces in the model train
Type	Functional
Related use case	Tech. Pilot #1
Additional note	

Table 25: Hardware for model train battery recharging

Requirement ID	PR22
Name	Hardware for model train battery recharging
Description	Project and develop the hardware for model train battery recharging. The battery will be used by the wireless interfaces coupled to the model train.
Type	Functional
Related use case	Tech. Pilot #1
Additional note	

Table 26: Hardware modules integration and validation

Requirement ID	PR23
Name	Hardware modules integration and validation
Description	Integrate and validate all the wireless testbed environment
Type	Functional
Related use case	Tech. Pilot #1
Additional note	

Table 27: Horizontal handoff service

Requirement ID	PR24
Name	Horizontal handoff service
Description	Application to perform horizontal handoff according to

	available access points
Type	Functional
Related use case	Tech. Pilot #1
Additional note	

Table 28: Vertical handoff service

Requirement ID	PR25
Name	Vertical handoff service
Description	Application to perform vertical handoff according to available wireless interfaces
Type	Functional
Related use case	Tech. Pilot #1
Additional note	

Table 29: Speed control firmware

Requirement ID	PR26
Name	Speed control firmware
Description	Firmware required by the speed control hardware of the model train
Type	Functional
Related use case	Tech. Pilot #1
Additional note	

Table 30: Battery recharger manager

Requirement ID	PR27
Name	Battery recharger manager
Description	Firmware required by the battery recharger hardware of the model train
Type	Functional
Related use case	Tech. Pilot #1
Additional note	

5.2 Virtualization Requirements (VR)

Virtualization Requirement category deals with the virtualization needed on network nodes, servers nodes for creating an isolated infrastructure slice. Below the requirements are listed in terms of resource virtualization.

Table 31: L2 network slice

Requirement ID	VR01
Name	L2 network slice
Description	Requirements for a slice of L2 network which is isolated from other slices.
Type	Functional
Related use case	Tech. Pilot #2, Tech. Pilot #3
Additional note	

Table 32: L1 network slice

Requirement ID	VR02
Name	L1 network slice
Description	Requirements for a slice of L1 network which is isolated from other slices.
Type	Functional
Related use case	Tech. Pilot #1, Tech. Pilot #2, Tech. Pilot #3
Additional note	

Table 33: Virtualization mechanism for traffic isolation

Requirement ID	VR03
Name	Virtualization mechanism for traffic isolation
Description	Virtualization techniques used in the FIBRE facility shall provide traffic isolation between virtual machines residing in the same physical host.
Type	Functional
Related use case	Tech. Pilot #1, Tech. Pilot #2, Tech. Pilot #3
Additional note	

5.3 Slice control and Management (Admin) Requirements

This category contains the requirements that are necessary for controlling and managing an island for an Administrator. It is very essential for the admin to have control over the entire infrastructure to make it function effectively. Below the requirements for slice control and management are listed from the perspective of the island administrator.

Table 34: L2 slice creation, allocation, monitoring and release

Requirement ID	AR01
Name	L2 slice creation, allocation, monitoring and release
Description	Island administration capability and tools for creation, allocation, monitoring and release of L2 slices
Type	Non-Functional
Related use case	Tech. Pilot #2, Tech. Pilot #3
Additional note	This can be needed for any of Tech. Pilots #2 and #3

Table 35: L1 slice creation, allocation, monitoring and release

Requirement ID	AR02
Name	L1 slice creation, allocation, monitoring and release
Description	Island administration capability and tools for creation, allocation, monitoring and release of L1 slices
Type	Non-Functional
Related use case	Tech. Pilot #1, Tech. Pilot #2, Tech. Pilot #3
Additional note	This can be needed for any Tech. Pilot

Table 36: Application server and traffic source/sink monitoring, allocation and release

Requirement ID	AR03
Name	Application server and traffic source/sink monitoring, allocation and release
Description	Island administration capability and tools for allocation, monitoring and release of servers (both real and virtual)
Type	Non-Functional
Related use case	Tech. Pilot #2, Tech. Pilot #3
Additional note	This can be needed for any of Tech. Pilots #2 and #3

Table 37: Island overall L2 resources/network monitoring

Requirement ID	**AR04**
Name	Island overall L2 resources/network monitoring
Description	Island administration capability to monitor status of island L2 resources and connectivity
Type	Non Functional
Related use case	Tech. Pilot #1, Tech. Pilot #2, Tech. Pilot #3
Additional note	This can be needed for any Tech. Pilot

Table 38: Island overall L1 resources/network monitoring

Requirement ID	**AR05**
Name	Island overall L1 resources/network monitoring
Description	Island administration capability to monitor status of island L1 resources and connectivity
Type	Non Functional
Related use case	Tech. Pilot #1, Tech. Pilot #2, Tech. Pilot #3
Additional note	This can be needed for any Tech. Pilot

Table 39: Island overall server resources monitoring

Requirement ID	**AR06**
Name	Island overall server resources monitoring
Description	Island administration capability to monitor the health, traffic generated from the Physical servers and Virtual machines.
Type	Non-Functional
Related use case	Tech. Pilot #1, Tech. Pilot #2, Tech. Pilot #3
Additional note	

Table 40: Monitoring of WAN links

Requirement ID	**AR07**
Name	Monitoring of WAN links
Description	Administrators shall be provided with tools to monitor WAN links used to interconnect FIBRE islands and report any error or fault on the link.
Type	Non-Functional
Related use case	Tech. Pilot #2, Tech. Pilot #3
Additional note	This is a general requirement for interconnectivity between

	islands

Table 41: Authentication, Authorization and Accounting

Requirement ID	AR08
Name	Authentication, Authorization and Accounting
Description	Controlled and secured access to the FIBRE facility
Type	Non- Functional
Related use case	Tech. Pilot #1, Tech. Pilot #2, Tech. Pilot #3
Additional note	

5.4 Experiment Control and Management (Experimenter) Requirements

This category contains requirements as seen from the perspective of an experimenter. The requirement components detailed in this category will be important and essential for the experimenter to use the facility and carry out research experiments successfully. Below requirements for experiment creation, control and management are listed from the perspective of the researchers/users.

Table 42: Experiment Traffic monitoring

Requirement ID	ER01
Name	Experiment traffic monitoring
Description	Capability to monitor traffic, measure bandwidth utilization across the infrastructure slice
Type	Functional
Related use case	Tech. Pilot #1, Tech. Pilot #2, Tech. Pilot #3
Additional note	

Table 43: Experiment user-to-server monitoring

Requirement ID	ER02
Name	Experiment user-to-server monitoring
Description	Capability to monitor user-to-server performance by retrieving related statistics (e.g. one-way delay, packet loss, available bandwidth and link utilization)
Type	Functional
Related use case	Tech. Pilot #2
Additional note	User-to-server statistics are required to trigger re-routing and to migrate bunch of users with the same characteristics to a new CDS.

Table 44: Experiment network node monitoring

Requirement ID	ER03
Name	Experiment network node monitoring
Description	User capability to monitor all the network nodes, which are part of the slice.
Type	Functional
Related use case	Tech. Pilot #1, Tech. Pilot #2, Tech. Pilot #3
Additional note	OpenFlow Stats/monitoring framework

Table 45: Experiment end node monitoring

Requirement ID	ER04
Name	Experiment End node monitoring
Description	User capability to monitor traffic for VMs, write logs for the end nodes.
Type	Functional
Related use case	Tech. Pilot #1, Tech. Pilot #2, Tech. Pilot #3
Additional note	

Table 46: SSH support for External users

Requirement ID	ER05
Name	SSH support for External users
Description	SSH access to physical servers, virtual machines and controllers for users using single logon
Type	Functional
Related use case	Tech. Pilot #1, Tech. Pilot #2, Tech. Pilot #3
Additional note	SSH access authenticated via a centralized entity (Single Logon)

Table 47: Federation Connectivity

Requirement ID	ER06
Name	Connectivity between federated Islands
Description	Able to reserve & provision resources from multiple islands with adequate QoS
Type	Functional
Related use case	Tech. Pilot #2, Tech. Pilot #3
Additional note	

Table 48: Integration of the model train and OMF

Requirement ID	ER07
Name	Integration of the model train and OMF
Description	The model train control must be performed by experimenters through OMF. Hence, we must develop an interface to integrate the control software and OMF.
Type	Functional

Related use case	Tech. Pilot #1
Additional note	

Table 49: Integration of the Bluetooth interfaces and OMF

Requirement ID	ER08
Name	Integration of the Bluetooth interfaces and OMF
Description	OMF must control the parameters of the Bluetooth interface, which is available in the train and in the access points.
Type	Functional
Related use case	Tech. Pilot #1
Additional note	

Table 50: Integration of the WiMax interfaces and OMF

Requirement ID	ER09
Name	Integration of the WiMax interfaces and OMF
Description	OMF must control the parameters of the WiMax interface, which is available in the train and in the access points.
Type	Functional
Related use case	Tech. Pilot #1
Additional note	

Table 51: Development of OML modules for access points

Requirement ID	ER10
Name	Development of OML modules for access points
Description	OML depends on the development of client modules to collect and store the proper measures of the access points.
Type	Functional
Related use case	Tech. Pilot #1
Additional note	

5.5 Operational Categories

In the previous section, we analysed the technology pilots' requirements and categorised them into functional and non-functional requirement types. In this section, we present the operational categories and map the requirement type to the corresponding operational categories.

Non-Virtualized Resource Access

Although it is an assumption that most of the physical resources that will be used in the FIBRE facility will be sliced/partitioned resource, there shall be some requirements from the user to have a non-virtualized resource access. We refer to non-virtualized resources as physical resources which can be virtualized (i.e.) servers and physical resources that cannot be virtualized (i.e. FPGA boards).

Monitoring Subsystem

The FIBRE facility shall be provided with the monitoring subsystem, which will allow monitoring of the hardware switches, servers, CPU, load on a server, links connecting the switches of the island as a whole. The monitoring subsystem will also allow slice monitoring by which we mean slice traffic monitoring, traffic generated by Virtual Machines etc.

Isolation Enforcement

Every slice should be isolated from each other to prevent causing any interference between the slices. Traffic isolation within slice should be implemented and controlled. The administrators shall have full access to terminate any misbehaving slices.

Island Operations

It is very essential to maintain the interconnectivity between different control entities for island and slice operations. The links connecting different island shall be monitored and any fault or failure in the link shall be reported by the notification subsystem to the island administrator. Any maintenance on the island shall be planned in advance and notified to the users.

Facility user service

Users shall be provided with an easy to use GUI to access the FIBRE resources, reserve them as part of their slice and use them in the experiments. They shall also be provided with a console based or XML-RPC application based control for setting up and managing the slice.

Slicing Functionality

Slices shall be provided to users based on the flowspace available at the time of request from the user. The created slices should be in tact or restored quickly during the outage of the slicing entity. Slicing entity by which we mean FlowVisor. FVmanagement shall be restricted to the island administrators although the users shall be enabled to run FlowVisor within their reserved slice.

VM management

In VM management, users shall have the access to VM using a secure shell. Users shall have privileges to create, read, update or delete a VM, add or delete a VM to a slice, manage the user credentials in the VMs.

5.6 Requirement mapping to Operational Categories

In this table, we summarize the requirement mapping to the operational categories resulting from the use case analysis.

Table 52: Requirement mapping to Operational Categories

Requirement/ Operational Category	**Heterogeneous Physical Resources Requirements (PR)**	**Virtualization Requirements (VR)**	**Slice control and Management (Admin)Requirements (AR)**	**Experiment Control and Management (Experimenter) requirements (ER)**
Non-Virtualized Resource Access	**PR03** - Content Delivery Servers **PR04** – WiFi/ Bluetooth/ WiMax access points **PR05** – Controlled mobile devices on model trains **PR17** - Railway track **PR19** - Location of access points **PR20** - Hardware for model train speed control			

	PR21 - Deployment of wireless interfaces in the model train **PR22** - Hardware for model train battery recharging **PR26-** Speed control firmware **PR27-** Battery recharger manager			
Monitoring subsystem	**PR18** - Measurement points		**AR01** - L2 slice creation, allocation, monitoring and release **AR02** - L1 slice creation, allocation, monitoring and release **AR03** - Application server and traffic source/sink monitoring, allocation and release **AR04** - Island overall L2 resources/network monitoring **AR05** - Island overall L1 resources/network monitoring	**ER01** – Experiment Traffic monitoring **ER02** – Experiment user-to-server monitoring **ER03** - Experiment network node monitoring **ER04** - Experiment End node monitoring **ER10** – Development of OML modules for access points

			AR06 - Island overall server resources monitoring **AR07** - Monitoring of WAN links	
Isolation enforcement		**VR03** - Virtualization mechanism for traffic isolation		
Island Operations			**AR04** - Island overall L2 resources/network monitoring **AR05** - Island overall L1 resources/network monitoring **AR07** - Monitoring of WAN links **AR08** - Authentication, Authorization and Accounting	**ER06**- Connectivity between federated Islands **ER07** - Integration of the model train and OMF **ER08** - Integration of the Bluetooth interfaces and OMF **ER09**- Integration of the WiMax interfaces and OMF **ER10** – Development of OML modules for access points
Facility user service	**PR06** – Experimental storage space **PR07** - High Definition media streaming			**ER07** - Connectivity between federated Islands

	server/client **PR08** – GMPLS controller **PR09** - High bandwidth application **PR10** - OpenFlow controller **PR11** - Monitoring Manager **PR12** – Flow-aware Path Computation Element network application **PR13** – Topology Manager **PR14** – Provisioning Manager **PR15** - Load Balancing application **PR16** – Replanning Manager **PR24**- Horizontal handoff service			

	PR25- Vertical handoff service			
Slicing functionality		**VR01** - L2 network slice **VR02** - L1 network slice	**AR01** - L2 slice creation, allocation, monitoring and release **AR02** - L1 slice creation, allocation, monitoring and release	
VM management				**ER05** - SSH support for External users

6 Building blocks

This section maps the requirements discussed in section 5 to the required building blocks for FIBRE Technology Pilots and it is split in different subcategories. The necessary use case building blocks can be categorised into 4 main sections:

- **Users:** experimenter's perspective of the use case. These define the various steps that the user should follow to use the facility to its maximum potential.
- **OpenFlow& Non-OpenFlow physical resources:** the use of OpenFlow (OF) is the key to the use case but some entities e.g. media servers, wifi devices etc. are not OpenFlow enabled. But the FIBRE facility should/will allow the access of both OF & non-OF resources for experimentation.
- **Software:** The experimenter should have access to various tools to carry out a successfully experimentation. These tools form a core part of the control framework and will be made available to users via programmatic interfaces. Though each experiment would require different set of tools, the FIBRE facility will provide a common set enabling easy use of the federated control framework. E.g.: monitoring, control applications etc.
- **Connectivity:** physical connectivity between islands plays an important role in large scale federation facility. Parameters such as bandwidth, QoS etc within and between islands define the scale of the experiments. The connectivity's can be divided into 2 divisions i.e. control & experimental (/data) networks. Control network provides the user with a network through which control messages to the resources can be peered. This interface can be released via low bandwidth links. Experimental is the data path network for experiments and is typically sliced among various users depending on the slice definition.

An important characteristic of the use case implementation is to identify the enhancements that are necessary on the FIBRE control framework and also to the OpenFlow based control environment (SDN controller). So each technology pilot building blocks categorise the various software elements and tool that are necessary to run the use case on the FIBRE facility. Based on these categories each use case will define the various requirements and the essential building blocks.

6.1 Technology Pilot 1

Technology Pilot 1 consists of five main phases: planning and pre-installation, physical installation, logical installation, infrastructure validation, and demonstration development. Below, we map the main building blocks of each phase into the specified requirements.

6.1.1 Building Blocks description:

The main building blocks required for the deployment of the Technology Pilot #1 are described below in Figure 5.

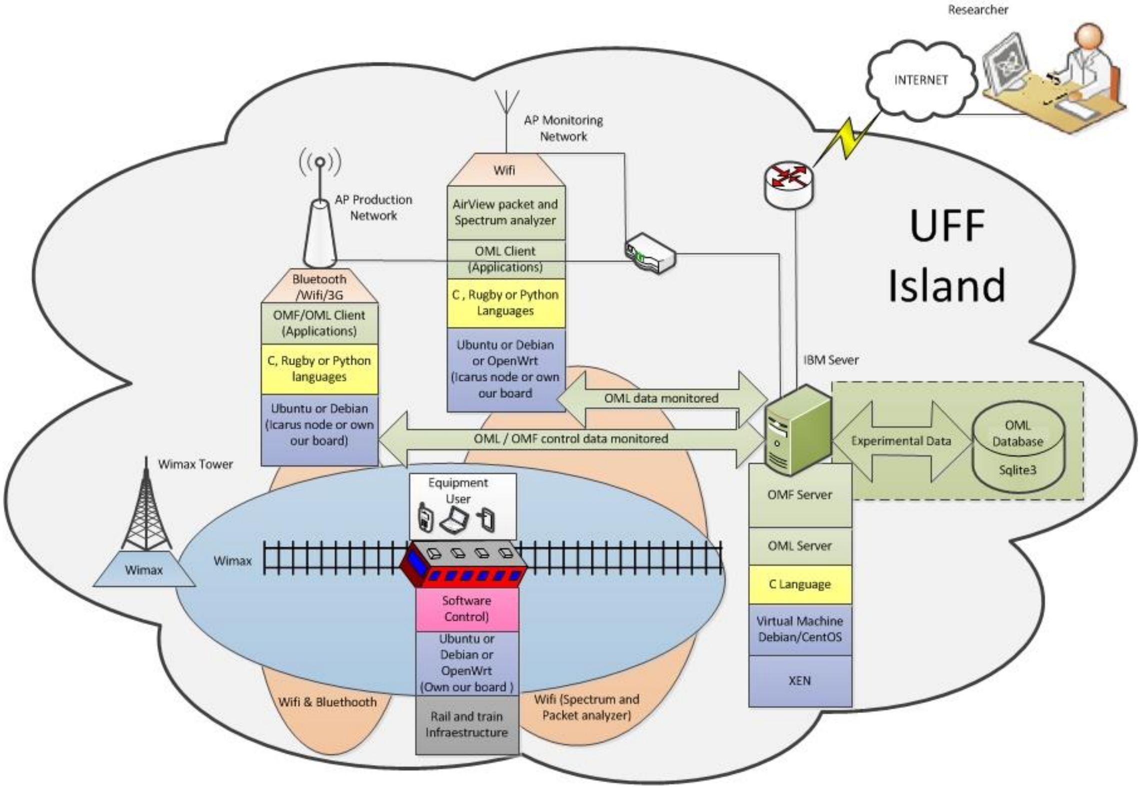

Figure 5: Building blocks of Pilot 1.

Pilot 1 implements a facility to analyze and utilize the capabilities of wireless networks to augment the seamless handoffs experience on networks formed by mobile users (people using smart phones, tablets, netbooks, and notebooks). These experiments include subjects such as vertical and horizontal handoff, wireless monitoring, and spectrum sharing models. Features such as load balancing, reliability, and QoS will be considered in these experiments. To achieve these objectives, we will implement all the phases/building blocks described below.

We split Pilot 1 building blocks into software building blocks and hardware building blocks. The hardware building blocks are:

1 –Wireless node design
2 – Wireless node building
3 – Node positioning design
4 – Icarus node installation
5 – Developed node installation
6 – Monitoring node installation
7 – Train hardware design
8 – Train hardware prototyping
9 – Train hardware building
10 – Train hardware installation

Currently, we are deploying the planning, pre-installation, physical installation and logical installation phases. In these phases, we are developing the building blocks related to the plan and installation of the wireless network. To begin the process we have done an infrastructure study to check the conditions to perform seamless handoffs. Based on measurements, we created an architecture in which we guarantee that a mobile user will experience handoffs, as described in Figure 6.

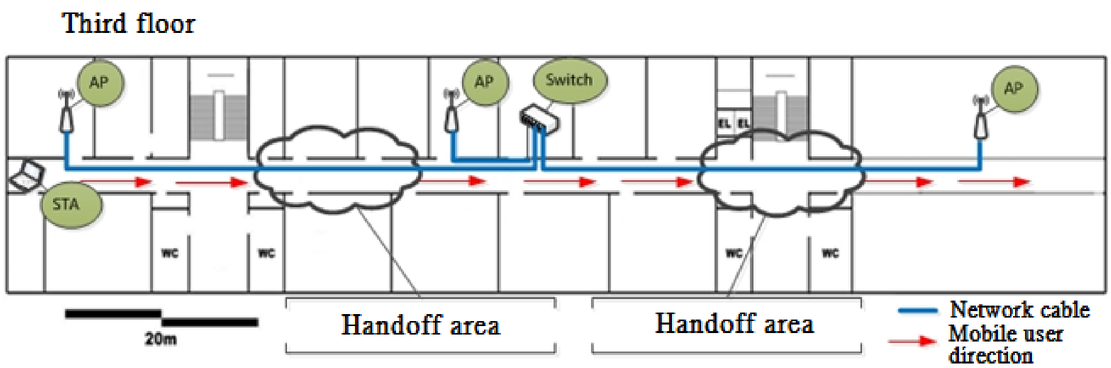

Figure 6: Node distribution in a single floor to perform seamless handoff.

We also mapped possible handoffs between Access Points located at different floors in the same building, in order to provide various scenarios for the demonstration using the "model train" mobile node. The results for this analysis are on Figure 7.

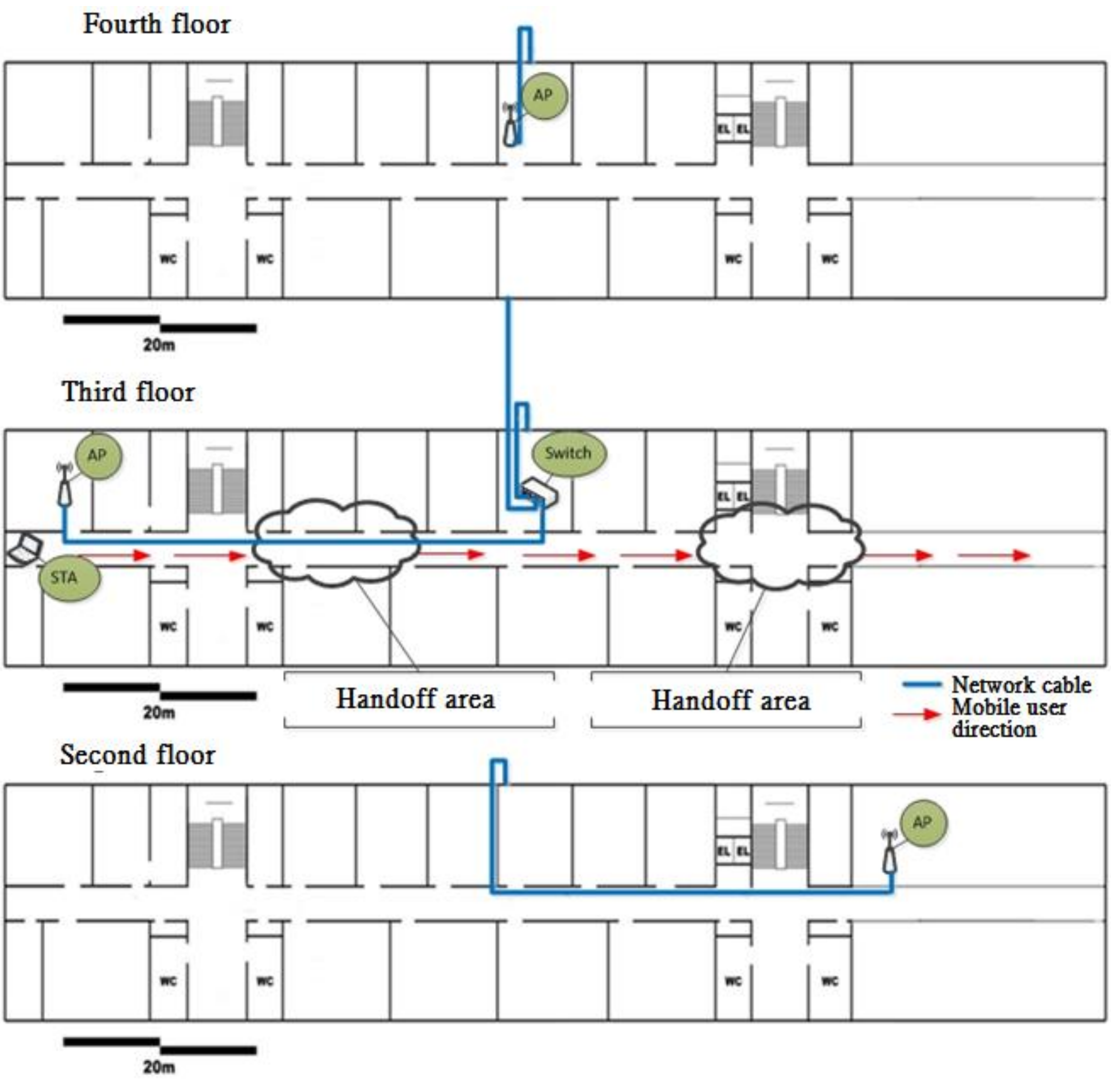

Figure 7: Topology to verify handoff between three Access Point in different stages.

These tests were performed using WireShark [14] and a small packet analyser antenna (airpcap). We used several APs with OpenWrt software and various clients (user devices) such as smartphones, tablets and mobile PCs (notebooks and netbooks).

The conclusions that were reached after the several tests are:

- It will be possible to cause Handoffs by moving the model train in our facility (the movement will take the mobile outside the range of normal WiFi APs).
- Handoffs are dependent on factors such as power level in the AP, user equipment selection, and network topology.
- Some mobile devices were unable to perform seamless handoffs due to technological constraints (in particular, the algorithms implemented in their firmware).

Another building blocking of the planning phase was to specify the coverage area of the facility. Hence, we measured the coverage map using different topologies to choose the best topology for the AP WiFi network and the AP monitoring network.

After some test, as described in Figure 8 and in Figure 9, we concluded that three WiFi-enabled programmable APs per floor are enough to perform a large range of tests. We also measured the monitoring requirements. Based on this data, we will deploy two more APs per floor to create the dedicated monitoring network.

Figure 8: Coverage map of a single floor using a wifi AP placed inside a classroom.

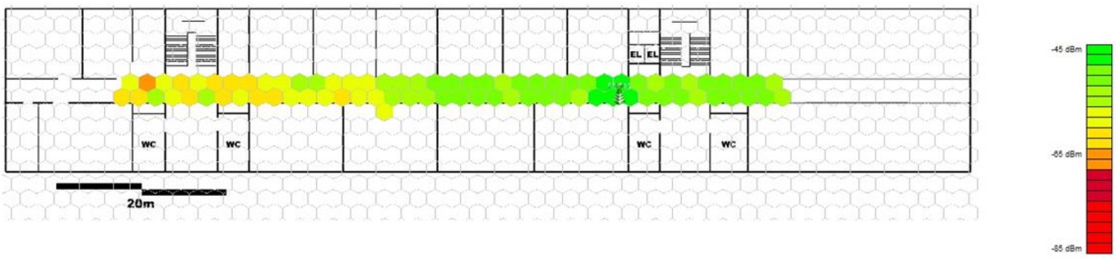

Figure 9: Coverage map of a single floor using a wifi AP placed in the corridor.

Another important building block of Pilot 1 is the deployment of the model train. This device represents a controlled mobile user in the facility. This building block can be subdivided in two sub-blocks, corresponding to the mechanical aspects and the control software.

We designed this mobile node using part of the existing facility on the building (the trellis framework that can be used to hang the equipment from). The mobile node consists of a structure connected to a controlled motor equipped with different wireless devices. We designed an interface that allows the experimenters to choose the speed and the time period that the train should be moving. The train movement is also tracked by the monitoring system, in order to provide a detailed scenario description to the experimenter. By now, the track-rail structure design is ready and is being implemented (physical structure and control software).

The installation of the wireless nodes has begun. We are placing the recently received Icarus nodes in the building according to the architecture that resulted from the measurements described above. We are also developing low-cost wireless nodes to increase the number of available nodes to the experimenter, supplementing the received Icarus nodes, in order to allow experimentation in high density scenarios.

Another building block for the demo is the installation and test of the OMF in our facility, which is under development on the scope of WP2. In this building block, we are installing and testing OMF and OML in the mobile node and APs.

After completing the installation of the wifi-nodes, we will move to next building block, which is the planning and installing of the other wireless equipment on the nodes distributed over our test scenario. This building block requires not only the installation, but the development of the interfaces to control the devices through OMF.
We also develop OMF and OML test scripts, to carry out several scenarios for testing seamless handoffs.
After the development of the hardware building blocks, we begin the development of the software building blocks, described in Figure 10.

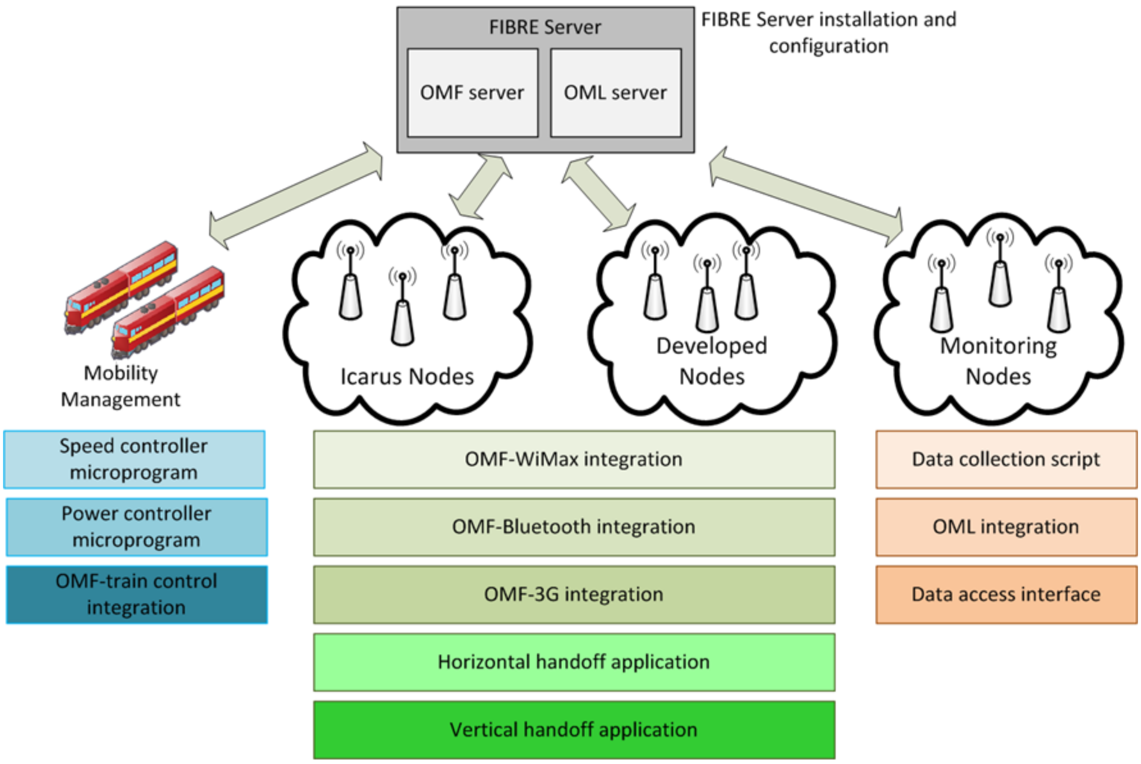

Figure 10: Software building blocks of Pilot 1.

All the interfaces and internal mechanisms of the software building blocks were defined, but the development of the software depends on the deployment of the physical infrastructure. Hence, the software building blocks are the next steps of Pilot 1.

6.1.2 Requirements for building blocks:

6.1.2.1 Users

This section describes the building blocks needed to satisfy user requirements for the Technology Pilot 1.

Requirements	
ER05, ER06	Access points, model train, virtual machines, and monitoring servers
VR02, VR03	Reserve resources via control framework
AR02, AR04, AR05, AR06, ER01, ER03, ER04, PR18	Monitoring the network environment as well as the experiment parameters

6.1.2.2 Non-OpenFlow resources

This sub-section includes the non-OpenFlow resources needed to satisfy user requirements for the Technology Pilot 1.

6.1.2.3 Wireless

Requirement ID	Description
PR04, PR19, AR09, AR10, AR11	WiFi/Bluetooth/WiMax access points with support to OMF/OML
VR02	L1 network slice
VR03	Virtualization mechanism for traffic isolation

6.1.2.4 Other equipment (e.g. Router, VPN etc.)

Requirement ID	Description
PR05, PR17, PR20, PR21, PR22, PR25, PR26, ER12	Controlled mobile devices on model trains, which demands hardware and software support for controlling node mobility
PR06, ER11, ER12	Experimental storage space, used for storing monitoring data from the experiments

6.1.2.5 Software

Requirement	
AR02, AR04, AR05	Reserve & access non-OF and virtual resources through control framework
PR04, ER09, ER10	Develop the OMF modules for controlling all the access point interfaces
PR05, ER08	Develop the OMF module for interact with the model train control
PR06, PR18, ER11, ER12	Monitoring modules for the wireless medium distributed in the network and the OML control module

6.1.2.6 Controllers

Requirement	
PR25, PR26	Remote control of model train hardware
ER08 – ER10	Interface of testbed with OMF

6.1.2.7 Applications

Requirement	
PR15	Development of a load balancing control application
PR23	Development of a horizontal handoff control application
PR24	Development of a vertical handoff control application

6.1.2.8 Monitoring

Requirement ID	Description
ER01	Experiment traffic monitoring
ER03	Experiment network node monitoring
ER04	Experiment End node monitoring
AR02	L1 slice creation, allocation, monitoring and release
AR04	Island overall L2 resources/network monitoring
AR05	Island overall L1 resources/network monitoring
AR06	Island overall server resources monitoring
ER11, ER12	Collect and store measures

6.1.2.9 User management

Requirement ID	Description
ER05	SSH support for External users
ER06	Authentication, Authorization and Accounting

6.1.2.10 Other

Requirement ID	Description
VR03	Virtualization mechanism for traffic isolation in the virtual machine server

6.1.2.11 Connectivity

6.1.2.12 Control

Requirement ID	Description
ER05	SSH support for External users
ER06	Authentication, Authorization and Accounting

6.1.2.13 Experimental

Requirement ID	Description
ER01, ER03, ER04	Design experiment monitoring
VR02, VR03	Experiment allocation and usage

NOTE: Use Case 1 was heavily modified which has been explained in the first FIBRE review and there has been a delay in this activity. Hence the development details are not fully described in the document.

6.2 Technology Pilot 2

6.2.1 Building Blocks description:

This use cases focuses on steering media clients to appropriate content servers based on various network & server load characteristics. In order to implement the use case on the FIBRE control framework a set of enhancements to the control framework and also to the control applications on SDN based OpenFlow controller (NOX) have been identified. This section details these enhancements in details

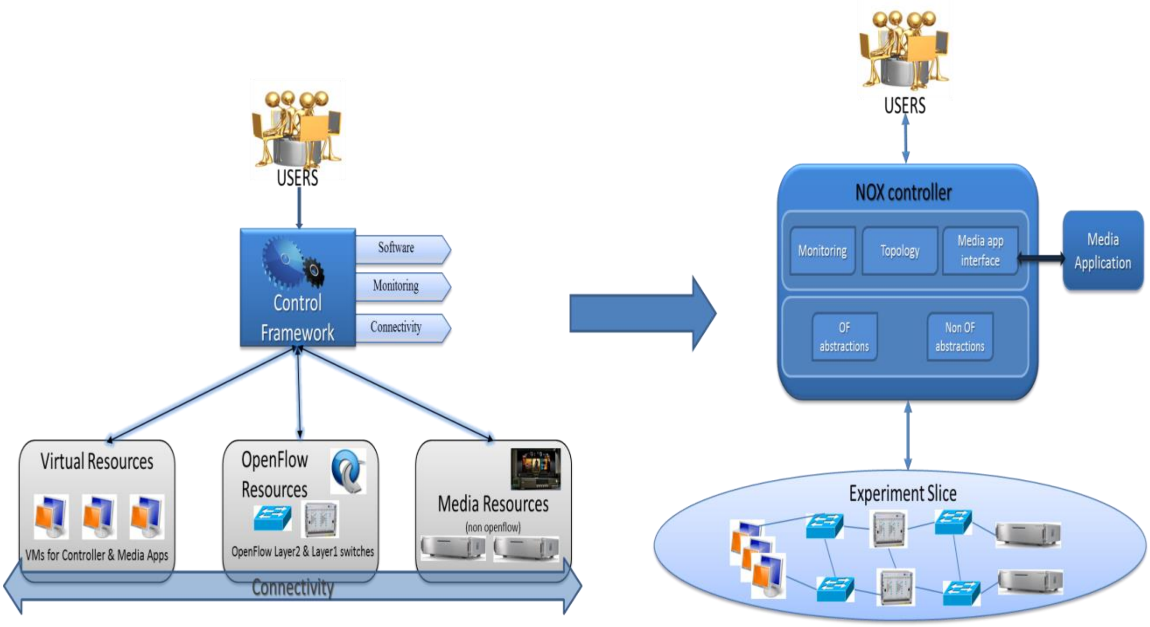

Figure 10: Technology Pilot 2 requirements mapping to building blocks

Experiment slice: The respective island control framework will provide a virtual slice for the use case. The slice will consist of media resources, OpenFlow ports on both L2 packet & L1 optical switches. The slice follows the flowspace that the user provides and is granted by the island admin. Furthermore the slice will consist of link between islands in EU & Brazil.

OF & Non-OF Abstraction on controller: The media use case delivers high bandwidth media streams to a large number of media clients which require high capacity low latency optical network. Hence the slice of the network resource will consist of not only packet port attributes but also optical ports. The OF abstraction for the packet follows the OF v1.0 specification defined by ONF [1], whereas the optical circuit switching follows the OF circuit addendum v0.3 and both are based on the flow switching concept with different flow attributes e.g. tcp/udp ports, vlan, wavelengths, timeslots etc. These abstractions are made available via the open source C++ OpenFlow controller called NOX [3], which supports both packet & optical abstractions.

The media server is a separate non-OF entity but reservable via the control framework. Once the media streaming server is reserved by the experiment slice it can be registered via a media broker using a web service API.

Media Broker: The media broker application available on the NOX controller will expose a web service based publish & subscribe interface through which the media servers are registered. Any participating server will publish its media and network attributes to the broker via the interface. An example of the attributes can be found in the appendix section 1.1. Once the media server location and its attributes (e.g. HD, 2k, 4K encoding, GPU, memory etc.) are subscribed by the media broker then any media client requesting streaming access can be serviced by the broker using the subscribed information. This subscribed media server information is also read by the topology app to build media aware networks.

Another important functionality of the broker is to service media client requests. It receives web service requests from media clients for specific content attributes and the broker resolves the appropriate media server. Once the destination is chosen then the path to the destination from the client is resolved by the media flow PCE which is configured by the NOX controller.

The media broker is also responsible to gauge the congestion/load statistics and make a decision to trigger redirection. Depending upon the monitor stats received from the monitoring application the broker will trigger copy process of the cache content and then delegate the new path insertion to the NOX controller.

Topology: Using the OpenFlow & the web service interface the topology application can collect the information of the whole media network including OF & media resources. The topology application is basically a database which stores information retrieved by the controller using the OF & media web service protocol. The DB maintains the node & link states of involved resources providing an update topology to the path computation application. In particular, Topology DB is composed by the following tables which hold following information:

- datapaths including information about all OF switches connected to NOX
- ports including information for all OF ports
- cports_bandwidth including the information regarding the ports of the connected circuit switches

- hosts including information for all attached and authenticated hosts
- links including all inter-switch links information
- flow_entries including detailed description for the active flow entries of all controlled OF switches
- port_stats including statistics (e.g. number of bytes received) for each OF port
- table_stats including description for the active tables of each OF switch connected to NOX
- Media server information: GPU, CPU, media encoding type, bandwidth supported

Media Flow PCE: The media flow Path Computation Engine (PCE) has necessary algorithms to compute the appropriate path from the media client requester to the server destination. Upon request from the controller for path computation, from client source to server destination, the PCE computes the best and backup paths for the request. The controller then pushes the appropriate flows via the OF & web service interfaces to realise the computed path. The algorithms consists of necessary path computations solutions for the integrated packet & circuit networks.

Monitoring: one of the key goals of the use case is to monitor the link & media server measurement points i.e. GPU, CPU, OF port & flow stats etc. The control framework provides an enhanced monitoring framework which the experimenter can use to collect statistics related to the required resources. Apart from this the OpenFlow protocol itself provides port and flow stats which can be used to verify the load thresholds and then trigger clients redirection.

6.2.2 Requirements for building blocks:

6.2.2.1 Users

This section describes the building blocks needed to satisfy user requirements for the Technology Pilot 2.

Requirements	
ER05, ER06	Accessing Media clients & content delivery servers

VR01, VR02, PR03	Reserve OF, non-OF & virtualized resources via control framework
Monitoring subsystem operation category covering requirements **AR01-04, ER01-04**	Monitoring Media Clients parameters (latency, link utilization etc.)

6.2.2.2 OpenFlow resources

OpenFlow extensions are needed for operation of the Technology Pilot 2.

Packet

Requirement ID	Description
PR01	OpenFlow L2 infrastructure
VR01	L2 network slice
VR03	Virtualization mechanism for traffic isolation

Optical

Requirement ID	Description
PR02	OpenFlow-enabled ROADMs
VR02	L1 network slice
VR03	Virtualization mechanism for traffic isolation

6.2.2.3 Non-OpenFlow resources

This sub-section includes the non-OpenFlow resources needed to satisfy user requirements for the Technology Pilot 2.

Media servers

Requirement ID	Description
PR03	Content Delivery Servers
PR07	High Definition media streaming server/client

Other equipment (e.g. Router, VPN etc.)

Requirement ID	Description
PR06	Experimental storage space

6.2.2.4 Virtual Resources

Requirements	
VR01	Running Media applications
Monitoring subsystem operation category covering requirements **AR01-04, ER01-04**	Monitoring VM status (CPU, memory etc.)

Software

Requirement	
AR01-04, ER05-06	reserve & access OF, non-OF and virtual resources through control framework

Controllers

Requirement ID	Description
PR10	OpenFlow controller
VR01, ER06	Running Media Applications on controller
PR01-02,	Abstractions for optical, media resources (e.g. OpenFlow for optical)
PR03, PR07	Users able to use non-OF interfaces & parameters. E.g. content delivery servers

Applications

Requirement ID	Description
PR11	Monitoring Manager (capable to monitor CDS)
PR12	Flow-aware Path Computation Element network application

PR13	Topology Manager
PR14	Provisioning Manager
PR15	Load Balancing between domains
PR15	Migrate users to new CDS based on trigger thresholds
PR16	Replanning Manager

Monitoring

Requirement ID	Description
ER01	Experiment traffic monitoring
ER02	Experiment user-to-server monitoring
ER03	Experiment network node monitoring
ER04	Experiment End node monitoring
AR01	L2 slice creation, allocation, monitoring and release
AR02	L1 slice creation, allocation, monitoring and release
AR03	Application server and traffic source/sink monitoring, allocation and release
AR04	Island overall L2 resources/network monitoring
AR05	Island overall L1 resources/network monitoring
AR06	Island overall server resources monitoring
AR07	Monitoring of WAN links
ER02	Retrieve server parameters (e.g. CPU, memory, energy)
ER01	Retrieve link utilization
ER02-04	Monitoring different attributes of the media traffic (client load, latency, etc.). These might involve attributes not available directly from control framework e.g. Flow stats through OpenFlow

User management

Requirement ID	Description
ER05	SSH support for External users
ER06	Authentication, Authorization and Accounting

Media

Requirement ID	Description

PR07	High Definition media streaming server/client
PR09	High bandwidth application

6.2.2.5 Other

Requirement ID	Description
VR03	Virtualization mechanism for traffic isolation

Connectivity

Control

Requirement ID	Description
ER05	SSH support for External users
ER07	Ability to reserve and provision resources from different islands

Experimental

Requirement ID	Description
ER07	Ability to reserve and provision resources from different islands
ER07	Sufficient bandwidth for running media experiments

QoS

The following table shows the requirements in terms of bandwidth for JPEG 2000 compressed streaming of different digital media formats (i.e. HD, 2K and 4K).

Digital Media Format	Bandwidth
HD	~20-60Mbps
2K	~65Mbps
4K	~250Mbps

6.2.3 Use Case Dissection:

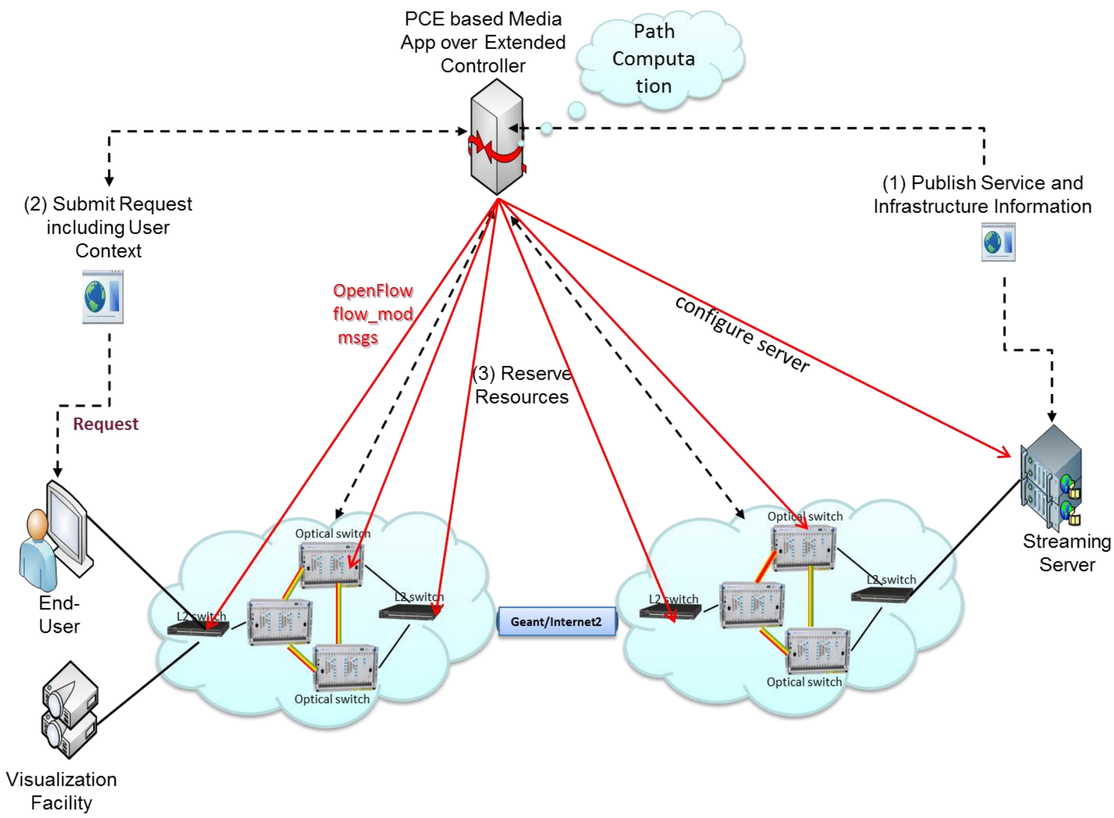

Figure 11 Use case 2 operation

Media Broker with NOX can support control and management modules that discover the underlying network infrastructure. NOX uses OpenFlow modules to obtain status information about the network. The Path Computation Engine (FlowPCE) leverages this status information to determine the existence of network connectivity with sufficient capabilities to ensure high QoS and QoE for media services delivered by the ME. If media and network infrastructure resources exist to provision end-users' requests, the Media Broker Interface coallocates the chosen media and network infrastructure resources. For network setup, the NOX uses a Standard OpenFlow Agent for packet switched domains and an Extended Optical OpenFlow Agent for circuit switched optical domains. The Extended Optical OpenFlow Agent is based on novel extensions to enable OpenFlow control and manage circuit switched optical networks [12]. Specifically, extensions to the OpenFlow switch features request/reply messages to enable OpenFlow discover optical nodes in the data plane, the CFLOW_MOD i.e. flow specification message was implemented for setup of cross-connections in the optical domain, and optical switching constraint considerations were introduced in the OFC. These agents are used to proactively push OpenFlow flow-rules to the network elements in the data plane to switch traffic on high QoS paths selected by the PCE. To reserve media infrastructure resources, media broker uses appropriate Media Resource Agents.

6.2.3.1 Media broker:

Components:

- Media server subscriber
- Load analyser

Interface

- Web service interface

Example wsdl script in Appendix A

6.2.3.2 OpenFlow Controller:

Components: NOX controller

Interfaces: web service interface

Example wsdl script in Appendix A

6.2.3.3 Topology:

Components: Mysql Database

Interfaces: XML-RPC interface

6.2.3.4 Media Flow PCE:

Components: Algorithms for path computation

Interfaces: XML-RPC interface

6.2.3.5 Monitoring:

Component: NOX monitoring application

6.2.4 Use case flow:

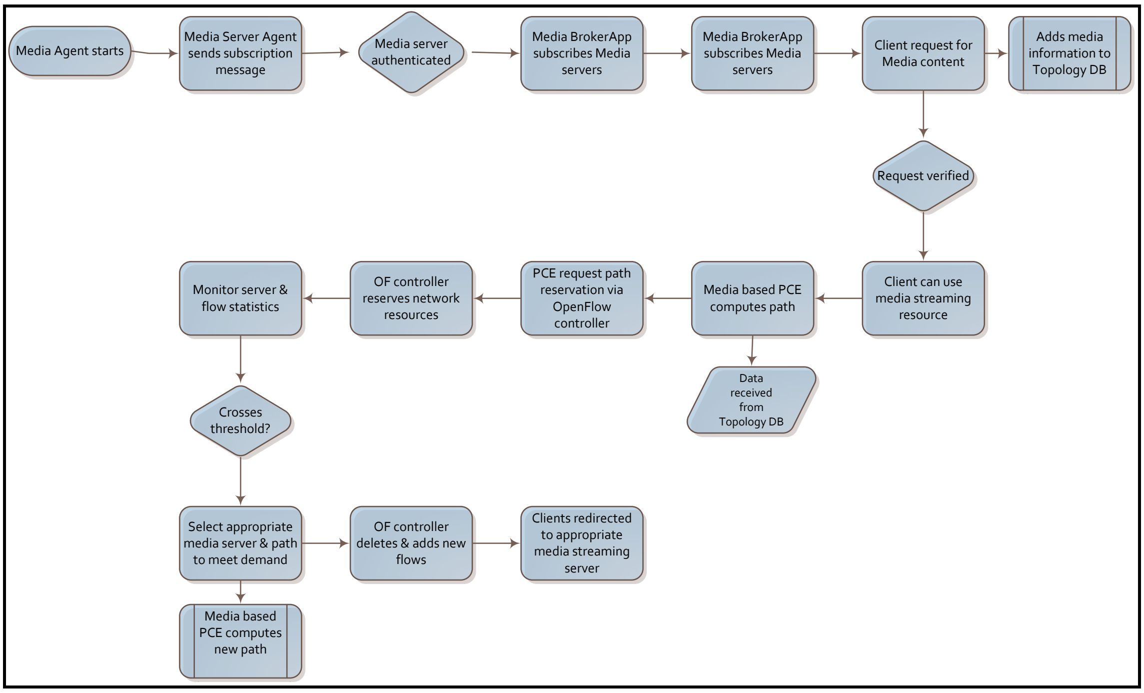

Figure 12 Use case 2 flow of events

The operation of the use case is as follows:

1. Media Broker process end-user's media request; determine requirements and end-user's context.
2. Execute semantic queries to identify appropriate media services and resources.
3. If media services and resources are available, compute high QoS path using Flow PCE.
4. If media service and resources are unavailable, reject request; go to (7).
5. If high QoS path is available, dynamically create and push OpenFlow flow-rules to elements in data plane and configure media resources to deliver service; go to (7).
6. If path without sufficient QoS (bandwidth) is unavailable, reject request; go to (7).
7. Notify end-user of acceptance or rejection of request.

6.3 Technology Pilot 3

6.3.1 Building Blocks description:

In the Tech-Pilot 3, we have chosen to adopt the GMPLS PCE approach instead of using the entire GMPLS stack.

The main building blocks required for the deployment of the Technology Pilot #3 are depicted in the following figure.

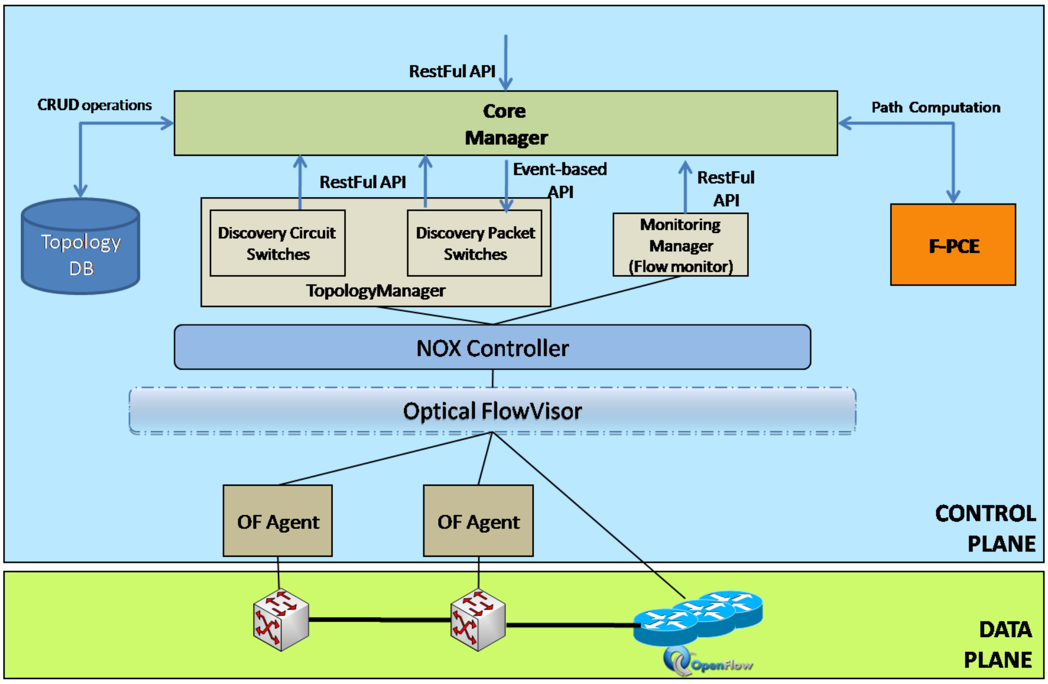

Figure 13 GMPLS BoD overlay on OpenFlow

As depicted in the Figure. 13, Data Plane includes both OF native devices and circuit switches. Control Plane is focused on the functionalities provided by the applications running on top of the NOX controller. It is worth underlying the key role covered by the OF Agents that provide a translation between extended OF messages (ref. [6]) and the SNMP commands to be sent to the circuit devices. Hence, the NOX controller has a full control of the overall underlying topology (i.e. both packet and circuit domains) by sending and receiving only OF messages.

The main functionalities of each depicted building block are described in details in the following sections.

6.3.1.1 Core-manager

Core Manager is a NOX component (written in Python) acting as web-server application. It is responsible for updating the topology information of GMPLS-PCE data model and Topology-DB tables. A detailed description of the allowed CRUD (Create, Read, Update and Delete) operations and corresponding URI resources is referred in the 6.3.3.1 paragraph.

In particular, Core Manager interacts via CORBA (Common Object Request Broker Architecture) with the GMPLS-PCE for updating internal network graph with nodes (vertexes) and links (edges) discovered by Discovery Packet Switches application. The aforementioned CORBA topology interface is described in 6.3.3.4 paragraph.

Furthermore, Core Manager is in charge for filling and updating the Topology-DB tables with the values received by Flow-Monitor application such as statistics of each active flow entry and/or OF port statistics (see B.2 for details about the Topology DB schema).

Moreover, Core Manager can control the discovery-packet application sending (NOX) events in order to create new flow entries including the following parameters:

- Ingress/egress datapath and port values
- Source/destination TCP/UDP port numbers
- Source/destination IP address and protocol number
- VLAN identifier

6.3.1.2 Discovery Packet Switches

Discovery Packet Switches (DPS) is a NOX application (written in Python) responsible for retrieving the whole underlying packet-domain topology including the controlled OF switches as well as their ports and capabilities, inter-switch links (using LLDP packets) and the attached (and authenticated) hosts. DPS uses the Core Manager interface (see 6.3.3.1 for details) in order to perform CRUD operations on the topology resources.

The following table summarizes the mechanisms used to collect all aforementioned information.

Table 53: Discovery functionalities, retrieved information and adopted mechanisms

Topology component	Discovery mechanism adopted	Main collected info
OF Switch	NOX Datapath_join_event raised whenever a new OF switch is connected NOX Datapath_leave_event raised whenever an OF switch is	Datapath ID (DPID) as unique identifier Capabilities Ports and related features

	disconnected (see 6.3.3.2 for more details)	
Links	Send periodically LLDP packets to every switch interface and then use the received LLDP packets to detect inter-switches links	Source DPID and port number Destination DPID and port number
Hosts	NOX Host_auth_event raised whenever a new host is authenticated NOX Host_bind_event raised whenever a new binding MAC/IP address for an authenticated host is detected (see 6.3.3.2 for more details)	DPID and ingress port number of the connected OF switch IP and MAC addresses

It is worth noting that DPS can be easily extended to send path computation request to GMPLS-PCE application (through core-manager) in case that a `packet_in_event` is raised by the NOX controller for an incoming packet that does not match any installed flow entry in the OF switch.

6.3.1.3 Discovery Circuit Switches

Discovery Circuit Switches (DCS) is a NOX application (written in C++) called lightpath app. It follows the openflow circuit switching addendum extensions v0.3 and supports WDM devices hosted in FIBRE facility. The application is used to retrieve the WDM network information which includes device characteristics (e.g. fixed or flexible WDM grid, etc), module details, port details, wavelengths details and switching constraints associated with the node.

The following table summarizes the mechanisms used to collect all aforementioned information.

Table 54: Circuit discovery functionalities, retrieved information and adopted mechanisms

Topology component	Discovery mechanism adopted	Main collected info

OF Switch	NOX `Datapath_join_event` raised whenever a new OF switch is connected NOX `Datapath_leave_event` raised whenever an OF switch is disconnected (see 6.3.3.2 for more details)	Datapath ID (DPID) as unique identifier Capabilities Ports and related features Wavelengths supported
Links	Uses proprietary mechanisms to discover inter node links but is propagated to controller via openflow features messages	Peer dpid and port number Associated wavelengths

It is worth noting that DCS can be easily extended to send path computation request to GMPLS-PCE application in case that a `packet_in_event` is raised by the NOX controller for an incoming packet in the OF switch in the packet domain a corresponding optical domain path (flows) can be established in parallel with the packet flows.

6.3.1.4 Flow-monitor:

Flow-monitor is a NOX application (written in Python) responsible for periodically updating three topology database's tables (i.e. `flow_entries`, `port_stats` and `table_stats` tables) with values extracted from the available OF switches.

Flow-monitor is controlled by three timer values and three internal queues:

- Flow timer and switch-identifier queue
- Port-stats timer and port-identifier queue
- Table-stats timer and switch-identifier queue

When the specific timer (e.g. flow-timer) expires, flow-monitor checks the status of the related identifier queue (e.g. switch-identifier queue). If the queue is empty, it requests the list of the available resources (i.e. switches or ports) to the core-manager application and stores the identifiers into the corresponding queue. Otherwise, it takes the first identifier from the queue and calls the NOX to fetch all the flow entries/port statistics/table statistics registered in the OF switch.

In particular, Flow-monitor is responsible for performing operations on the following Topology DB tables (ref. B.2):

- `port_stats` including the statistics for all ports of the OF switches connected to the NOX controller (e.g. number of bytes received and/or transmitted)
- `table_stats` including the statistics referring to the table of the OF switches (e.g. number of active entries)
- `flow_entries` including the description of each flow entry

6.3.1.5 Topology-DB:

Topology DB maintains topology information received by the Core Manager application including the OF switches with related ports and capabilities, the inter-switch links and the attached hosts. It is extended with the monitoring information received by the Flow Monitor application including OF switches tables and port statistics.

In particular, Topology DB is composed by the following eight tables described in details (i.e. in terms of fields and types) in B.2:

- datapaths including information about all OF switches connected to NOX
- ports including information for all OF ports
- `cports_bandwidth` including the information regarding the ports of the connected circuit switches
- hosts including information for all attached and authenticated hosts
- links including all inter-switch links information
- `flow_entries` including detailed description for the active flow entries of all controlled OF switches
- `port_stats` including statistics (e.g. number of bytes received) for each OF port
- `table_stats` including description for the active tables of each OF switch connected to NOX

6.3.1.6 GMPLS Flow-aware Path Computation Element:

GMPLS Flow-aware Path Computation Element (F-PCE) is a new functional element introduced for FIBRE project in order to provide the routing functionalities required during all lifecycle of the Technology Pilots #2 and #3.

F-PCE is based on the IETF PCE architecture specified in [RFC4655]. It is responsible for the composition of the network services related to the end-to-end flow routes. In particular, F-PCE is the centralized entity to be queried by the NOX routing applications in order to receive the end-to-end path including the involved OF switches in the packet domain (and the related flow entries to be added) and the switching resources to be used in the circuit domains in the transit network.

F-PCE maintains an internal Traffic Engineering Database (TED) populated via CORBA by the Core-Manager application whenever a topology change is detected (e.g. a new OF switch is connected to the NOX and/or a new inter-switch link is detected). In this manner F-PCE can compute the end-to-end path taking into account of the actual network topology information.

It is worth noting that F-PCE supports the hierarchical deployment based on proper iterations between child and parent F-PCE elements. This important feature may be exploited for potential enhancements in the future project activities for the integration of local and federated showcases. In particular, the inter-island path computation may be performed by using multiple F-PCE elements where each Child F-PCE located in an island is charging for the intra-island path computation and queries to a single Parent F-PCE for the inter-island path computation in order to be able to trigger the overall end-to-end inter-island path computation. The aforementioned hierarchical approach is shown in the following figure:

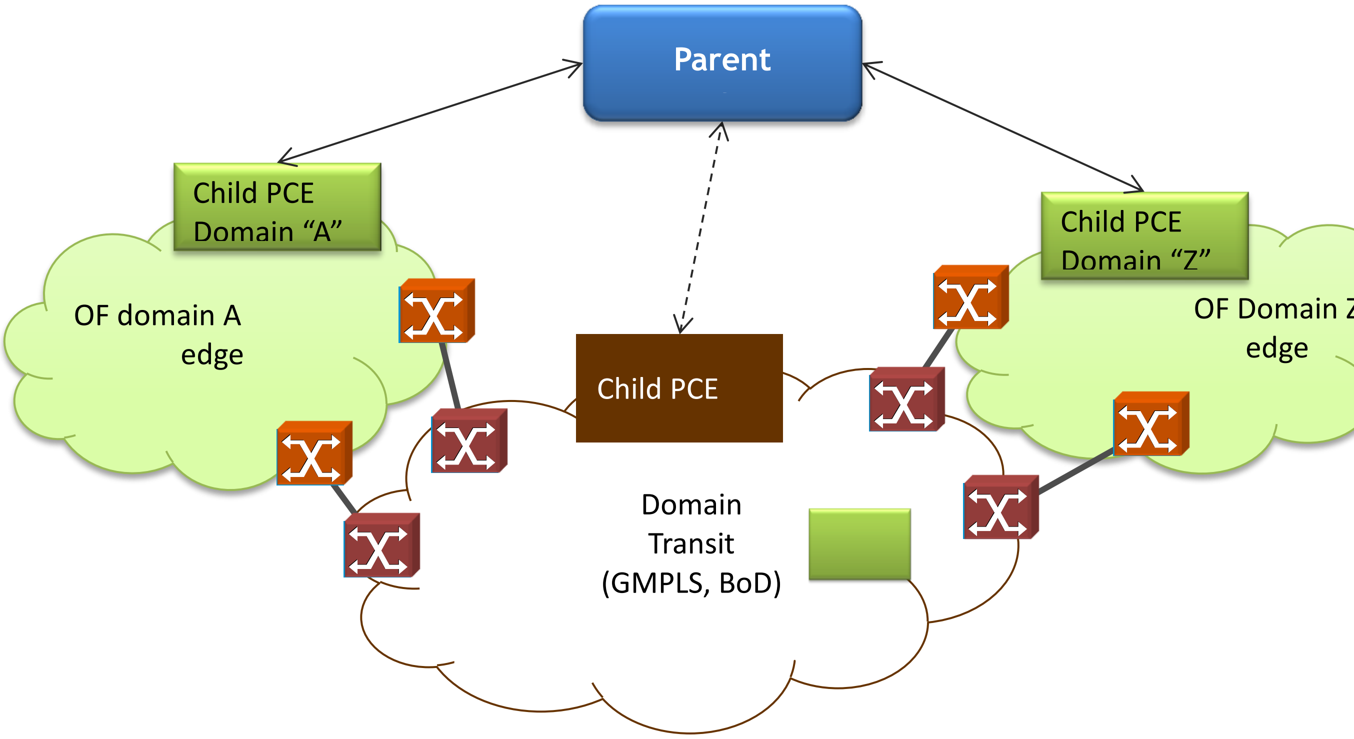

Figure 14 F-PCE potential extension: hierarchical approach for the inter-island path computation

Another potential extension is the possibility to extend the topology information with IT end-resources and to take into account also the monitoring information in the path computation processing.

6.3.2 Requirements for building blocks:

6.3.2.1 Users

This section describes the building blocks needed to satisfy user requirements for the Technology Pilot 3.

Requirements	
VR01	Accessing Media clients (content delivery servers)
PR03	Reserve OF, non-OF & virtualized resources via control framework
Monitoring subsystem operation category covering requirements **AR01-04, ER01-04**	Monitoring Media Clients parameters

6.3.2.2 OpenFlow resources

OpenFlow extensions are needed for operation of the Technology Pilot 3.

6.3.2.3 Packet

Requirement ID	Description
PR01	OpenFlow L2 infrastructure
VR01	L2 network slice
VR03	Virtualization mechanism for traffic isolation

6.3.2.4 Optical

Requirement ID	Description
PR02	OpenFlow-enabled ROADMs
VR02	L1 network slice
VR03	Virtualization mechanism for traffic isolation

6.3.2.5 Non-OpenFlow resources

6.3.2.6 Media servers

Requirement ID	Description
PR07	High Definition media streaming server/client

6.3.2.7 Other equipment (e.g. Router, VPN etc.)

Requirement ID	Description
PR06	Experimental storage space

6.3.2.8 Virtual Resources

Requirements	
VR01	Running Media applications

Monitoring subsystem operation category covering requirements **AR01-04, ER01-04**	Monitoring VM status (CPU, memory etc.)

6.3.2.9 Software

Requirements	
VR01, AR01-04	Control Framework to support reserving OF, non-Of and virtual resources
Monitoring subsystem operation category covering requirements **AR01-04, ER01-04**	Monitoring different attributes of the media traffic (client load, latency, etc.)

6.3.2.10 Controllers

Requirement ID	Description
PR08	GMPLS controller
PR10	OpenFlow controller

Requirements	
VR01	Running Media Applications on controller
Monitoring subsystem operation category covering requirements AR01-04, ER01-04	Abstractions for optical, media resources (e.g. OpenFlow for optical)
PR03	Users able to use media encoders and other sources
PR13	Retrieve topology from GMPLS and OpenFlow regions
PR12, PR14	Compute and Provisioning of the lightpath across multi regions
	Monitor resource status

6.3.2.11 Applications

Requirement ID	Description
PR11	Monitoring Manager
PR12	Flow-aware Path Computation Element network application

PR13	Topology Manager
PR14	Provisioning Manager

6.3.2.12 Monitoring

Requirement ID	Description
ER01	Experiment traffic monitoring
ER03	Experiment network node monitoring
ER04	Experiment End node monitoring
AR01	L2 slice creation, allocation, monitoring and release
AR02	L1 slice creation, allocation, monitoring and release
AR03	Application server and traffic source/sink monitoring, allocation and release
AR04	Island overall L2 resources/network monitoring
AR05	Island overall L1 resources/network monitoring
AR06	Island overall server resources monitoring
AR07	Monitoring of WAN links

6.3.2.13 User management

Requirement ID	Description
ER05	SSH support for External users
ER06	Authentication, Authorization and Accounting

6.3.2.14 Media

Requirement ID	Description
PR07	High Definition media streaming server/client
PR09	High bandwidth application

6.3.2.15 Other

Requirement ID	Description
VR03	Virtualization mechanism for traffic isolation

6.3.2.16 Connectivity

6.3.2.17 Control

Requirement ID	Description
ER05	SSH support for External users

6.3.2.18 QoS

The requirements in terms of bandwidth for both local and federated BoD resource provisioning is from 10Mbps to 10Gbps.

6.3.3 Use Case Dissection:

6.3.3.1 Core-manager interface

Core-manager is a Python NOX application developing a web-server interface. Its main functionality is the communication with the GMPLS-PCE application and Topology-DB module.

Core-manager develops two (logical) interfaces:

- northbound interface
- southbound interface

Core-manager uses proper events to communicate with the discovery-packet application. More details in Appendix B.3: interfaces section.

Events interface

The interfacing between multiple running NOX applications is guaranteed by usage of the events posting mechanism. In this manner the other NOX applications can be notified and receive related information regarding the event.

An example of how posting the Flow-Entry event is described in the appendix code:

```
cflows = convert_flows_from_index(flows)
for d_in, p_in, d_out, p_out in cflows:
     evt_ = nxw_utils.Pckt_flowEntryEvent(dp_in=d_in,
                                           port_in=p_in,
                                           dp_out=d_out,
```

```
                                        port_out=p_out,
                                        ip_src=ip_src_,
                                        ip_dst=ip_dst_,
                                        tcp_dport=dst_port_,
                                        tcp_sport=src_port_,
                                        ip_proto=ip_proto_,
                                        vid=vlan_id_,
                                        etype=default_etype,
                                        action=default_action,
                                        idle=default_idle,
                                        hard=default_hard,

prio=default_priority)
      WLOG.debug(str(evt_))
      PROXY_POST(evt_.describe())
```

6.3.3.2 Discovery-packet management events

Discovery Packet Switches, located in the src/nox/coreapps/examples folder, is a NOX Python application depending on two native NOX applications, i.e. discovery and pyauthenticator.

DPS is an event-driven software application. It is written to respond to the events raised by depending applications for the topology discovery phase and core-manager application for the provisioning phase. In particular, the following table summarises the managed events and related callbacks:

Table 55 packet discovery events

Registered event	Event Description	Handler	Handler description
datapath_join	Raised whenever a new OF switch is detected on the network	datapath_join_handler	Trigger (through core manager) datapath insertion in the topology-DB and in GMPLS F-PCE
datapath_leave	Raised whenever an OF switch has left the network	datapath_leave_handler	Trigger (through core manager) datapath deletion in the topology-DB and in GMPLS F-PCE

packet_in	Raised whenever a new packet (that does not match any flow entry) is received by NOX	packet_in_handler	Trigger (through core manager) path computation request sending towards GMPLS F-PCE
Link_event	Raised for each link change detected in the network	link_event_handler	Trigger (through core manager) link insertion/deletion in the topology-DB and in GMPLS F-PCE data-model
Host_auth_event	Raised for each new host authenticated	host_auth_handler	Add host information into Topology-DB and in GMPLS F-PCE data model
Host_bind_event	Raised whenever a new binding MAC/IP address for an authenticated host is detected	host_bind_handler	Update host information with received addresses into Topology-DB and in GMPLS F-PCE data model
Pckt_flowEntryEvent	Raised whenever a flow entry is created/updated/deleted in the OF switch	flow_entry_handler	Update flow entries into Topology DB

6.3.3.3 Flows-monitor management events

FlowMonitor is a Python NOX application that depends on the core-manager application and flow-fetcher (NOX) module.

The list of functions imported from *nox.lib.core* is described in the following table:

Function	Description
post_callback(timer-value, timer-handler)	register a timer callback

register_for_port_stats_in(handler)	register port statistics handler
register_for_table_stats_in(handler)	register table statistics handler
send_table_stats_request(dp-id)	request table statistics
send_port_stats_request(dp-id, port-no)	request port statistics

Extracted code in appendix section B.4.

6.3.3.4 GMPLS-PCE interface

GMPLS-PCE provides two CORBA interfaces to perform the required both topology and routing operations.

Topology interface

Method	Description
nodeAdd	Add a Node to network Graph
netNodeUpdate	Update Node information
nodeDel	Delete a Node
linkAdd	Add a Link to network Graph
teLinkUpdateCom	Update common Link information
teLinkUpdateGenBw	Update general bandwidth Link information
teLinkAppendIsc	Append an ISC to Link information
teLinkUpdateStates	Update admin/operational states Link information
linkDel	Delete a Link

Routing interface

Method	Description
connectionRoute	connection route request

6.3.4 Use Case Flow:

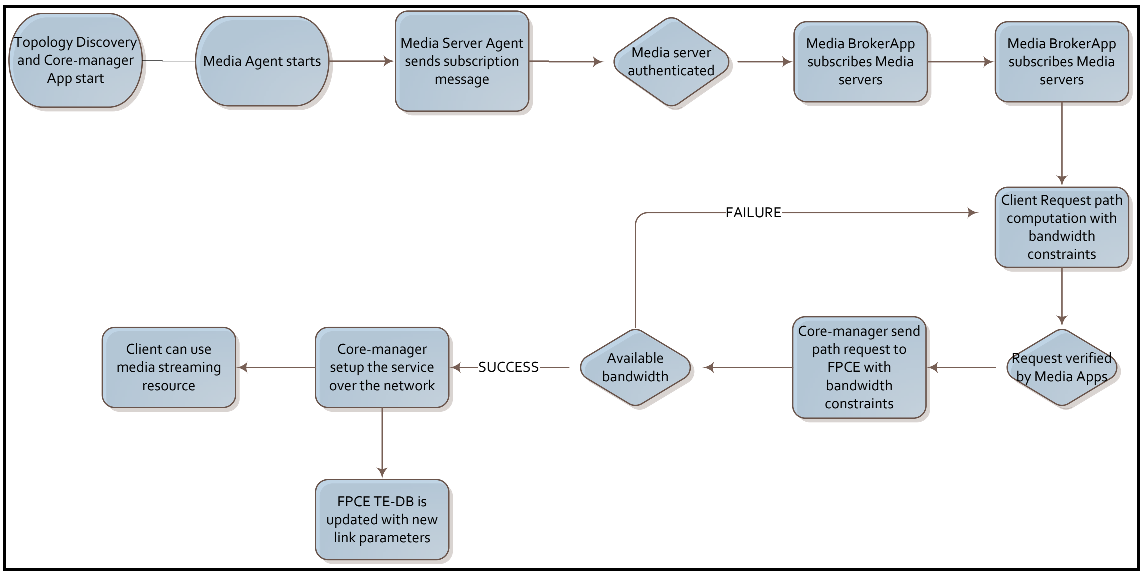

Figure 15 use case 3 flow of events

7 Conclusions

We have described the three technology pilots and we have included the extensive requirements. Operational categories map the technology pilots' requirements to functionalities for better traceability.

From the requirements and the operational categories, we describe the various building blocks for the technology pilots and we have categorised them in terms of users, resources, software and connectivity.

In detail explained the various components of the use case along with the functional and interface details. The flow of events for the use case is also described.

This document gives more details about the technology pilots' building block design and provides the basis for their implementation in Task 5.2. Furthermore it also provides valuable feedback to WP2, WP3 and WP5 activities.

European Commission
Information Society and Media
COOPERATION
CNPq
Conselho Nacional de Desenvolvimento Científico e Tecnológico
60 ANOS
Ministério da
Ciência, Tecnologia
e Inovação
GOVERNO FEDERAL
BRASIL
PAÍS RICO É PAÍS SEM POBREZA

Appendix A. Tech Pilot 2 building blocks details

```
<wsdl:definitions targetNamespace="http://dev.mcp">
	<wsdl:documentation>
		Please Type your service description here
	</wsdl:documentation>

	<wsdl:types>
		<xs:schema attributeFormDefault="qualified"
elementFormDefault="qualified" targetNamespace="http://dev.mcp">
			<xs:import namespace="http://dev.mcp/xsd"/>

			<xs:element name="updateResource">
				<xs:complexType>
					<xs:sequence>
						<xs:element minOccurs="0"
name="resUpdate" nillable="true" type="ax22:DESC"/>
					</xs:sequence>
				</xs:complexType>
			</xs:element>

			<xs:element name="publishMSB">
				<xs:complexType>
					<xs:sequence>
						<xs:element minOccurs="0"
name="resourceCap" nillable="true" type="ax22:DESC"/>
					</xs:sequence>
				</xs:complexType>
			</xs:element>

			<xs:element name="publishMSBResponse">
				<xs:complexType>
					<xs:sequence>
						<xs:element minOccurs="0" name="return"
nillable="true" type="xs:string"/>
					</xs:sequence>
				</xs:complexType>
			</xs:element>
		</xs:schema>

		<xs:schema attributeFormDefault="qualified"
elementFormDefault="qualified" targetNamespace="http://dev.mcp/xsd">
			<xs:complexType name="DESC">
				<xs:sequence>
					<xs:element minOccurs="0" name="activeSessions"
type="xs:long"/>
					<xs:element minOccurs="0" name="atLocation"
nillable="true" type="xs:string"/>
					<xs:element minOccurs="0" name="connectedMSB"
nillable="true" type="xs:string"/>
					<xs:element minOccurs="0" name="cpuSpeed"
type="xs:double"/>
					<xs:element minOccurs="0" name="fileName"
nillable="true" type="xs:string"/>
					<xs:element minOccurs="0" name="fileType"
nillable="true" type="xs:string"/>
```

```
                                    <xs:element minOccurs="0" name="frameRate"
nillable="true" type="xs:string"/>
                                    <xs:element minOccurs="0" name="freeDiskSpace"
type="xs:long"/>
                                    <xs:element minOccurs="0" name="freeMemSize"
type="xs:long"/>
                                    <xs:element maxOccurs="unbounded" minOccurs="0"
name="hasFile" nillable="true" type="xs:string"/>
                                    <xs:element minOccurs="0" name="imgRes"
nillable="true" type="xs:string"/>
                                    <xs:element minOccurs="0" name="maxSessions"
type="xs:long"/>
                                    <xs:element minOccurs="0" name="resourceType"
type="xs:int"/>
                                    <xs:element minOccurs="0" name="resoureURI"
nillable="true" type="xs:string"/>
                                    <xs:element maxOccurs="unbounded" minOccurs="0"
name="supportedCS" nillable="true" type="xs:string"/>
                                    <xs:element maxOccurs="unbounded" minOccurs="0"
name="supportedFR" nillable="true" type="xs:string"/>
                                    <xs:element maxOccurs="unbounded" minOccurs="0"
name="supportedFT" nillable="true" type="xs:string"/>
                                    <xs:element maxOccurs="unbounded" minOccurs="0"
name="supportedImgRes" nillable="true" type="xs:string"/>
                                    <xs:element minOccurs="0" name="totalDiskSpace"
type="xs:long"/>
                                    <xs:element minOccurs="0" name="totalMemSize"
type="xs:long"/>
                              </xs:sequence>
                        </xs:complexType>

                        <xs:complexType name="RESP">
                              <xs:sequence>
                                    <xs:element minOccurs="0" name="desc"
nillable="true" type="xs:string"/>
                                    <xs:element minOccurs="0" name="nodeURI"
nillable="true" type="xs:string"/>
                              </xs:sequence>
                        </xs:complexType>
                  </xs:schema>
            </wsdl:types>

            <wsdl:message name="updateResourceRequest">
                  <wsdl:part name="parameters" element="ns:updateResource"/>
            </wsdl:message>

            <wsdl:message name="publishMSBRequest">
                  <wsdl:part name="parameters" element="ns:publishMSB"/>
            </wsdl:message>

            <wsdl:message name="publishMSBResponse">
                  <wsdl:part name="parameters" element="ns:publishMSBResponse"/>
            </wsdl:message>

            <wsdl:portType name="IMSBPortType">
                  <wsdl:operation name="updateResource">
                        <wsdl:input message="ns:updateResourceRequest"
wsaw:Action="urn:updateResource"/>
```

```
              </wsdl:operation>

              <wsdl:operation name="publishMSB">
                     <wsdl:input message="ns:publishMSBRequest"
wsaw:Action="urn:publishMSB"/>
                     <wsdl:output message="ns:publishMSBResponse"
wsaw:Action="urn:publishMSBResponse"/>
              </wsdl:operation>
       </wsdl:portType>

       .......
       .......
       .......
       .......
       .......

</wsdl:definitions>
```

Appendix B. Tech Pilot 3 building blocks details

B.1 Software Requirements

The list of the requirements refers to Ubuntu 10.10 (Desktop edition) operating system. In particular, the main dependences for the NOX installation (branch ZAKU) are listed in the following table:

Package	Version	Description
autoconf	2.67-2ubuntu1	automatic configure script builder
automake	1:1.11.1-1	A tool for generating GNU Standards-compliant Makefiles
g++	4:4.4.4-1ubuntu2	The GNU C++ compiler
libtool	2.2.6b-2ubuntu1	Generic library support script
swig	1.3.40-3ubuntu1	Generate scripting interfaces to C/C++ code
make	3.81-8	An utility for Directing compilation.
libboost1.42-all-dev	1.42.0-3ubuntu1	Boost C++ Libraries development files (ALL)
libssl-dev	0.9.8o-1ubuntu4.6	SSL development libraries, header files and documentation
python-twisted	10.1.0-2	Event-based framework for internet applications (transitional package)
python-simplejson	2.1.1-1	simple, fast, extensible JSON encoder/decoder for Python
python-dev	2.6.6-2ubuntu2	header files and a static

		library for Python (default)

While the main dependences for the F-PCE installation are listed below:

Package	Version	Description
libomniorb4-dev	4.1.3-1	omniORB core libraries development files
omniidl	4.1.3-1	omniORB IDL to C++ and Python compiler
omniorb	4.1.3-1	IOR and naming service utilities for omniORB
python-omniorb	3.3-1ubuntu1	Python bindings for omniORB

B.2 Topology-db schema

Topology-DB is a MySQL database composed by eight tables. It stores all available topology information (i.e. switches and hosts details, ports capabilities, inter-switches links), tables and ports statistics and the whole flow-entries installed in every switch.

Table	Description
Datapaths	OpenFlow switches connected to NOX
cports_bandwidth	Ports of the circuit switches connected to NOX
flow_entries	Flow entries installed into the datapaths
hosts	Hosts connected to the switches
links	Links between switches
port_stats	Statistics referring to the ports of the controlled switches
ports	Ports of the OpenFlow switches connected to NOX
table_stats	Statistics referring to the tables of all controlled switches

European Commission Information Society and Media
COOPERATION
CNPq 60 ANOS
Ministério da Ciência, Tecnologia e Inovação
GOVERNO FEDERAL BRASIL PAÍS RICO É PAÍS SEM POBREZA

Datapaths table

Field	Type	Description
id	bigint(20) unsigned	Datapath identifier
dID	tinyint(3) unsigned	Unique datapath identifier
name	varchar(20)	Datapath name
ofp_capabilities	int(11) unsigned	Bitmap representing the supported capabilities
ofp_actions	int(11) unsigned	Bitmap representing the supported actions
buffers	int(11) unsigned	Maximum number of packets buffered at once
tables	tinyint(3) unsigned	Number of tables supported by datapath
cports	tinyint(3) unsigned	Number of circuit ports

Ports table

Field	Type	Description
datapath_id	bigint(20) unsigned	Datapath identifier
port_no	smallint(8) unsigned	Port number
hw_addr	varchar(18)	MAC address
name	varchar(16)	Port name
config	int(11) unsigned	Spanning tree configuration flags
state	int(11) unsigned	Spanning tree state
curr	int(11) unsigned	Current features
advertised	int(11) unsigned	Features being advertised by the port
supported	int(11) unsigned	Features supported by the port
peer	int(11) unsigned	Features advertised by peer
sw_tdm_gran	int(11) unsigned	TDM switching granularity flags

sw_type	smallint(8) unsigned	Bitmap of switching type flags
peer_port_no	smallint(8) unsigned	Discovered peer switching port number
peer_dpath_id	bigint(20) unsigned	Discovered peer switching datapath identifier
nodeID	smallint(5) unsigned	Unique node identifier

Cports_bandwidth table

Field	Type	Description
dpid	bigint(20) unsigned	Datapath identifier
port_no	smallint(8) unsigned	Circuit switch port number
num_bandwidth	smallint(8) unsigned	Number of bandwidth array elements
bandwidth	bigint(20) unsigned	Bandwidth value

Hosts table

Field	Type	Description
dpid	bigint(20) unsigned	Datapath identifier
mac_addr	varchar(18)	Host MAC address
ip_addr	varchar(18)	Host IP address
in_port	smallint(8) unsigned	Number of the datapath's port connected to the host
hostID	smallint(5) unsigned	Host unique identifier

Links table

Field	Type	Description
src_dpid	bigint(20) unsigned	Source datapath identifier
src_pno	smallint(8) unsigned	Source port number
dst_dpid	bigint(20) unsigned	Destination datapath identifier

dst_pno	smallint(8) unsigned	Destination port number

Flow_entries table

Field	Type	Description
flow_id	int(11) unsigned	Flow identifier
dpid	bigint(20) unsigned	Datapath identifier
table_id	int(8) unsigned	Unique Table identifier
in_port	int(16) unsigned	Input switch port
idle_timeout	int(16) unsigned	Number of seconds idle before expiration
hard_timeout	int(16) unsigned	Number of seconds before expiration
priority	int(16) unsigned	Priority of the entry
action	varchar(18)	Action to be performed when flow entry is matched
cookie	bigint(64) unsigned	Opaque controller-issued identifier
dl_type	int(16) unsigned	Ethernet frame type
dl_vlan	int(16) unsigned	Input VLAN identifier
dl_vlan_pcp	int(8) unsigned	Input VLAN priority
dl_src	varchar(18)	Ethernet source address
dl_dst	varchar(18)	Ethernet destination address
nw_src	varchar(18)	IP source address
nw_dst	varchar(18)	IP destination address
nw_src_n_wild	int(11)	Bitmap for the network source address wildcard
nw_dst_n_wild	int(11)	Bitmap for the network destination address wildcard
nw_proto	int(8) unsigned	IP protocol or lower 8 bits of ARP code

tp_src	int(16) unsigned	TCP/UDP source port
tp_dst	int(16) unsigned	TCP/UDP destination port

Port_stats table

Field	Type	Description
datapath_id	bigint(20) unsigned	Datapath identifier
port_no	smallint(8) unsigned	Port number
rx_pkts	bigint(64) unsigned	Number of received packets
tx_pkts	bigint(64) unsigned	Number of transmitted packets
rx_bytes	bigint(64) unsigned	Number of received bytes
tx_bytes	bigint(64) unsigned	Number of transmitted bytes
rx_dropped	bigint(64) unsigned	Number of packets dropped by receiver
tx_dropped	bigint(64) unsigned	Number of packets dropped by sender
rx_errors	bigint(64) unsigned	Number of receive errors
tx_errors	bigint(64) unsigned	Number of transmit errors
rx_frame_err	bigint(64) unsigned	Number of errors concerning frame alignment
rx_over_err	bigint(64) unsigned	Number of packets with receiver overrun
rx_crc_err	bigint(64) unsigned	Number of CRC errors
collisions	bigint(64) unsigned	Number of collisions

Table_stats table

Field	Type	Description
datapath_id	bigint(20) unsigned	Datapath identifier
table_id	smallint(8) unsigned	Unique table identifier
max_entries	bigint(32) unsigned	Maximum number of entries supported

active_count	bigint(64) unsigned	Number of active entries
lookup_count	bigint(64) unsigned	Number of packets looked up in table
matched_count	bigint(64) unsigned	Number of packets that match the table

B.3 Interfaces

Northbound interface

VERB	URI	Description
GET	/dpids	Lists datapath identifiers
GET	/dpids/<dpid>	Show detailed information about a specific datapath identifier
GET	/ports	Lists port identifiers
GET	/ports/<*dpid*>&<*portno*>	Show detailed information about a specific port
GET	/links	Lists links information
GET	/hosts	Lists hosts information
GET	/pckt_flows/<dpid>	Lists packet-flows information associated to a specified datapath
DELETE	/pckt_flows/<dpid>	Destroys the specified packet-flow entry
POST	/pckt_flows	Creates a new packet-flow
GET	/pckt_port_stats	Lists packet port statistics
POST	/pckt_port_stats	Creates a new packet port statistics
DELETE	/pckt_port_stats	Destroys the specified packet port statistics
GET	/pckt_table_stats	Lists packet table statistics
POST	/pckt_table_stats	Creates a new packet table statistics
DELETE	/pckt_table_stats	Destroys the specified packet table statistics
POST	/pckt_host_path	Creates a new packet host-path request

Southbound interface

VERB	URI	Description

POST	/pckt_dpid	Creates a new packet-datapath
DELETE	/pckt_dpid/<dpid>	Destroys the specified packet-datapath
POST	/pckt_intersw_link	Creates a new packet-interswitch link
DELETE	/pckt_intersw_link	Destroys the specified packet-interswitch link
POST	/pckt_host	Creates a new packet-host
DELETE	/pckt_host	Destroys the specified packet-host

B.4 Flow Monitoring code

The extracted code to perform the flow entries request is described below:

```
from nox.netapps.flow_fetcher.pyflow_fetcher import flow_fetcher_app

def install(self):
        self._ffa = self.resolve(flow_fetcher_app)

...
def __dpid_request(self):
        try:
            dpid = datapathid.from_host(self._queue.pop())
            FFLOG.debug("Request flows for dpid=%s", str(dpid))
            ff = self._ffa.fetch(dpid, {},
                          lambda: self.__flows_replay(dpid, ff))

        except Exception as e:
            FFLOG.error(str(e))
```

The list of CRUD operations performed on core-manager resources:

- GET datapaths identifier
- GET ports identifier
- POST packet flow entry (using JSON format)
- POST packet table statistics (using JSON format)

- POST packet port statistics (using JSON format)
- DELETE packet flows entries
- DELETE packet table statistics
- DELETE packet port statistics

While an example of how requesting a table statistic by using JSON format is described below:

```
def __table_stats_create(self, dpid, info):
        h_ = {'content-type': 'application/json'}
        try:
            payload = {"dpid": dpid,
                       "table_id": info['table_id'],
                       "max_entries": info['max_entries'],
                       "active_count": info['active_count'],
                       "lookup_count": info['lookup_count'],
                       "matched_count": info['matched_count']}

            r_ = requests.post(url=self._url + 'pckt_table_stats',
                               headers=h_, data=json.dumps(payload))
            if r_.text != 'Operation completed':
                FFLOG.error("An error occurring during table-stats-post!")

        except Exception as e:
            FFLOG.error(str(e))
```

B.5 GMPLS-PCE Interface

A pseudo-code from topology.idl file:

```
module TOPOLOGY {
...
         interface Info {
                void
            nodeAdd(in  gmplsTypes::nodeIdent  id)
                raises(<Exceptions>);

            void
            nodeDel(in  gmplsTypes::nodeIdent  id)
```

```
        raises(<Exceptions>);

    void
    netNodeUpdate(in  gmplsTypes::nodeId        id,
              in  gmplsTypes::netNodeParams  info)
        raises(<Exception>);

    void
    linkAdd(in  gmplsTypes::teLinkIdent ident)
        raises(<Exceptions>);

    void
    linkDel(in  gmplsTypes::teLinkIdent ident)
        raises(<Exceptions>);

      void
    teLinkUpdateCom(in  gmplsTypes::teLinkIdent     ident,
               in  gmplsTypes::teLinkComParams  info)
        raises(<Exceptions>);

    void
    teLinkUpdateGenBw(in gmplsTypes::teLinkIdent  ident,
               in gmplsTypes::bwPerPrio    availBw)
        raises(<Exceptions>);

    void
    teLinkAppendIsc(in gmplsTypes::teLinkIdent     ident,
               in gmplsTypes::iscSeq         iscs)
        raises(<Exceptions>);

      void
    teLinkUpdateStates(in gmplsTypes::teLinkIdent  ident,
                in gmplsTypes::statesBundle states)
          raises(<Exceptions>);
  };
```

A pseudo-code from pcera.idl file:

```
module PCERA {
...
    interface RoutingServices {
          void
    connectionRoute(in  gmplsTypes::connEndPoint   src,
               in  gmplsTypes::connEndPoint   dst,
               in  gmplsTypes::callIdent      callId,
```

```
                    in  gmplsTypes::lspParams        lspInfo,
                    in  gmplsTypes::routeMetricSeq bounds,
                    out gmplsTypes::eroSeq           wEro,
                    out gmplsTypes::eroSeq           pEro)
               raises(<Exceptions>);
       };
```

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

END OF DOCUMENT