Integrating OpenFlow in IMS Networks and Enabling for Future Internet Researchand Experimentation

Christos Tranoris[1], Spyros Denazis[1],Nikos Mouratidis[2],
Phelim Dowling[3], and Joe Tynan[3]

[1] University of Patras, Greece
tranoris@ece.upatras.gr, sdena@upatras.gr
[2] Creative Systems Engineering (CSE) , Greece
n.mouratidis@creativese.eu
[3] TSSG, Ireland
{Pdowling,jtynan}@tssg.org

Abstract. The IP Multimedia Subsystem (IMS) is an architectural framework for delivering IP multimedia services. The appearance of Software Defined Networks (SDNs) concept in the IMS fabric can unleash the potential of the IMS technology which enables access agnostic services including applications like video-conferencing, multi-player gaming, white boarding all using an all-IP backbone. SDN requires some method for the control plane to communicate with the data plane. One such mechanism is OpenFlow which is a standard interface for controlling computer networking switches. This work presents our experience and implementation efforts in integrating OpenFlow mechanisms within IMS. Since this work also is done within the Future Internet Research and Experimentation domain, we also describe how we enabled our infrastructures with experimentation mechanisms.

Keywords: IMS networks, OpenFlow, Software Defined Networking, Future Internet research, Experimentation.

1 Introduction

Experimental networks are built to support experimentally-driven research towards the Future Internet Research (FIRE)[1]. The term experimental networks characterizes diverse communication technologies networks, composed of virtual or physical nodes that allow experimenters to test early prototypes of new protocols, protocol stacks, routing algorithms, services and applications. The main questions regarding the suitability of a testbed to incubate experiments towards the Future Internet are summarized in the capability to support large scale experiments, and to be able to federate with other testbeds. The first prerequisite has been addressed with the introduction of technologies for network virtualization.

OpenFlow was born by the need of experimenting on new technologies, however on commercial grade networks. The OpenFlow technology and the associated OpenFlow Protocol have come to lower the entry-barrier of new ideas application in commercial grade networks. In contrast to the virtual networks technology, OpenFlow

A. Galis and A. Gavras (Eds.): FIA 2013, LNCS 7858, pp. 77–88, 2013.

offers a flexible way of manipulating the routing tables of Ethernet switches. Providing a generic approach to this manipulation, OpenFlow is a technology that allows formation of large testbeds, thus enables large scale experimentation.

In what concerns the second prerequisite, testbed federation has become possible via integration within the context of control frameworks. Such frameworks allowed for testbed resources visualization and provisioning to the experimenters. Nowadays, there are several control frameworks in use, with the most prominent ones being the OMF [2], PII project's Teagle [3][4]. Information exchange between the control frameworks and the underlying testbeds is implemented using particular APIs, such as the Slice-Based Facility Architecture-SFA [5] and the ProtoGENI [6][7].

Building on both enablers of large scale experimentation, the present effort coming from the OpenLab project [8], aims at providing experimenters with more agility concerning exploitation of testbed resources. In particular, we aim at extending experimentation across mixed virtual and physical infrastructure nodes in order to allow running experiments requiring mixtures of pragmatic offerings (e.g. QoS) as well as virtual ones (e.g. non-IP based algorithms). Under the proposed setting, an experimenter may work on a new algorithm using the clean-slate virtual network environment but at the same time can test deployment-related issues such as performance, mapping a set of nodes on physical infrastructure nodes.

As a first step towards this approach, in this paper we present an approach towards OpenFlow deployment in IMS testbeds to be exploited for QoS-based experiments. Section 2 presents why and how we introduced SDN concepts in IMSs, Section 3 provides the implementations to the respective testbeds. Section 4 presents our first approach on enabling these infrastructures for experimentation while section 5 provides some potential experiments. Finally, we conclude our work.

2 Introducing SDN to IMS: Integrating OpenFlow

The IP Multimedia Subsystem (IMS) is an architectural framework for delivering IP multimedia services. It was originally designed by the wireless standards body 3rd Generation Partnership Project (3GPP), as a part of the vision for evolving mobile networks beyond GSM. Unfortunately in many respects it remained just a vision, with sporadic deployments. Operators and Network Providers somehow could not manage to find any compelling reason to re-architect their network by introducing IMS network elements. On the other hand there were no promising applications. But IMSs definitely hold enormous potential and a couple of breakthroughs in key technologies can result in the ‘tipping point’ of this great technology which promises access agnostic services including applications like video-conferencing, multi-player gaming, white boarding all using an all-IP backbone.[9]

However the last few years the concept of Software Define Networks (SDNs) appeared. SDN decouples the system that makes decisions about where traffic is sent (the control plane) from the underlying system that forwards traffic to the selected destination (the data plane). This architecture allows network administrators to have programmable central control of network traffic without requiring physical access to the network's hardware devices. SDN requires some method for the control plane to communicate with the data plane. One such mechanism is OpenFlow which is a

standard interface for controlling computer networking switches. An OpenFlow Controller is able to control network resources in a programmatic manner. These network resources can span routers, hubs and switches, known as a slice, and can be controlled through the OpenFlow Controller. The OpenFlow protocol can manage resources by slicing them in a virtualized manner and that aspect of the OpenFlow protocol can be integrated in an IMS infrastructure.

In the context of the OpenLab[8] project, two IMS testbeds participate in order to offer the IMS technology for further experimentation scenarios, the OSIMS testbed from the University of Patras and the TSSG IMS Testbed. These two testbeds were also used in the Panlab[4] project. Within OpenLab, a goal for these two testbeds is to deploy OpenFlow and SDN characteristics in order to introduce QoS and interconnection features. The OpenFlow Controller will be able to dynamically re-route the traffic to alternate network resources, or a different 'network slice' in cases of congestion or applying QoS in IMS networks through the IMS policy network elements.

3 Enhancements and Implementation on IMS Testbeds

This section describes how OpenFlow is integrated into to IMS testbeds: The OSIMS testbed at University of Patras and a Telco Cloud Testbed at TSSG. The two testbeds participate in the OpenLab project and have adopted proposed federation mechanisms like SFA [5], making thus possible to be included in much more complex and federated experimentation scenarios.

3.1 The UoP OSIMS Testbed

The UoP OSIMS testbed[10] with its current setup is (partially) depicted in Figure 1. The core of OSIMS system is based on the Open Source IMS core of OpenIMS[11]. To ease experimentation on an IMS testbed, OSIMS offer many services available by accessing the Patras Platforms for Experimentation (P2E) portal [10].

OSIMS consists of the following services:

- A Home Subscribe Server (HSS)
- A Serving-CSCF Module (scscf)
- A Proxy-CSCF Module (pcscf)
- An Interrogating-CSCF Module (icscf)
- An OpenSIPS server and a presence server based on OpenSIPS
- A presence server based on OpenSIPS
- An XDMS service for storing directory information
- An Asterisk server for connection to other phone services
- A media server, streaming video channels

Figure 1 displays also the undergoing extensions on OSIMS testbed, which are based on the considerations presented on section 2 about introducing QoS and OpenFlow on the IMS fabric. To apply QoS in IMS networks two network elements are needed: the Policy Decision Point (PDP) and the Policy Enforcement Point (PEP). The PDP retrieves related policy rules, in response to a RSVP message, which the PDP then sends to the PEP. These are implemented by two IMS components: a Policy and Charging Rule Function (PCRF), and a Policy and Charging Enforcement Point (PCEF).

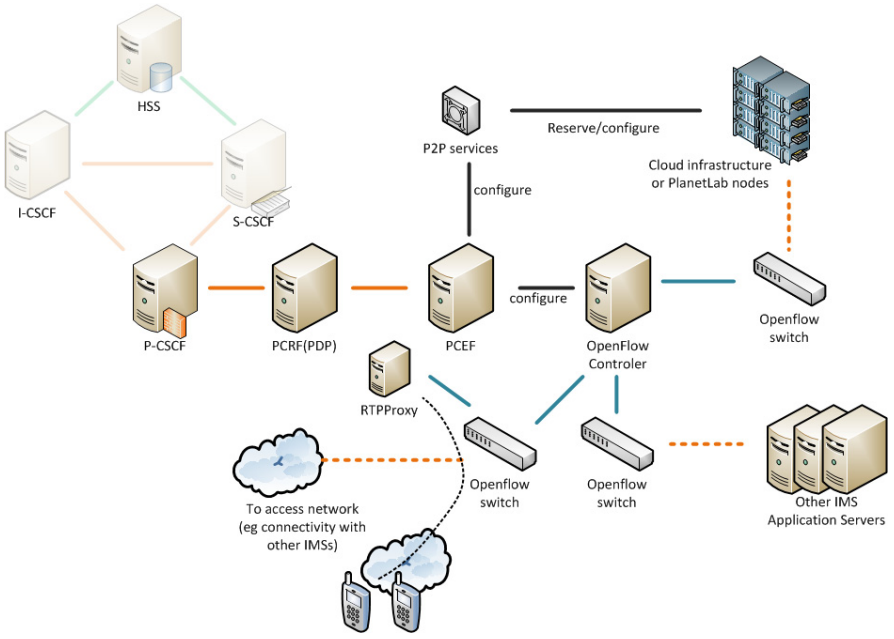


Fig. 1. UoP OSIMS testbed with OpenFlow controller and switches deployed

The P-CSCF is a central IMS core element for SIP signaling. It is the only element that receives information about the user signaling for multimedia sessions. It sends a description of the session the user tries to establish to the PCRF. The PCRF, which plays the role of the PDP, is the element which authorizes the session and does the policy control and flow based charging. PCRF sends through any interface commands to the PCEF. PCEF, which plays the role of PEP, is co-located with the domain gateway and its role is to enforce the policies that the PCRF requires.

In an IMS call flow, the SDP (Session Description Protocol) message is encapsulated within SIP and carries the QoS parameters. The PCRF examines the parameters, retrieves appropriate policies and informs the PCEF for that traffic flow. The advantage of using OpenFlow Controller/OpenFlow switch to the PDP/PEP

combination would be the ability to adapt the network flow according to bandwidth changes and traffic.

In OSIMS we installed two open source components of PCRF and PCEF based on the OpenIMSCore. We installed the Floodlight[12] OpenFlow controller while the OpenFlow switches are based on OpenVSwitch[13]. The approach is depicted in Figure 2. We replaced the PCEF with our own functionality in order to communicate with the OpenFlow Floodlight Controller via its RESTful interface. We enhanced the PCEF with a custom API that it is able to identify: i) bandwidth speeds, ii) the network slice which the flows should follow. These will configure our OpenFlow based OpenVSwitches to control network resources, to alternate network slices, or for example connect to cloud resources, to other Application servers or provide connectivity (for example to other IMSs). Using OpenFlow together with policies we will have the ability to dynamically adapt and reroute the network flow according to bandwidth traffic, to an alternate network resource or slice.

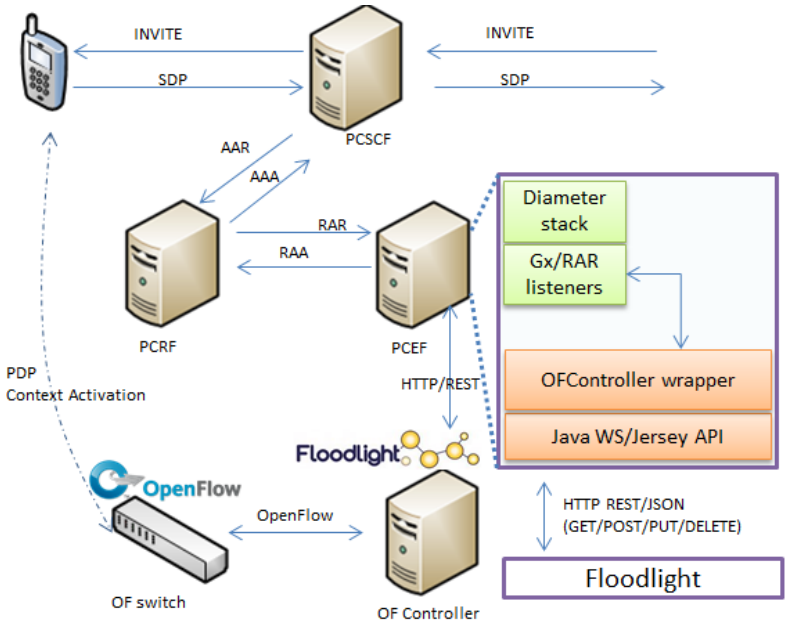
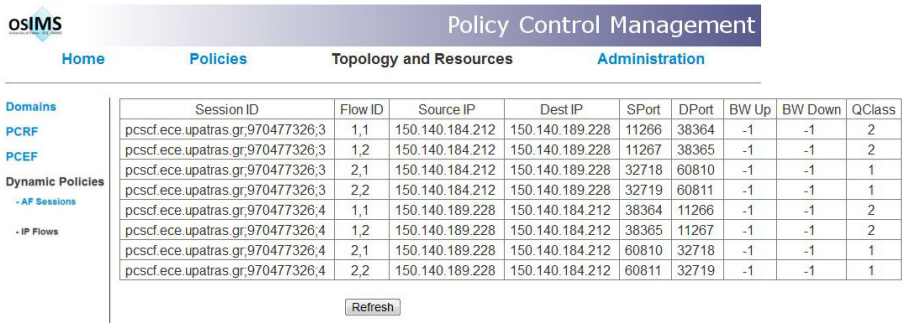


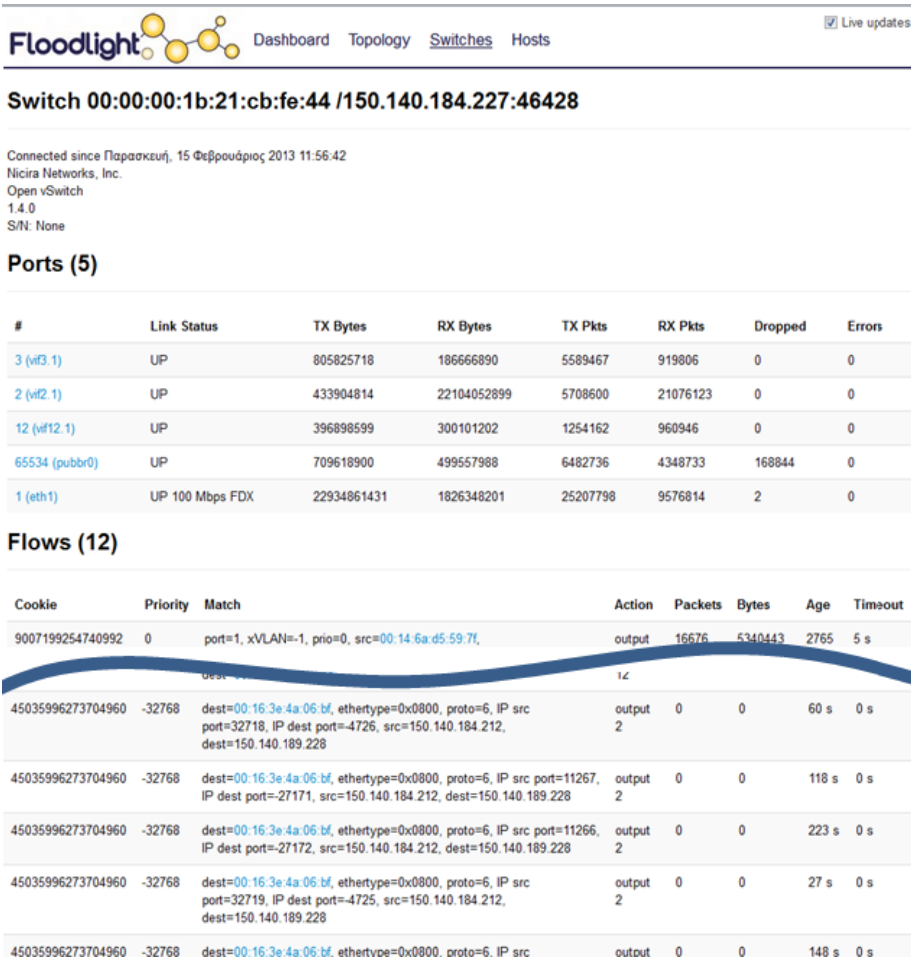
Fig. 2. Openflow integration in OSIMS

Figure 3 and Figure 4 display both what is reported during a video-call between two IMS clients. Figure 3 displays the 8 flows that have been identified that are need in order to establish the call. We need 4 flows for each client. 2 flows are for audio and 2 for video, because they are RTP streams. For each flow there is also an identified Quality of Service Class from the SDP headers, while it is also possible to define the available Bandwidth according to some policies defined within the Policy Control Management service. Figure 4 displays the control panel of the Floodlight Controller, where some static flows are injected through the PCEF component. The figure displays all the flows currently existing in the switch.



Session ID	Flow ID	Source IP	Dest IP	SPort	DPort	BW Up	BW Down	QClass
pcscf.ece.upatras.gr;970477326;3	1,1	150.140.184.212	150.140.189.228	11266	38364	-1	-1	2
pcscf.ece.upatras.gr;970477326;3	1,2	150.140.184.212	150.140.189.228	11267	38365	-1	-1	2
pcscf.ece.upatras.gr;970477326;3	2,1	150.140.184.212	150.140.189.228	32718	60810	-1	-1	1
pcscf.ece.upatras.gr;970477326;3	2,2	150.140.184.212	150.140.189.228	32719	60811	-1	-1	1
pcscf.ece.upatras.gr;970477326;4	1,1	150.140.189.228	150.140.184.212	38364	11266	-1	-1	2
pcscf.ece.upatras.gr;970477326;4	1,2	150.140.189.228	150.140.184.212	38365	11267	-1	-1	2
pcscf.ece.upatras.gr;970477326;4	2,1	150.140.189.228	150.140.184.212	60810	32718	-1	-1	1
pcscf.ece.upatras.gr;970477326;4	2,2	150.140.189.228	150.140.184.212	60811	32719	-1	-1	1

Fig. 3. A view of the Policy Control Panel while a video call



#	Link Status	TX Bytes	RX Bytes	TX Pkts	RX Pkts	Dropped	Errors
3 (vif3.1)	UP	805825718	186666890	5589467	919806	0	0
2 (vif2.1)	UP	433904814	22104052899	5708600	21076123	0	0
12 (vif12.1)	UP	396898599	300101202	1254162	960946	0	0
65534 (pubbr0)	UP	709618900	499557988	6482736	4348733	168844	0
1 (eth1)	UP 100 Mbps FDX	22934861431	1826348201	25207798	9576814	2	0

Cookie	Priority	Match	Action	Packets	Bytes	Age	Timeout
9007199254740992	0	port=1, xVLAN=-1, prio=0, src=00:14:6a:d5:59:7f,	output	16676	5340443	2765	5 s
45035996273704960	-32768	dest=00:16:3e:4a:06:bf, ethertype=0x0800, proto=6, IP src port=32718, IP dest port=-4726, src=150.140.184.212, dest=150.140.189.228	output 2	0	0	60 s	0 s
45035996273704960	-32768	dest=00:16:3e:4a:06:bf, ethertype=0x0800, proto=6, IP src port=11267, IP dest port=-27171, src=150.140.184.212, dest=150.140.189.228	output 2	0	0	118 s	0 s
45035996273704960	-32768	dest=00:16:3e:4a:06:bf, ethertype=0x0800, proto=6, IP src port=11266, IP dest port=-27172, src=150.140.184.212, dest=150.140.189.228	output 2	0	0	223 s	0 s
45035996273704960	-32768	dest=00:16:3e:4a:06:bf, ethertype=0x0800, proto=6, IP src port=32719, IP dest port=-4725, src=150.140.184.212, dest=150.140.189.228	output 2	0	0	27 s	0 s
45035996273704960	-32768	dest=00:16:3e:4a:06:bf, ethertype=0x0800, proto=6, IP src	output	0	0	148 s	0 s

Fig. 4. The Floodlight web Control Panel with the programmed flows

3.2 The TSSG Testbed

The software implementation for the aforementioned OpenFlow integration in an IMS environment, is based on the OSGI (Open Services Gateway initiative) framework with the use of Maven and PAX. Maven is the build automation tool and OPS4J' PAX's plugin provides the build goals to create, build, manage and deploy OSGI bundles into a contained executable environment. The major benefits of implementing the framework into an OSGI container include effective runtime environment, module life cycle, standard services, and common deployment platform. If the system requires additional resources, OSGI has the ability to distribute modules to other containers via declarative services. Another key feature of the framework is the capability to run modules at distinctive start time levels, permitting the system to categorise functionality into start levels e.g. system debug logging at start level 6, views at start level 5, where core logic modules starting at lower start levels and so on.

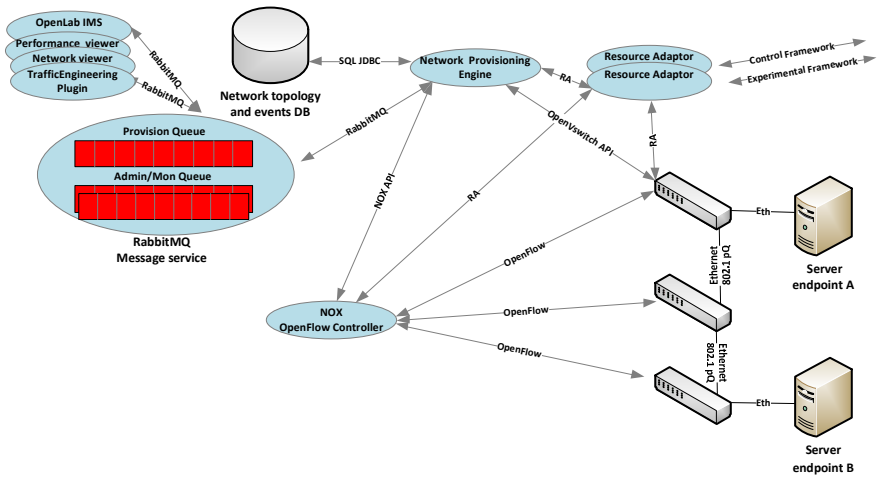


Fig. 5. OpenFlow implementation architecture, queues and protocol flow diagram in TSSG's OpenIMS platform

Further to the components been developed to run control logic in the system, the framework statically deploys elements such as OpenVSwitch, OpenFlow enabled switches and OpenFlow controller parts. These component parts are mostly incorporated in the lower layers of the system where they perform the necessary role in controlling, provisioning and administering underlying network resources. These network elements are the back bone of the data plane and carry out the network path/flow decisions established via the OpenFlow protocol. In this architecture we use OpenVSwitch which provides a native ingress/egress rate limit attribute, with a valuable implementation of the OpenFlow protocol. The OpenVSwitch at the "control plane" performs as a flow filter, flow entry and exit points and additionally provides a platform for transit flows.

At the upper layer there is an API implementation in Java/RabbitMQ that will grant access into the architectures control plane logic and in turn access to OpenFlow which manages network elements. This allows network applications a generic API to the network resources whether it's for topology discovery, slice provisioning, bandwidth provisioning, performance stats, etc. Also in the upper layers, and hosted in the OSGI platform , are the testbed resource adapters, so testers and experimenters alike can provision, evaluate and probe the system for findings.

Bolted onto the system is the capacity for presentence. Here JavaDB ties into the OSGI stack effectively, offering both memory based cache for speed queries and a more permanent read/write location.

The Network Provisioning Engine module, acts as an OpenFlow IMS bandwidth/slice controller that allows the administration of bandwidth or flows that transverse through the underlying communication network. With a flexible implementation of components like RabbitMQ for messaging, JavaDB for both long term and short persistence and Floodlight as an OpenFlow controller, it allows for a loose but robust coupling to the Network Provision Engine. The module has the ability to register and activate message queues for the purposes of logging, performance records, topology views, interact with the data store, provisioning etc.

The network provisioning engine implementation is divided into two logical regions. The first is a Discovering / Monitor service and the second is a Provisioning service:

1. Discovering / Monitor services: This service will use the Floodlight controller to build a logical network overview. The overview will compose of nodes, links and interfaces, all modelled and stored in the local data store. Events derived from the under lying network, such as node failure, operational state of links, etc., will be passed on the north bound interface for processing and likewise archived in the local database. An event ACL can be applied to the north bound interface so the upper layer application or plugin will only secure events it requires.
2. Provision services: This service will allow OpenFlow switched nodes to be provisioned, enabling applications outside the network layer in this instance of the IMS plugin access to data transport resources. It is at this point flows are deleted and established across the network, connecting endpoints together.

Operationally the system can have any number for plugins, mimicking a telco north bound interface. Enabling a plugin is as easy as that plugin joining a functional topic in the relevant message queue or queues. One such plugin is the Open IMS plugin, which disseminates network slices from a predefined algorithm set which it contains and on a network topology it has learnt prior. In essence, taking control of the switches flow table to establish a data flow in the network.

4 Enabling for Experimentation

To provide the given testbed setup for experimentation, it is necessary to encode its resources so that they are compatible with the provisioning framework called Teagle. Such encoding is possible by implementing particular XML description tables called Resource Adapters (RA). In Figure 6, we display the concept of the resource adapter

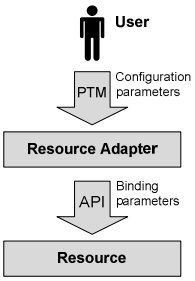


Fig. 6. A Resource Adapter configures a resource through an API

As it is illustrated in Figure 7, a PTM may include several RAs, with each one being dedicated to representing to the Teagle a particular resource or capability of the underlying network. In our realisation we have implemented three RAs, one representing the network setup (VM), another implementing ingress and egress policing rates and bursts (qos_rate_limit), and a third one for sflow monitoring client configuration (sflow). Using the qos_rate_limit RA it is possible to run QoS performance experiments by changing the ingress, egress traffic rates. These RAs have already implemented in OSIMS while in the TSSG testbed is under adaptation and adoption.

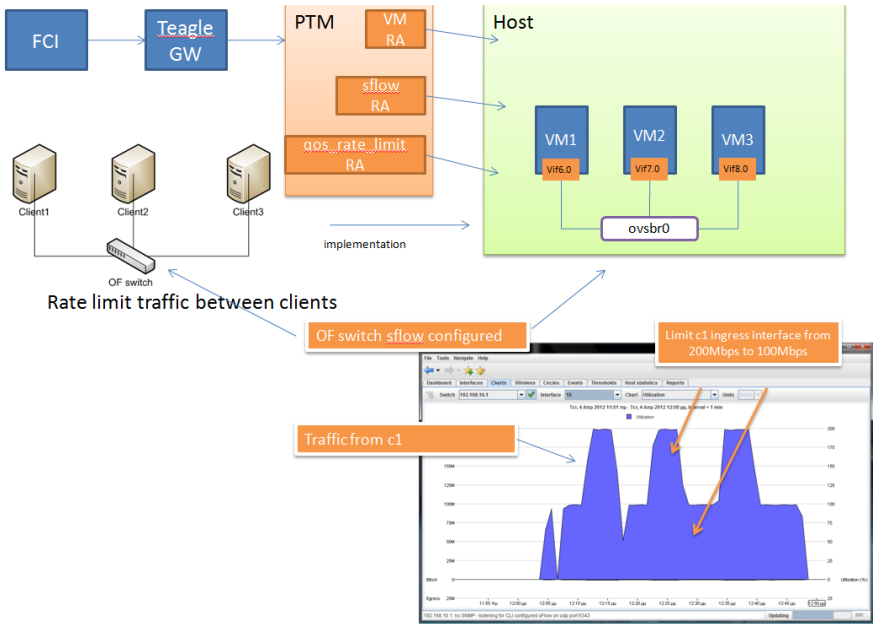


Fig. 7. Enabling the testbed for experimentation

The overall testbed-provisioning framework configuration depicted in Figure 7, is administered by an experiment controller called Federation Computing Interface (FCI) [14][15]. The FCI is used by the experimenters to request and configure resources.

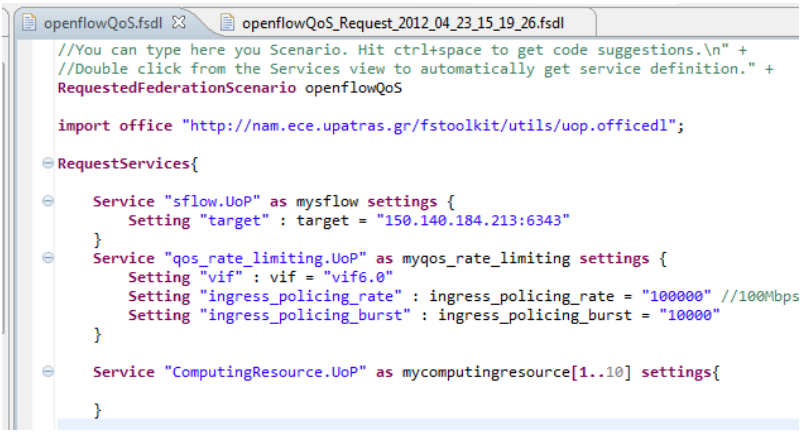


Fig. 8. Requesting and configuring testbed resources

Access to the testbed resources by the experimenters is possible via the Federation Scenario Toolkit [16]. With this tool, an experimenter can create specific scenarios and configure the network resources by expressing the experiments in the Federation Scenario Description Language (FSDL), an example of which is shown in Figure 8. In the current example, the user requests use of virtual machines of the testbed and configures the virtual interface of one of them with maximum ingress policing rate. In addition, the experimenter configures the monitoring sflow agent's IP address, where the openvswitch sends information.

Throughout experimentation, the experimenters can also access the allocated to the experiment network switches through public IPs.

4.1 Integration with other Control and Monitoring Frameworks

The Slice Federation Architecture (SFA) [5] aims to become a standard for testbed resources discovery, authentication, and authorization. The SFA architecture is entirely decentralized and thus enables massive scale experimentation through federation with other control frameworks, and does not assume resource description models but convey them as-is.

Seen as a control framework, the SFA takes cares of two main testbed related issues; the testbed resources description and reservation. The provisioning of the aforementioned OpenFlow functionality to the experimenters though the SFA control framework, requires the existence of an SFA-compliant interface for the Aggregate Manager (AM) within the PTM component. In SFA terminology, components are the offered resources that are independently owned and operated. Components can be organized into aggregates, which are groups of resources owned and administered as

an ensemble by some organization. The AM is the domain authority that exposes and controls these resources.

The AM is the point of convergence of the PTM functionality towards the provisioning framework. Thanks to the AM resource specifications testbed resources may become available to the experiments through the SFA control framework. Integration of the PTM component into the SFA framework is done through an implemented SFA compliant API that translates SFA messages into PTM compliant ones and vice versa. The implementation is accomplished by adopting the so called SFAWrapper [17] implemented within the OpenLab project and by enhancing it with our testbed specific entities.

5 Target Experiments/Experimenters

Currently the testbeds were used to experiment with the IMS technology alone. We have envisaged the following usage scenarios that one can execute over such federated infrastructures:

5.1 QoS with Policy Enforcement and OpenFlow Control

Let the experimenter define policies in the PCRF and his IMS Client. Monitor the SIP/SDP message and how the policies are enforced from the OpenFlow Controller to an OpenFlow switch. Experiment with new functionality within the OpenFlow controller.

Expected experimentation results: Telcos can experiment on the results of having an integration of the OpenFlow technology and SDN concepts into their core network prior applying this into their own solutions.

5.2 Prioritizing Traffic between 2 IMS Cores Exchanging Data

In OpenLab we have two IMS Core testbeds and the PlanetLab infrastructure. In such a scenario one can try to define link bandwidth between the two IMS Cores over a best-effort internet connection. Send data between the two networks, but prioritize the SIP traffic. All these while establishing calls.

Expected experimentation results: Having the same link between the two cores, while there is a high demand, the experimenter can monitor how SIP traffic is prioritized over UDP traffic.

6 Conclusions and Future Work

Introducing SDN concepts within the IMS fabric seems to be quite promising as discussed in section 2. Integrating the OpenFlow Protocol to the IP Multimedia network can provide better resource control and advanced QoS support as section 3 presented on the integration within our IMS fabric. We expect also to provide much more interesting applications with these new deployments such as those presented in section 5. Experimenters can benefit by exploiting these new potentials, while not

having to deal with complex deployments before they decide to do so. They can test applications and algorithms involving such new technologies by investing less time in preparing and configuring equipment. Finally, for both our testbeds, we plan to provide ready scenarios for certain use cases, to ease experimenters with the learning process of the whole experimentation lifecycle.

Acknowledgments. The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) from project under grant agreement n° 287581 – OpenLab.

References

[1] Gavras, A., Karila, A., Fdida, S., May, M., Potts, M.: Future Internet Research and Experimentation: The FIRE Initiative. ACM SIGCOMM Computer Communication Review 37(3) (July 2007), doi:http://doi.acm.org/10.1145/1273445.1273460
[2] OMF control framework, http://www.mytestbed.net/
[3] FITeagle, http://fiteagle.org/
[4] Wahle, S., Tranoris, C., Denazis, S., Gavras, A., Koutsopoulos, K., Magedanz, T., Tompros, S.: Emerging testing trends and the Panlab enabling infrastructure. IEEE Communications Magazine 49(3), 167–175 (2011), doi:10.1109/MCOM.2011.5723816
[5] SFA, http://svn.planet-lab.org/wiki/SFATutorial
[6] ProtoGENI, http://protogeni.net/
[7] National Science Foundation, GENI website, http://www.geni.net
[8] OpenLab FP7 EU project, http://www.ict-openlab.eu/
[9] http://gigadom.wordpress.com/2011/10/04/adding-the-openflow-variable-in-the-ims-equation/
[10] Patras Platforms for Experimentation, http://nam.ece.upatras.gr/ppe
[11] OpenIMS core, http://www.openimscore.org/
[12] Floodlight Openflow controler, http://floodlight.openflowhub.org/
[13] openVSwitch, http://openvswitch.org/
[14] Tranoris, C., Denazis, S.: Federation Computing: A pragmatic approach for the Future Internet. In: 6th IEEE International Conference on Network and Service Management (CNSM 2010), Niagara Falls, Canada, October 25-29 (2010)
[15] Federation Computing Interface (FCI), Panlab wiki website (February 12, 2012), http://trac.panlab.net/trac/wiki/FCI
[16] Federation Scenario Toolkit (FSToolkit) web site (February 12, 2012), http://nam.ece.upatras.gr/fstoolkit
[17] SFAWrapper, http://sfawrap.info/