HetNet Cloud: Leveraging SDN & Cloud Computing for Wireless Access Virtualization

M. M. Rahman
Dept. of Electrical Engineering
ETS, University of Quebec
Montreal, Canada
Email: mohammad-moshiur.rahman.1@ens.etsmtl.ca

Charles Despins
Prompt Inc.
Montreal, Canada
Email: CDespins@promptinc.org

Sofiène Affes
INRS-EMT
University of Quebec
Montreal, Canada
Email: affes@emt.inrs.ca

***Abstract*—Cellular network traffic is growing at an exponential rate and it is predicted that the novel services of future 5G networks will demand even more data rate with varying quality of service (QoS). In this context, network operators and vendors are opting for network virtualization that will facilitate novel service provisioning while ensuring efficient resource utilization. We present a software defined networking (SDN) based cloud architecture, named HetNet Cloud, for implementing virtual heterogeneous wireless networks. We propose to utilize the spare bits of the OpenFlow packet model to facilitate implementation of virtual networks.To showcase the effectiveness of HetNet cloud model in managing virtual wireless networks, interference management and traffic offloading between two virtual heterogeneous wireless networks have been demonstrated.**

I. Introduction

Virtualization of wireless networks is getting traction in recent time [1]. Due to its ability to ensure efficient resource utilization by facilitating sharing of a common subset of network resources, major telecommunication vendors and operators are opting for implementing network function virtualization (NFV) [2]. With the advent of smart phones and tabs, new user applications and services are emerging that demand high data rate as well high quality of service (QoS). Catering to such need with limited radio spectrum and vendor locked-in devices is not feasible. To alleviate this problem, the way forward is to re-architect the network architecture, so that, it offers more flexibility [3] in provisioning new services as well as ensures efficient network resource utilization. We deem SDN and cloud computing as two enabling technologies for implementing virtual wireless networks.

SDN is a disruptive technology that has the unique ability to separate the network control from the data plane and providing absolute control over the network in the form of programming the network. SDN has been proposed to use for WiFi/WiMAX networks [4], [5]; for LTE core and access networks [6], [7], etc. In cloud computing, network resources are pooled in centralized locations and multiple tenant can share the resources according to some pre-defined service level agreement (SLA). Different cloud architectures have been proposed for cellular core [8], [9] and access [10] networks. In [11], SDN is proposed to implement a heterogeneous cloud radio access network (HCRAN) system but specific technical detail on controller platform, as well as virtualization technology to be used, is not discussed. The above mentioned proposals focus on different aspects of wireless network virtualization, but a comprehensive solution to virtualize both the wireless core and access network is absent in open literature. In this respect, we present a detailed layered architecture of the HetNet Cloud model for virtualizing both the core and access networks, emphasizing the use of open northbound application programming interface (API) to facilitate programming virtualized wireless heterogeneous networks (HetNet). The rest of the article develops as follows, in Section II detailed architectural description of the HetNet Cloud model is presented, Section III describes interference management and traffic offloading mechanisms in a HetNet Cloud model, finally concluding remarks and future work are presented in Section IV.

II. A generic HetNet Cloud Architecture

The HetNet Cloud is composed of distributed wireless data centers (WDCs), where baseband processing takes place and radio access to the users is provisioned by fiber-fed remote radio heads (RRHs) that span from the WDCs. The novelty of this architecture is that, it is a SDN-based implementation of both core and access networks which comprises of fiber-fed multiple radio access technologies (multi-RAT) enabled RRHs. In this model, the virtual network operators (VNOs) lease network resources (i.e., compute, storage, networking and radio resources) from the infrastructure providers (InPs) and build their own customized network by *programming* the underlying network fabric. Fig. 1 shows a schematic diagram for the HetNet Cloud model. Different layers of the system model is described below following a top-down approach.

A. Application layer

In a HetNet Cloud architecture, different network nodes (e.g. switching gateway (SGW), packet data network gateway (PGW), mobility management entity (MME), etc.) are implemented as software instances. Also network control functions are written as applications by the individual VNOs. These applications dictate the end-to-end network connectivity by writing the forwarding behaviour of the underlying network substrate which consists of switches and radio front ends.

978-1-4673-6555-0/15/$31.00 ©2015 IEEE

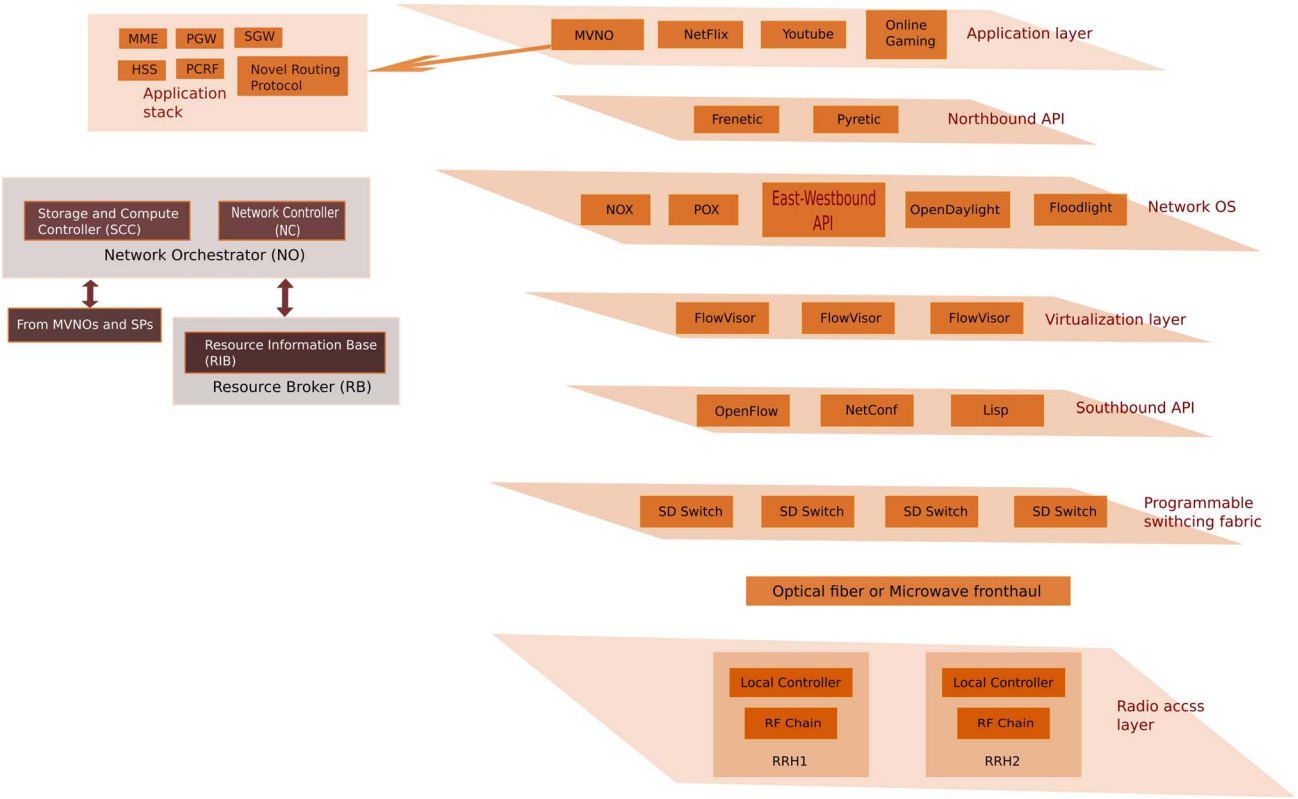

Fig. 1: Schematic of a HetNet Cloud.

B. Northbound API

A VNO needs multiple applications to work synchronously for an end-to-end service provisioning, these applications include routing, load-balancing, firewall, deep packet inspection (DPI), etc. To write a single monolithic controller application, using match-action programming semantic [12] of network operating systems (POX [13], OpenDaylight [14], etc.), for performing all of the above mentioned tasks is technically daunting. There is very high probability of programming bugs and there is certain situation where the functioning of one task overlaps or collides with another one. To alleviate this problem a high level abstraction for programming the underlying software defined network fabric is necessary [15]. Pyretic [16] is such a domain specific programming language (based on Python programming language) with efficient runtime system that enables to develop sophisticated network applications at much ease. It is also possible to write modular network programs and eventually *compose* them together *sequentially or parallelly* [16] according to their mutual dependence.

C. Network OS (NOS) & East-Westbound API

A well defined abstraction of the underlying network substrate expedites innovation in a SDN echo-system. A NOS abstracts the global view of the network and allows a network programmer to write different control applications as centralized programs. Also for *horizontal* communication among the controllers, it is necessary to design a east-westbound API.

D. Virtualization layer

Virtualization allows multiple VNOs to share a common subset of network resources where all the VNOs are isolated from each other. In the HetNet cloud architecture, flow-level virtualization [17] is used. The virtualizer acts as a transparent proxy sitting between the network controller and the southbound API and enforces isolation between slices by rewriting policies, dropping conflicting rules, etc.

E. Southbound API

A southbound API takes instructions from th controller platform and dynamically programs the underlying network fabric. The de-facto southbound API is OpenFlow [12], that is used extensively by both academia and industry alike. Other popular southbound APIs are NetConf [18], LISP [19], etc.

F. InP's resource management layer

The resource management layer of an InP keeps track of the usage of physical and virtual resources. It consists of a network orchestrator (NO) and a resource broker (RB).

1) Network Orchestrator (NO): A NO manages the computing, storage and networking resources shared by different VNOs. It consists of two sub-modules:

Network controller (NC): It is responsible for provisioning and management of virtual network nodes to VNOs.

Storage & compute controller (SCC): It is in-charge of assigning and subsequent management of storage and computing resources to various VNOs.

2) Resource broker (RB): It is the central resource information base for the InP. It manages the usage status of resources (compute, storage & networking), so that, the NC and SCC can have a global usage view of VNOs and manage them efficiently.

G. Baseband processing

Multiple radio access technologies (RATs) are supported in the HetNet Cloud model, VNOs can provision either cellular (LTE, WiMAX, 3G, etc.) networks, WiFi ISP, or device-to-device (D2D) communication, e.g., smart grids or even sensor network services. To facilitate such diverse RATs, radio processing chains are decomposed into different PHY and MAC layer processing blocks, so that, a VNO can choose the blocks required for its service and provision its customized network. It is to be noted that carrying out all the baseband processing is a WDC might not be optimal for traffic of all QoS classes as some might have very tight requirement of processing delay. Hence the delay-sensitive traffic (voice, live video, etc.) should be processed at the RRHs capable of baseband processing, while more delay tolerant traffic might be pushed to the WDC for processing. It is worth noting that, the length of fiber optic cable from the WDC to the RRH is a very important design consideration [20].

H. Radio access plane

Radio access plane consists of distributed fiber-fed RRHs across a service area. The high volume of PHY layer processing signals justifies the use of optical fiber front-haul. Due to the varying nature of wireless environment, dynamic provisioning of radio resources, strict QoS management and handling user mobility requires frequent exchange of control information between the controller and the underlying network substrate. For a centralized control architecture this might pose as a serious bottle-neck. Hence, local controllers are needed to be installed at RRHs to handle frequent local events, like: user mobility between neighboring cells, transmission power management, dynamic frequency allocation, etc.

III. Using northbound API to facilitate virtual wireless network management

Domain specific programming language like Pyretic [16] (built on top of POX [13] controller platform) make building modular network programs an ease. We propose to use northbound API like Pyretic to build modular applications for virtual wireless networks. The parallel and sequential composition operators of the language makes it possible to compose complicated network applications by composing (in parallel or in series) more simpler applications. Using the abstract packet model in Pyretic, OpenFlow [12] packet header fields can be extended to include virtual fields, that can be used to associate packets with high level meta data. We propose to extend the abstract packet model in Pyretic to implement virtual wireless networks through *abstract topology*. The spare bits (i.e., VLAN, MPLS fields) in an OpenFlow packet are used for specifying virtual networks (VNO id), virtual network node (VSW id) and radio spectrum (RS id) to be used for transmitting a particular packet. One such packet model is shown in Fig. 2.

A. Interference management

Interference mitigation in wireless environment is a critical challenging issue. Interference management application can be implemented as a dynamic policy that changes the network behavior dynamically depending on network state. If a UE experiences an interference level above the acceptable threshold, the controller can get this information from the channel quality information (CQI) fed-back from the UE. Upon receiving this information, the controller can adjust transmission parameters (e.g., transmission power, DL frequency) to alleviate the problem. For example, if the interfering base station (BS) belongs to the same operator, it is very convenient for the operator to change the DL frequency for the interrupted UE as it has a global view of the network. In case, the interfering BS belongs to a different VNO sharing the network resources from the same InP, the operator can exchange the interference information relatively quicker (through high speed network interconnect of the wireless data center) to resolve the issue. This will reduce service degradation and increase the quality of experience (QoE) for the UEs.

Fig. 3 shows an example of message exchanges between two local controllers for managing interference to a UE located at the cell border of RRH1 and RRH2 belonging to the same VNO. From the CQI sent by the UE, the Local Controller1 (LC1) located at RRH1 (cf. Fig. 1) identifies that the UE is experiencing interference in the DL direction from the neighboring RRH2. To resolve the interference problem, LC1 requests (using the radio link denoted by the dotted line) Local Controller2 (LC2) situated in RRH2 to decrease its transmission power in the DL direction. RRH2 responds positively by lowering its DL transmission power. LC1 again checks the CQI from the UE and observes that it has not yet improved above the threshold level. According to its *action logic sequence (specified during the programming phase)* it notifies the Central Controller (CC), located at the wireless data-center, about the interference and requests a change in DL frequency for the particular UE. Having a global view of the whole network the CC selects a DL frequency for the RRH1 that will not interfere with the used frequency of any of its neighbors. RRH1 changes the DL frequency for the UE and it continues communication with satisfactory QoE. Cells (RRHs) of different VNOs can also communicate among themselves to resolve such interference issue.

B. Traffic offloading in a HetNet eco-system

Now a days users have access to multiple networks (cellular, fixed) at the same time. Also mobile devices (smart phones, tablets, etc.) run a plethora of applications having different quality of service (QoS) requirements. Through a collaborative effort from both the VNO and UEs, delay-tolerant, high data rate demanding user applications can be offloaded to WiFi networks while more delay-sensitive traffic can be provisioned

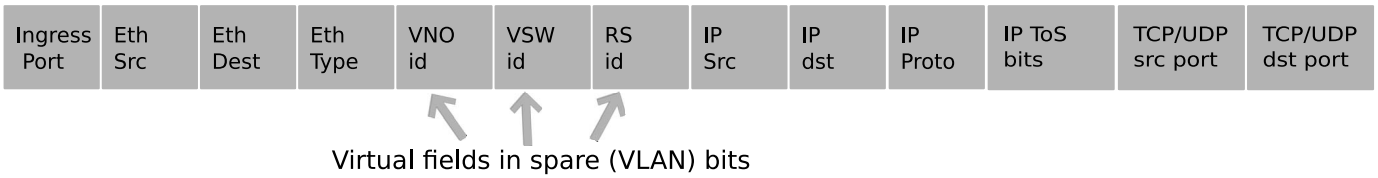

Fig. 2: OpenFlow modified packet model with virtual fields for virtualization.

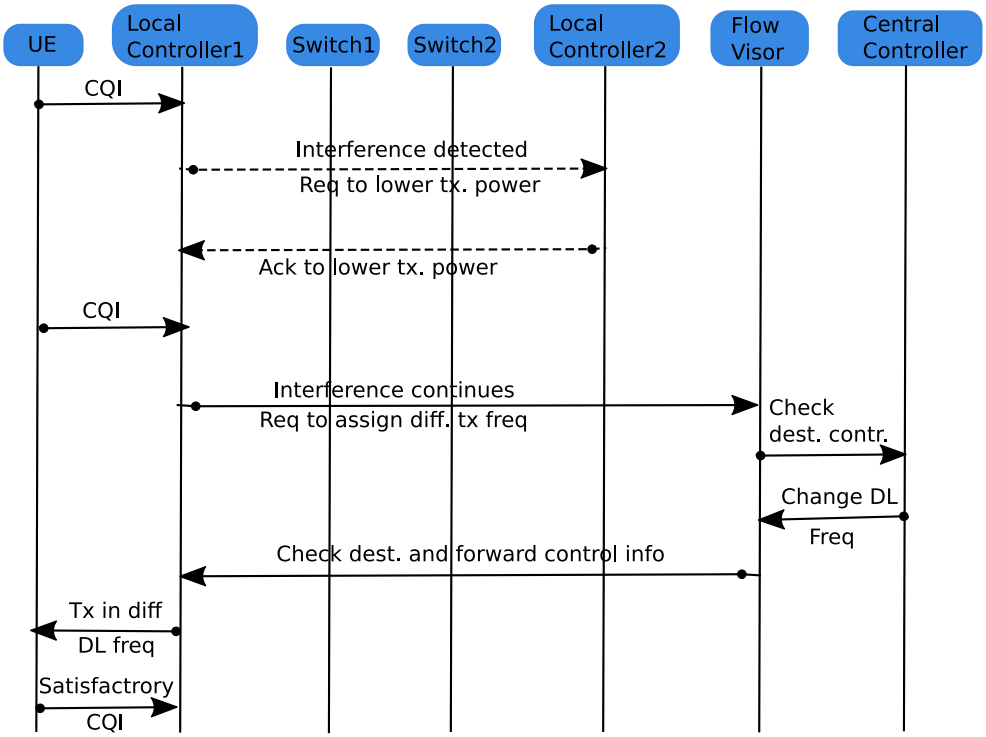

Fig. 3: Interference management message exchange between Local Controllers belonging to the same VNO.

by the cellular network. Using the high level abstraction provided by Pyretic [16], traffic routing of users can be controlled per user application granularity. As an example, for traffic offloading in a heterogeneous network, lets consider the message exchange diagram in Fig. 4. A UE served by VNO-A (a cellular network) comes in the coverage range of a fixed (WiFi) network, VNO-B. In real life, the situation is similar to, when a cellular user enters a shopping mall or coffee shop that has public Hot-Spot (either free or paid). The LC-A of VNO-A receives traffic from the UE and from the UDP port number (e.g., 8011) of the packet it immediately identifies that the UE is streaming video traffic which is a lower priority traffic class. Hence, LC-A sends request to the CC-A to initiate offloading of UE's video traffic to VNO-B (WiFi Hot-Spot). The control message is intercepted transparently by the FlowVisor (cf. Fig. 1) which directs the message to CC-A, recognizing that it is the correct destination from the virtual header fields. CC-A sends a handover request to CC-B for traffic offloading, eventually after positive acknowledgement from the CC-B, the video traffic is offloaded to VNO-B.

A service differentiation use case of the HetNet cloud model was illustrated by the authors of this paper in [21], where traffic offloading and load balancing between two virtual wireless networks were emulated in Mininet [22] emulation platform. Pyretic [16] was used to write the applications for virtualization, offloading and load balancing and subsequently compose them together.

IV. Conclusion

We have presented detailed network architecture of a novel SDN-based cloud-enabled virtualization platform for implementing virtual heterogeneous wireless networks. We have proposed to use the spare bits of the OpenFlow packet model to identify virtual network entities, i.e., VNO id, VSW id and RS id. To emphasize the benefit of northbound API in virtual network management, interference handling and traffic offloading between virtual heterogeneous wireless networks have been shown. In our ongoing endeavour, we are investigating the architectural challenges and virtual network function (VNF) placement algorithms in the context of the HetNet Cloud.

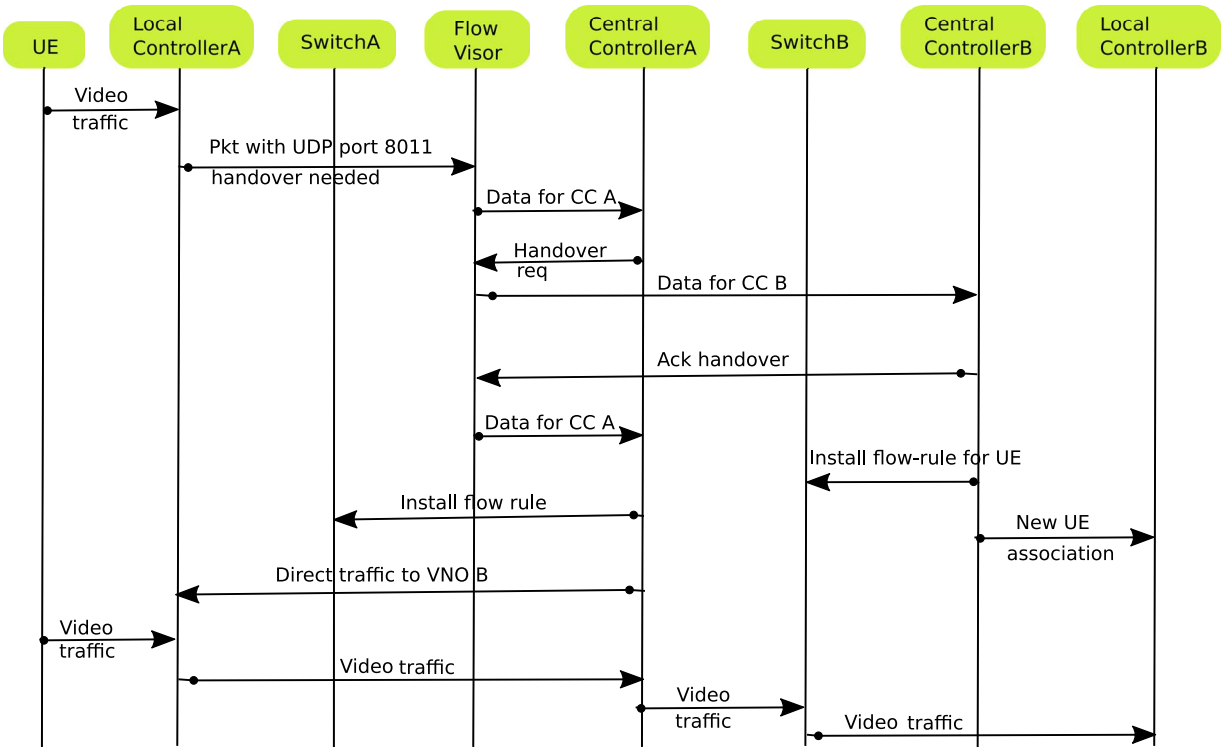

Fig. 4: Message exchanges for traffic offloading between VNOs.

REFERENCES

[1] X. C.-Perez, J. Swetina, T. Guo, R. Mahindra, and S. Rangarajan, "Radio access network virtualization for future mobile carrier networks," *IEEE Communications Magazine*, vol. 51, no. 7, 2013.

[2] M. Chiosi, D. Clarke, P. Willis, and et al., "Network Function Virtualization: An Introduction, Benefits, Enablers, Challenges & Call for Action," in *SDN and OpenFlow World Congress*, Oct. 2012.

[3] T. Koponen, S. Shenker, H. Balakrishnan, N. Feamster, I. Ganichev, and et al., "Architecting for innovation," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 3, 2011.

[4] K.-K. Yap, M. Kobayashi, R. Sherwood, T.-Y. Huang, M. Chan, N. Handigol, and N. McKeown, "OpenRoads: Empowering Research in Mobile Networks," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, 2010.

[5] L. Suresh, J. S.-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards Programmable Enterprise WLANs with Odin," in *proc. of the HotSDN*, Aug. 2012.

[6] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "SoftRAN: Software Defined Radio Access Network," in *proc. of the second ACM SIGCOMM workshop on HOt topics in software defined networking, HotSDN*, 2013.

[7] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "SoftCell: Taking Control of Cellular Core Networks," *rXiv preprint arXiv:1305:3568*, 2013.

[8] J. Kempf, B. Johansson, S. Pettersson, H. Luning, and T. Nilsson, "Moving the Mobile Evolved Packet Core to the Cloud," in *proc. of the Fifth International Workshop on Selected Topics in Mobile and Wireless Computing*, 2012.

[9] T. Taleb, M. Corici, C. Parada, and et al., "EASE: EPC as a Service to Ease Mobile Core Network Deployment over Cloud," *IEEE Network*, vol. 29, no. 2, 2015.

[10] "iJOIN project, revised definition of requirements and preliminary definition of the iJOIN architecture," *INFSO-ICT-317941 iJOIN, D5.1, version 1.0*.

[11] M. Peng, Y. L. an Z. Zhao, and C. Wang, "System Architecture and Key Technologies for 5G Heterogeneous Cloud Radio Access Networks," *IEEE Network*, vol. 29, no. 2, 2015.

[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. P. J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, 2008.

[13] J. Mccauley, "POX: A Python-based OpenFlow Controller," *available at: http://www.noxrepo.org/pox/about-pox/*.

[14] "OpenDaylight project," *available at: http://www.opendaylight.org/*.

[15] N. Foster, A. Guha, M. Reitblatt, A. Story, and et al., "Languages for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 2, 2013.

[16] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker, "Modular SDN Programming with Pyretic," *;login:, The USENIX Magazine*, vol. 38, no. 5, 2013.

[17] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "FlowVisor: A Network Virtualization Layer," *OPENFLOW-TR-2009-1*, 2009.

[18] *Juniper Networks, Contrail Architecture, White Paper*, 2013.

[19] A. R. Ratal, L. Jakab, M. Portolés, and et al., "LISP-MN: Mobile Networking Through LISP," *Wireless Personal Communications*, vol. 70, no. 1, 2012.

[20] M. M. Rahman, C. Despins, and S. Affes, "Configuration cost vs. qos trade-off analysis and optimization of sdr access virtualization schemes," in *proc. of the IEEE NetSoft'15-Soft5G workshop*, Apr. 2015.

[21] M. M. Rahman, C. Despins, and S. Affes, "Service Differentiation in Software Defined Virtual Heterogeneous Wireless Networks," in *accepted for publication in proc. of the IEEE International Conference on Ubiquitous Wireless Broadband (ICUWB)*, 2015.

[22] "Mininet: An Instant Virtual Network on your Laptop (or other PC)," *available at: http://mininet.org/*.