# Towards a Carrier Grade SDN Controller: Integrating OpenFlow With Telecom Services

Caio Ferreira[†], Natal Neto[†], Alex Mota[†], Luiz C. Theodoro[†], Flávio de Oliveira Silva[*†],
João Henrique de Souza Pereira[*], Augusto Neto[‡§],Daniel Corujo[‡],
Carlos Guimarães[‡], Pedro Frosi Rosa[†], Sergio Takeo Kofuji[*] and Rui Aguiar[‡]
[*]Polytechnic School, University of São Paulo - Brazil
São Paulo, São Paulo, 05424-970
Email: flavio@pad.lsi.usp.br, joaohs@usp.br, kofuji@pad.lsi.usp.br
[†]Faculty of Computing, Federal University of Uberlândia - Brazil
Uberlândia, Minas Gerais, 38400-902
Email: {flavio, frosi}@facom.ufu.br, {caiocf,natal,alex,lclaudio}@algartelecom.com.br
[‡]Instituto de Telecomunicações, Universidade de Aveiro - Portugal
Aveiro, Portugal, 3810-193
Email: {dcorujo,carlos.guimaraes,ruilaa}@ua.pt
[§]Dept. de Informática e Matemática Aplicada(DIMAp), Universidade Federal do Rio Grande do Norte - Brazil
Natal, Rio Grande do Norte, 59078-970
Email: augusto@dimap.ufrn.br

***Abstract*—Software-Defined Networking (SDN) essentially decouples the hardware from the software that controls it. Currently, some SDN abstractions are materialized by OpenFlow and several OpenFlow controllers, based on different programming paradigms and architectures, are available. Usually, these controllers are bundled with some sample applications that enables the construction of new ones by using their particular way. However, these applications focus on specific services being tightly coupled with the switch behavior. In this scenario, SDN community is working to build a SDN control layer that meets carrier grade requirements such as throughput, availability and scalability. This work proposes a new SDN controller architecture that is integrated with a carrier grade service level execution environment, based on Service Logic Execution Environment (SLEE) architecture, defined under the Java APIs for Integrated Networks (JAIN) initiative. The proposed approach extends SDN based services by integrating OpenFlow with several network resources and communication protocols providing a cross layer platform that can satisfy these telecom operators requirements.**

***Keywords–Software-Defined Networking; Carrier Grade; Controller; Telecommunications.***

## I. INTRODUCTION

Software-Defined Networking (SDN) [1], [2] is a promising networking technology since it has the potential to enable innovation and also to give the network operators more control of their infrastructure. SDN market is expected to reach thirty five billion dollars by 2018 [3], [4]. Essentially, SDN decouples the forwarding plane from the control plane. Currently, Openflow [5] materializes some concepts of SDN. In such approach, a signaling protocol is defined for the networking control plane, which enables controllers to orchestrate OpenFlow-compliant network devices (e.g., switch and wireless access points) in a programmable way by a controller.

The OpenFlow-enabled SDN is a key contribution that is propelling the networking research community towards the definition of the Future Internet, allowing the use and evaluation of innovating mechanisms for both network control and data transport. The list of Future Internet mechanisms include, among others, innovative approaches for routing, mobility, Quality of Service/Experience control, optical resource control. Future Internet Testbeds Experimentation Between Brazil and Europe (FIBRE) [6], Global Environment for Network Innovations (GENI) [7], OpenFlow in Europe: Linking Infrastructure and Applications (OFELIA) [8] and Abstraction Layer for Implementation of Extensions in Programmable Networks (ALIEN) [9] are examples of research projects across the world that are envisaging to spread the use of SDN by enabling the experimentation on top of an OpenFlow-enabled infrastructure of some of these innovate mechanisms.

The high demands for improving more and more the reliable aspects of their network systems and also the ability to take a complete control of their infrastructure, allowing customization and optimization and thus reducing overall capital and operational costs recently sparked the interests of telecom companies in exploring the use of OpenFlow-enabled SDN.

However, SDN poses some research challenges [10] to the scientific community. A key challenge is related with the controllers that should meet carrier grade requirements [11] that encompasses high availability, scalability, high performance, reliability, fault tolerance and manageability in order to foster the SDN adoption in mission critical environments, such as the ones handled by telecom operators.

Currently, several telecom operators have services [12] deployed on top of a mature platform, known as JAIN SLEE. The JAIN [13] is a set of APIs dedicated to creating voice and data convergent services. The JAIN SLEE [14] is a component model that supports the deployment of event driven applications that requires carrier grade requirements [15], [16], [17]. JAIN SLEE is available as commercial products, such as Rhino [18], and also as open source platform, such as Mobicents [19].

 ISBN: 978-1-61208-360-5 

At this moment, the SDN community is still in pursuit of a carrier grade SDN control layer that is suitable for the demands of such environments. This work presents a contribution to this research by using an approach that differs from efforts currently undertaken by the SDN community in relation to this quest. This approach consists of constructing a controller layer that uses the component model defined by JAIN SLEE.

The remainder of this paper is structured as follows: Section II describes related projects and presents the ecosystem of OpenFlow controllers currently available. Section III presents the JAIN Slee component model and the components created as the basis for the carrier grade SDN controller that enables a cross layer approach by integrating OpenFlow with protocols used by telecom's voice and data services. Section IV presents an application scenario where the proposed approach was used to integrate OpenFlow and the Multimedia Independent Handover (MIH) protocols [20] and finally, in Section V, we present some concluding remarks and future work.

## II. OpenFlow Controllers Ecosystem

Currently, the literature notices the availability of a number of OpenFlow controllers. In essence, they differentiate each other by programming paradigms or focusing on applications. For instance, Nox [21], [22] was the first designed OpenFlow controller deployed based on C/C++ programming language. Java based version of controllers were also created such as Beacon [23] and Maestro [24]. POX [22] is a Python based version of the controller, which offers a simplified programming interface enabling rapid deployment of new network applications. Ryu [25] is also based on Python and supports OpenFlow versions 1.0, 1.2 and 1.3. Trema [26] is a controller based on Ruby and FlowER is an Erlang based Openflow Controller [27].

These controllers came with sample code that shows how to create new network applications by using each controller proposed approach. A classical example of such applications is a *Learning Switch*. Usually, these applications offer low-level services and the network developer is responsible to build new ones according to particular requirements. While they are suitable as a starting point to use SDN concept, these controllers are not enough to achieve reliability and performance carrier grade networks demands [28], [29]. Such demands are not related to the telecommunication capabilities of the network entities but with aspects such as high availability, scalability, high performance, reliability and resiliency. The open issues of the aforementioned OpenFlow controller guided current efforts, including FloodLight [30], Onix [31] and more recently, OpenDaylight project [32] and ONOS [33].

FloodLight was created as a fork of Beacon, where the focus is to build a commercial controller with enterprise class. The open source version does not offer resiliency or scalability such as the commercial version, called Big Network Controller [34], which is based on a clustered servers in a HA deployment and uses in its core FloodLight.

Motivated by the inability in satisfying neither reliability nor scalability, the NOX creators proposed Onix, which is a distributed system over the network control plane that offers a global view of the network and defines an API that can use this information to build new control plane services and addresses such requirements. Onix was the basis for the software offered by Nicira [35] and its approach is used by the Modern SDN Stack [36] project.

The OpenDaylight project [32] aims to create a common architecture that can be exploited by the industry to create new and innovative services which use SDN abstractions. This architecture comprises several layers, one of them being the Service Abstraction Layer (SAL) that could interact with different protocols that would be exposed by plug-ins. Another layer is the Controller Platform [32] that controls the network devices, such as routers and switches, and defines a common API that will be used by the upper layer applications. One of the objectives of OpenDaylight project is to create a carrier grade architecture [37].

Another open source controller that currently is under development is ONOS (Open Network Operation System) [33], which aims at providing an architecture focused on fault tolerance and distribution of the state in various controllers, and providing a graphical high-level abstraction of the network status. According to ON.LAB, these features make ONOS a good alternative for service providers and also large WAN operators. A prototype was presented by OnLab at ONS 2013 and also at 2014 indicating that ONOS is still on the way to reach its goals [38].

Big Network Controller and also Onix are not available as open source and thus the SDN community is not able to run experiments using these particular solutions. OpenDayLight project and also ONOS have a road ahead.

Moreover, all these SDN controllers do not offer by default an integration with other signaling approaches used by telecommunications operators, such as: Session Initiation Protocol (SIP) [39]; DIAMETER [40]; Extensible Messaging and Presence Protocol (XMPP) [41]; Media Gateway Control Protocol (MGCP) [42]; among others.

This scenario indicates some open issues and this work contributes to bridge this gap.

## III. Carrier Grade OpenFlow Controller

The network infrastructure itself has no value. The value is in the applications and services that can be created on top of this infrastructure. SDN abstractions enable the deployment of new and innovative services and even completely new network architectures [43]. However, these abstractions are being deployed at this moment and the SDN community is still in pursuit of a carrier grade platform.

The work presented here is deployed on top of the JAIN [13]. The JAIN is a set of APIs [44] dedicated to creating voice and data convergent services. The goal of these APIs is to abstract the underlying network, so those services can be developed independently of network technology. This approach couples with SDN abstractions.

The JAIN SLEE [14] defines a component model that supports event driven applications suitable for carrier-grade environment concerned with requirements such as high throughput, low latency, scalability [15] and availability [17]. Currently, several telecom operators have services deployed by using JAIN SLEE. JAIN SLEE is available as commercial products,

 ISBN: 978-1-61208-360-5 

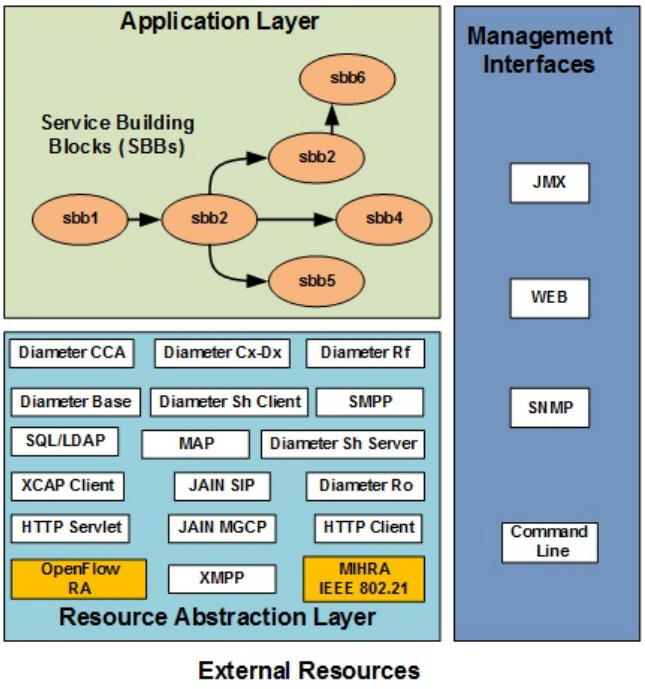


Figure 1: JAIN SLEE Architecture Main Components.

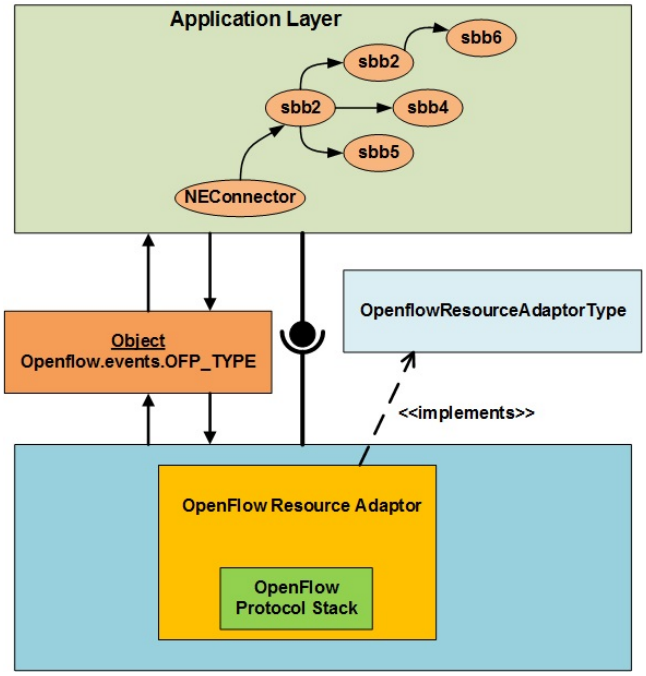


Figure 2: Openflow Resource Adaptor.

such as Rhino [18] and also open source, such as the Mobicents platform [19].

The main components of JAIN SLEE are presented in Fig. 1. The component model consists of two different layers: the application layer that represents the services which run at the JAIN SLEE Application Server and the Resource Adaptation Layer which abstracts underlaying protocol stacks and adapts them to the JAIN SLEE model. The Service Build Block (SBB) contains the application and service logic. Each SBB can be composed of one or more children SBBs and they are organized as a graph. A Service is a deployed and managed artifact which specifies its root SBB and the default event delivery priority. A registered SBB is able to capture and fire events. An Activity is a related stream of events, such as a phone call, that are captured by SBBs entities. The state of these entities can be replicated in a clustered deployment.

According to the JAIN SLEE specification, a *Resource* represents a system that is external to the JAIN SLEE. The Resource Adaptation layer (depicted in Fig. 1) enables several control plane protocols, currently used at the telecommunication protocol stack, to plug in at the JAIN SLEE component model, thus fostering the development of new services and applications that can exploit SDN benefits.

By using clustering, JAIN SLEE supports high availability and fault tolerance. The fault tolerance mode works with state replication and thus a cluster can be viewed as only one virtual container which encompasses all nodes that are active in that cluster. This way, all activity context and SBB entities data are replicated across the cluster nodes. While all the internal components of the JAIN SLEE are fully fault tolerant, the resource adaptors at border with the JAIN SLEE container and the outside environment are not replicated by default, but they can be created to be cluster-aware by using the Fault Tolerant Resource Adaptor API.

Considering all these features, JAIN SLEE is a platform suitable to handle carrier-grade throughput, latency and fault tolerance requirements over a general purpose IT infrastructure [16].

### A. *OpenFlow Resource Adaptor*

The OpenFlow protocol and a controller are resources to the JAIN SLEE. To interact with its component model, these resources need to be adapted to its component model, what is accomplished by a *Resource Adaptor* (RA). A RA receives messages from this external system by using a protocol and submits them as events that are produced inside the RA. The RA may, also, consume events created by the services running inside the JAIN SLEE.

The JAIN SLEE specification defines a *Resource Adaptor Type* that basically consists of a set of interfaces that represents common characteristics that must be implemented by a RA of that type. Moreover, the *Resource Adaptor type* references the Events that a *Resource Adaptor* will produce and consume. By using this approach, different resources can be plugged into the JAIN SLEE components.

Usually, resources such as SIP, DIAMETER, XMPP and MGCP protocol stacks have a RA already defined [19]. Regarding OpenFlow there is no *Resource Adaptor* already defined, being this definition one of the contributions of this work. The *OpenFlow Resource Adaptor* (OpenFlowRA), presented in Fig. 2 is responsible for the interaction with the services that run inside JAIN SLEE. The events are captured by the SBB entities according to the service configuration.

The OpenFlowRA implements an *OpenFlowResourceAdaptorType*. This resource type references all the

 ISBN: 978-1-61208-360-5 

events that might be fired from the OpenFlow stack. In this case, to each message type defined within the *enum ofp_type* at the OpenFlow 1.0 specification [5] an Event that will be sent and also received from the JAIN SLEE was defined.

When the OpenflowRA is actived by the JAIN SLEE, it starts the FloodLight Controller [30]. After this point, all OpenFlow Messages are converted into Events that are sent to the JAIN SLEE. In the same manner the services that run inside JAIN SLEE can dispatch events to the OpenFlowRA that corresponds to OpenFlow messages such as OFPT_FLOW_MOD and OFPT_PACKET_OUT that will be received the OpenFlowRA and then will be forwarded to an OpenFlow Switch by the Controller as OpenFLow Messages.

The implementation uses MessageEvent, a JAIN SLEE feature. Each message received by OpenFlowRA is converted to a JAIN SLEE event. For instance, when a OFPT_PACKET_IN is received by OpenflowRA, it is converted into a MessageEvent of type PacketIn, and it is sent to Abstraction Layer. After be treated by RA, all messages are sent to the Abstraction Layer.

To decouple the services (SBBs) from the signaling control, a default SBB named NEConnector is created. This SBB will be responsible to retrieve the OpenFlow related events, inspect them and fire the corresponding events related to the service that is being created inside a set of SBBs. By using this approach, the NEConnector is the only SBB that needs to care of OpenFlow events. This design enhances further compatibility with new OpenFlow protocol versions, such as OpenFlow 1.3, thus contributing to a low coupling between OpenFlow protocol and other SBBs. The NEConnector abstraction allows the architecture to support several protocols but, if necessary, other SBB can be created to handle other requirements, thus providing a flexible architecture that supports different signaling protocols from the telecommunications world and also from the computer networks world.

## IV. Case Study

This case study highlights how the abstraction proposed by this work can be used to integrate OpenFlow with other infrastructure control protocols enabling the deployment of new services and architectures.

The proposed approach was used to integrate OpenFlow with the MIH protocol [20]. The main purpose of the IEEE 802.21 standard for MIH is to facilitate and optimize inter technology handover processes by providing a set of primitives for obtaining link information and controlling link behavior.

The IEEE 802.21 standard offers an abstraction of the control of wireless access links and in this case, an IEEE 802.21 resource adaptor was also created. The extensible component model defined by JAIN SLEE, enabled the integration with this protocol by defining a new resource adaptor. Considering that this RA was not available before, the RA built during this work is also an important contribution to the community that builds JAIN SLEE based services.

The approach envisaged by the OpenFlow was also applied here. Thus the NEConnector is responsible to receive the events generated and route them to the corresponding SBB which is interested in this particular event.

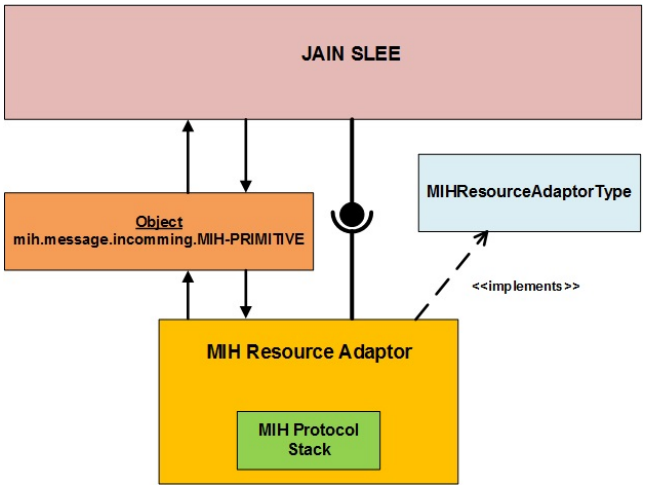


Figure 3: IEEE 802.21 MIH Resource Adaptor.

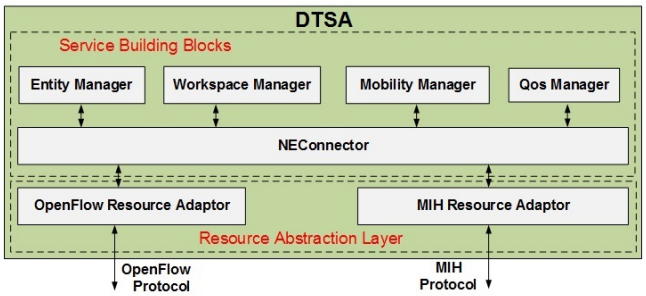


Figure 4: DTSA Components based on JAIN SLEE Component Model.

The MIHRA, as depicted in Fig. 3, implements an *MIHResourceAdptorType*. This resource type references all the events that might be fired from the MIH stack. Thus, there is one different message type to each service primitive defined in the IEEE 802.21 specification.

Moreover, by using the approach proposed in this work and bringing together OpenFlow and also the MIH resource adaptors, it was possible to create the main component of a new network architecture, named Entity Title Architecture (ETArch).

ETArch [45] is a clean slate network architecture, where naming and addressing schemes are based on a topology-independent designation that uniquely identifies an entity. This designation is named Title. ETArch also defines a channel that gathers multiple communication entities. This channel is called Workspace. A key component of this architecture is the Domain Title Service (DTS), which deals with all control aspects of the network. The DTS is composed of Domain Title Service Agents (DTSAs), which maintain information about entities registered in the domain and the workspaces that they are subscribed to, aiming to configure the network devices to implement the workspaces and to allow data to reach every subscribed entity.

The DTSA was created, deployed and tested using the Mobicents JAIN SLEE. Fig. 4 presents the overall DTSA architecture based on the JAIN SLEE component model pre-

 ISBN: 978-1-61208-360-5

sented in this work. The NEConnector decouples the DTSA SBBs from the control protocols of the infrastructure, then, the services expressed by the SBBs can also interact with the infrastructure and adapt it, in this case do create the workspace concept on top of a OpenFlow enabled infrastructure and also to support seamless entity mobility by optimizing handover using the IEEE 802.21 MIH protocol. Fig. 4 also presents the SBBs that were created to implement the DTSA, each one dealing with the main aspects of the architecture: the control of entities (EntityManager); the workspaces supported by the DTSA in a given moment (WorkspaceManager); mobility procedures of entities (MobilityManager); and, the QoS enforcement (QoSManager) inside the workspace in order to meet specific communication requirements.

## V. Concluding Remarks and Future Work

SDN abstractions can enable new services and applications. Considering OpenFlow, the current materialization of some SDN concepts, the controller is a central piece of the architecture.

The currently available open source controllers are not suitable to meet carrier grade requirements. This kind of controller is a work in progress being conducted by the SDN community through some projects.

This work proposes a new SDN controller architecture, which is integrated with a carrier grade service level execution environment, based on the JAIN SLEE specification. Such controller can abstract several different protocol stacks and provides a common component model where new services and applications could be deployed. JAIN SLEE current implementations are adopted and being used, at plant floor ground, by several telecom operators.

To foster this integration, this work created an OpenFlow resource adapter which enables a cross layer approach where services are able to control the network infrastructure during run-time, by using the OpenFlow protocol.

To showcase the approach presented in this work, an IEEE 802.21 MIH resource adapter was also constructed. The MIH protocol purpose is to abstract the control of wireless access network infrastructure, thus enabling services that need to handle mobility requirements.

By putting together OpenFlow and MIH protocols, the presented case study uses JAIN SLEE SDN capable control layer to deploy a clean slate network architecture named ETArch. In this case, the adopted component model enabled the evolution of the network architecture, enabling new services to be gradually deployed and tested on top of it.

The resource adapters presented here are publicly available, thus collaborating with the research which aims to define, design and deploy next generation computer network architectures.

The carrier grade approach proposed in this work is aligned with most current trends regarding a SDN control layer,but in addition to other proposals, it enables an integration of the protocols that control the network hardware, such as OpenFlow, with the ones the control the applications, thus enabling new types of network services. Moreover, the extensible model can accommodate new protocols and future initiatives, thus preserving investments and becoming an interesting outcome that can be exploited by telecom operators.

As future work, the proposed and constructed carrier grade SDN control layer will be tested under different scenarios in order to demonstrate its fault tolerant and scalability. An OpenFlow 1.3 compliant RA will also be deployed and plugged into the architecture.

The innovative approach proposed under this work presents to the research community a SDN control layer that is suitable to meet carrier grade requirements and is a viable alternative to bring SDN into the telecom infrastructure.

## Acknowledgment

This work has been partially funded by the European Community's Seventh Framework Programme, under grant agreement n. 258365 (OFELIA project), by the Brazilian agencies: CAPES, CNPq and FAPEMIG and also by PROPP/UFU.

## References

[1] Open Networking Foundation. Software-defined networking: The new norm for networks. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf [retrieved: May, 2014]

[2] G. Goth, "Software-Defined networking could shake up more than packets," IEEE Internet Computing, vol. 15, no. 4, Aug. 2011, pp. 6–9.

[3] M. Palmer. SDN market size to exceed $35b in 2018. [Online]. Available: http://www.sdncentral.com/sdn-blog/sdn-market-sizing/2013/04/ [retrieved: May, 2014]

[4] PLEXXI, LIGHTSPEED, and SDNCentral. SDN market sizing. [Online]. Available: http://cdn.sdncentral.com/wp-content/uploads/2013/04/sdn-market-sizing-report-0413.pdf [retrieved: May, 2014]

[5] N. McKeown et al., "OpenFlow: enabling innovation in campus networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, Mar. 2008, p. 6974, ACM ID: 1355746.

[6] FIBRE. FIBRE Project - Future Internet Testbeds Experimentation Between Brazil and Europe. [Online]. Available: http://www.fibre-ict.eu/ [retrieved: May, 2014]

[7] GENI. OpenFlow - GENI. [Online]. Available: http://groups.geni.net/geni/wiki/OpenFlow [retrieved: May, 2014]

[8] OFELIA. OpenFlow in europe - linking infrastructure and applications. [Online]. Available: http://www.fp7-ofelia.eu/about-ofelia/ [retrieved: May, 2014]

[9] ALIEN. FP7 ALIEN project. [Online]. Available: http://www.fp7-alien.eu/ [retrieved: May, 2014]

[10] S. Sezer et al., "Are we ready for SDN? implementation challenges for software-defined networks," IEEE Communications Magazine, vol. 51, no. 7, Jul. 2013, pp. 36–43.

[11] I. T. UNION, The carrier grade open environment reference model, ser. SERIES Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks. International Telecommunication Union, Dec. 2006. [Online]. Available: http://www.itu.int/rec/T-REC-Y.2901-200612-I/en

[12] TeleStax. TeleStax open source cloud communications - success stories. [Online]. Available: http://www.telestax.com/case-studies/ [retrieved: May, 2014]

[13] ORACLE. JAIN general Q&A. [Online]. Available: http://www.oracle.com/technetwork/java/qa-137977.html [retrieved: May, 2014]

[14] D. Ferry. JAIN SLEE (JSLEE) 1.1 specification, final release. [Online]. Available: http://www.jcp.org/en/jsr/detail?id=240 [retrieved: May, 2014]

[15] M. Femminella et al., "Scalability and performance evaluation of a JAIN SLEE-based platform for VoIP services," in Teletraffic Congress, 2009. ITC 21 2009. 21st International, 2009, pp. 1–8.

 ISBN: 978-1-61208-360-5 

[16] M. Gomez, E. Torres, J. Chamorro, T. Hernandez, and E. Mendez, "On the integration and convergence of IN and IP mobile service infrastructures," in International Conference on Telecommunications, 2009. ICT '09, May 2009, pp. 143–148.

[17] M. Femminella, R. Francescangeli, E. Maccherani, and L. Monacelli, "Implementation and performance analysis of advanced IT services based on open source JAIN SLEE," in 2011 IEEE 36th Conference on Local Computer Networks (LCN), Oct. 2011, pp. 746–753.

[18] OpenCloud. Rhino SLEE carrier grade system. [Online]. Available: http://www.opencloud.com/products/rhino-application-server/carrier-grade/ [retrieved: May, 2014]

[19] MOBICENTS. Mobicents JAIN SLEE. http://www.mobicents.org/slee/intro.html. [Online]. Available: http://www.mobicents.org/slee/intro.html [retrieved: May, 2014]

[20] IEEE, "IEEE standard for local and metropolitan area networks- part 21: Media independent handover," IEEE Std 802.21-2008, 2009, pp. c1 –301.

[21] N. Gude et al., "NOX: towards an operating system for networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 3, Jul. 2008, p. 105110. [Online]. Available: http://doi.acm.org/10.1145/1384609.1384625

[22] NOXREPO. About NOX. [Online]. Available: http://www.noxrepo.org/nox/about-nox/ [retrieved: May, 2014]

[23] D. Erickson. Beacon. [Online]. Available: https://openflow.stanford.edu/display/Beacon/Home [retrieved: May, 2014]

[24] Z. Cai. Maestro. [Online]. Available: http://code.google.com/p/maestro-platform/ [retrieved: May, 2014]

[25] NTT Communications. Ryu SDN framework. [Online]. Available: http://osrg.github.io/ryu/ [retrieved: May, 2014]

[26] H. Shimonishi, Y. Chiba, Y. Takamiya, and K. Sugyo. Trema repository. [Online]. Available: http://trema.github.io/trema/ [retrieved: May, 2014]

[27] Travelping. FlowER - erlang OpenFlow development platform. [Online]. Available: http://travelping.github.io/flower/ [retrieved: May, 2014]

[28] D. Staessens, S. Sharma, D. Colle, M. Pickavet, and P. Demeester, "Software defined networking: Meeting carrier grade requirements," in 2011 18th IEEE Workshop on Local Metropolitan Area Networks (LANMAN), 2011, pp. 1–6.

[29] F. Tam, "On engineering standards based carrier grade platforms," in Proceedings of the 2007 workshop on Engineering fault tolerant systems, ser. EFTS '07. New York, NY, USA: ACM, 2007. [Online]. Available: http://doi.acm.org/10.1145/1316550.1316554

[30] Big Switch Networks. Floodlight OpenFlow controller. http://floodlight.openflowhub.org/. [Online]. Available: http://floodlight.openflowhub.org/ [retrieved: May, 2014]

[31] T. Koponen et al., "Onix: a distributed control platform for large-scale production networks," in Proceedings of the 9th USENIX conference on Operating systems design and implementation, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, p. 16. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924968

[32] OpenDaylight. OpenDaylight technical overview. [Online]. Available: http://www.opendaylight.org/project/technical-overview [retrieved: May, 2014]

[33] ON.LAB. ONOS - open network operating system. [Online]. Available: http://tools.onlab.us/onos.html [retrieved: May, 2014]

[34] B. S. Networks. Big network controller. [Online]. Available: http://www.bigswitch.com/products/SDN-Controller [retrieved: May, 2014]

[35] NICIRA. Nicira. [Online]. Available: http://nicira.com/ [retrieved: May, 2014]

[36] O. Research. Modern SDN stack project. [Online]. Available: http://onrc.stanford.edu/research_modern_sdn_stack.html [retrieved: May, 2014]

[37] C. Matsumoto. What OpenDaylight really wants to do. [Online]. Available: http://www.lightreading.com/blog/software-defined-networking/what-opendaylight-really-wants-to-do/240152993 [retrieved: Apr., 2014]

[38] ON.LAB. ONOS at ONS 2014. [Online]. Available: http://www.slideshare.net/ON_LAB/onos-at-ons-2014 [retrieved: Mar., 2014]

[39] E. Schooler et al. SIP: session initiation protocol. [Online]. Available: http://tools.ietf.org/html/rfc3261 [retrieved: May, 2014]

[40] J. Arkko, E. Guttman, P. R. Calhoun, and J. Loughney. Diameter base protocol. [Online]. Available: http://tools.ietf.org/html/rfc3588 [retrieved: May, 2014]

[41] P. Saint-Andre. Extensible messaging and presence protocol (XMPP): core. [Online]. Available: http://tools.ietf.org/html/rfc6120 [retrieved: May, 2014]

[42] B. Foster and F. Andreasen. Media gateway control protocol (MGCP) version 1.0. [Online]. Available: http://tools.ietf.org/html/rfc3435 [retrieved: May, 2014]

[43] F. de Oliveira Silva, J. de Souza Pereira, P. Rosa, and S. Kofuji, "Enabling future internet architecture research and experimentation by using software defined networking," in 2012 European Workshop on Software Defined Networking (EWSDN), 2012, pp. 73–78.

[44] ORACLE. JAIN API specifications. [Online]. Available: http://www.oracle.com/technetwork/java/api-specs-137688.html [retrieved: May, 2014]

[45] C. Guimarães et al., "IEEE 802.21-enabled entity title architecture for handover optimization," in IEEE WCNC'14 Track 3 (Mobile and Wireless Networks) (IEEE WCNC'14 Track 3 : NET), Istanbul, Turkey, Apr. 2014, unpublished article. [Online]. Available: http://www.facom.ufu.br/~flavio/wcnc-2014/IEEE_802.21-enabled_ETArch_for_Handover_Optimization.pdf

 ISBN: 978-1-61208-360-5 