Service Differentiation in Software Defined Virtual Heterogeneous Wireless Networks

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*—Wireless network virtualization is seen as a key component for future 5G networks. Virtualization enables flexible, on-demand, elastic network provisioning. We deem software defined networking (SDN) as a key enabling technology for virtualizing future wireless networks. In this regard, we introduce HetNet cloud, a software-defined cloud infrastructure for heterogeneous virtual wireless networks. As a use case, we implement two virtual network slices and study service differentiation in a HetNet wireless environment. We implement the emulation scenario in Mininet platform using hight level programming abstract of Pyretic northbound API. Emulation results show that, in the HetNet cloud model, virtual wireless networks are able to achieve the quality of service (QoS) requirement of carrier networks while ensuring efficient resource utilization through sharing a common subset of network infrastructure.**

I. Introduction

There has been a drastic hike in cellular network traffic in recent time and its growing at and ever-increasing rate. Also with the advent of smart phones and tablets novel services are emerging that have high QoS requirement. Mobile operators, with their limited licensed spectrum and vendor locked-in network gear are struggling to cope with such paradigm shift of the traditional voice-centric networks to a more data-centric one. In such a context, network operators and vendors all over the world, are seriously considering network function virtualization (NFV) [1], as an inevitable evolution of carrier networks, to ensure efficient resource utilization while decreasing capital and operational expenditure (CAPEX & OPEX). Virtualization is the process of abstracting network resources (both physical nodes and radio spectrum), so that, multiple isolated virtual network operators (VNOs) can have shared access to these resources to build their own customized (virtual) networks.

SDN [2] has emerged as a disruptive technology that can be greatly beneficial for implementing NFV. It should be noted that, while SDN is not a pre-requisite for NFV, it can work as an enabling technology for deploying the former. Due to its unique ability to separate network control from the data plane, SDN provides absolute control over the underlying network in the form of programming the network.

As a unified solution to NFV, we propose HetNet cloud, a infrastructure as a service (IaaS) cloud model to implement cloud-based virtual heterogeneous wireless networks. In this paper, we briefly describe the HetNet cloud model and as a proof of concept, we report on the initial emulation results for service differentiation per user basis for two virtual networks that share the same physical infrastructure.

The rest of the article is structured as follows: in Section II, a brief summary of related work is presented. HetNet cloud architecture is described in Section III. In Section IV, we present the emulation results for service differentiation for two virtual wireless networks that share the same physical infrastructure. Challenges of implementing HetNet cloud model is discussed in Section V, finally we conclude the paper in Section VI.

II. Related work

Use of SDN and cloud computing for implementing wireless networks is getting increased attention from both industry and academia alike in recent time. OpenRoads [3] is one of the first works on virtual wireless network using SDN, where multiple virtual networks running on a common switching fabric are isolated at flow level using a FlowVisor [4]. A software defined centralized control plane was proposed in SoftRAN [5], in this work, distributed base stations in a certain geographic area is abstracted as single big base station with a programmable central controller. MobileFlow [6] is a SDN based implementation of LTE core network. In this proposal, MobileFlow forwarding engines (MFFEs) incorporates different tunnelling protocols like GTP-U, GRE encapsulation/decapsulation, etc. An SDN based cellular network deployment strategy is proposed in CellSDN [7], it uses CellVisor which is an extension of FlowVisor [4] for slicing the cellular networks.

A cloud RAN (C-RAN) [8] model was proposed by China Mobile Research Institute (CMRI) that proposes partial and full centralization of baseband signal processing for RANs. Moving EPC to the cloud was proposed by Kempf et al. in [9]. A RAN as a service (RANaaS) model is analyzed by iJOIN [10] project, here RAN is implemented in a cloud infrastructure. EPC as a service (EASE) [11] proposes a cloud-based elastic mobile core network model, this article also describes different implementation models of EASE.

III. HetNet Cloud architecture

In its most generic form, a HetNet cloud architecture is composed of distributed wireless data-centers (WDCs) inter-

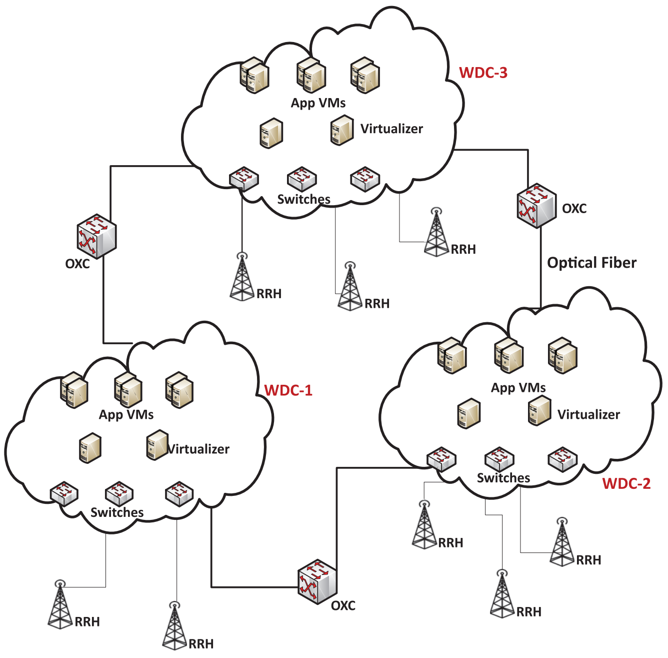

Fig. 1. A HetNet Cloud architecture consisting of distributed WDCs.

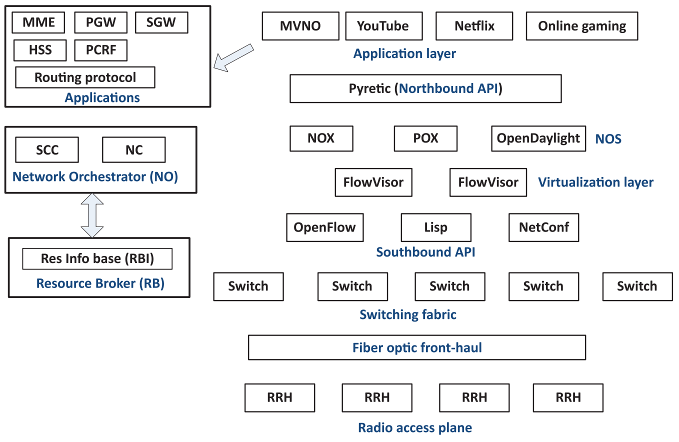

Fig. 2. Schematic of a wireless data-center (WDC) processing blocks in HetNet Cloud.

connected by high capacity optical network. In its business model, the physical and virtual infrastructure is deployed and managed by an infrastructure provider (InP) and the mobile virtual network operators (MVNOs) or service providers (SPs) lease the virtual nodes from the InP and deploy their own customized services. It is to be noted that a InP can also play the role of a MVNO or SP.

A typical HetNet cloud architecture for an urban area is presented in Figure 1, where the the WDCs are connected by a high capacity optical fiber network composed of fiber optic cables and optical cross connects (OXCs) for wavelength routing. In this form of implementation, radio access to the user equipments (UEs) are provided through optical fiber fronthaul, connecting the transmitting RRHs/APs to the WDCs. The functional blocks of a WDC appears in Figure 2, in this section, we briefly describe different parts of a WDC.

A. Application layer

A MVNO or SP (e.g. online gaming provider, YouTube, Netflix, etc.) implements its service functionalities by selecting the network processing components from the InPs either as software components in virtual machines (VMs), and/or dedicate physical hardware modules (specially for baseband processing for services requiring high QoS). Network applications (e.g. routing, load balancing, offloading, mobility management, etc.) in the application layer manages the end-to-end network connectivity by dictating the forwarding behavior of the underlying programmable switches, router and RRHs. Proper synchronization among different application modules are very important for predictable network behavior, a northbound API [12] can ensure the synchronous composition of various network applications.

B. Software modules

Different network functional and processing nodes, implemented as software modules in VMs belong to this layer. It consists of the software implementation of various EPC nodes, a.e. PGW, SGW, MME, PCRF, HSS. It also contains software modules for baseband processing, e.g. soft-eNB, soft-BS, soft-APs, etc.

C. Northbound API

For a VNO operation, different functionalities are needed for end-to-end service provisioning. For example, a routing application should program the switches to route the packets to its destination; a load-balancer should divert excess traffic to a neighboring lightly-loaded cell when a cell become overloaded; to filter malicious traffic, a firewall application is needed; specific applications are also needed for different deep packet inspection (DPI) purposes. To write a single monolithic controller application using the match-action based programming semantic of OpenFlow [13] (which is the de-facto SDN southbound API) is technically challenging and there is high possibility of coding bugs that disrupts proper network functioning. To alleviate this problem, a high level abstraction for programming the underlying network, made possible by northbound APIs like Frenetic [14] and Pyretic [12] is extremely useful. Using these northbound APIs, modular and re-usable network applications can be built and composed in sequence or in parallel [12].

D. NOS & East-Westbound API

A network operating system (NOS) abstracts global view of a network and allows a network programmer to write different control applications as a centralized platform. Most popular NOSs are: NOX [15], POX [16], OpenDaylight [17] which use OpenFlow [13] based flow level control mechanism. For horizontal control information exchange between NOSs from different platforms, standardization of a east-westbound API is also important.

E. Virtualization layer

Virtualization enables multiple isolated VNOs to share a subset of network nodes as well as the radio spectrum. In the HetNet cloud architecture, flow-level virtualization [4] 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.

F. Southbound API

The controller platform modifies the forwarding behavior of the forwarding elements (switches, APs, RRHs, etc.) via a southbound API, it acts like a compiler for transforming the controller instructions to low level instructions that the nodes understands. The de-facto southbound API is OpenFlow [13], that is used extensively by both academia and industry alike. Other popular southbound APIs are NetConf [18], LISP [19], etc.

G. 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 has two sub-modules:
a network controller (NC): responsible for provisioning and management of virtual network nodes to VNOs; a 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 acts as 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.

H. Baseband processing

The HetNet cloud support multiple radio access technologies (RATs), 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].

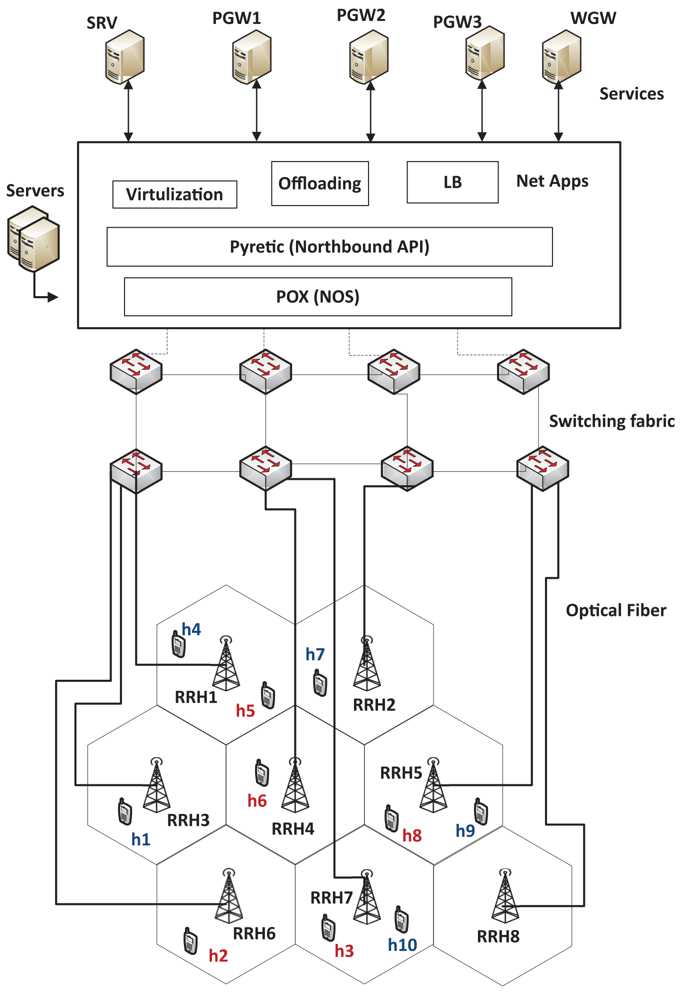

Fig. 3. Virtual wireless networks emulation scenario.

I. Radio access plane

Radio access to the UEs is provided by fiber-fed RRHs, 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.

IV. Service differentiation in HetNet cloud

As a proof of concept, we have implemented two VNOs in Mininet [21] emulation platform. These VNOs are implemented as two isolated slices sharing the same physical resources, e.g. computing & storage nodes, network switches, RRHs, etc. We envision a NFV implementation for the operators, where, various network processing nodes (e.g. PGW, SGW, MME, PCRF, HSS, baseband processing units, etc.) are implemented as software modules in data center servers. The schematic of the emulation structure is shown in Figure 3. In this experiment, we studied service differentiation provisioning for virtual wireless networks in a HetNet cloud model. We study how various mobile services can be provided differentiated QoS depending on the application requirement and also the user subscription category. More specifically, we studied load balancing for users of VNO1 that have different subscription categories (prioritized and normal) and also the offloading of delay tolerant traffic from VNO1 to VNO2. As performance metrics we measure round trip transmission delay (RTTD) and achievable throughput while implementing the traffic offloading and load balancing. Network applications, i.e. virtualization (slicing), offloading and load balancing are written using Pyretic [12], a domain specific programming language (DSPL), which is a northbound API, that uses POX [16] as the network operating system (NOS). While the southbound API, OpenFlow [13], populates the forwarding table of the underlying programmable switches (open virtual switch (OVS)) to forward traffic from the respective VNOs. UEs of VNO2 have degraded link quality than that of VNO1, hence their lower throughput.

In the emulation setup, VNO1 is a MVNO providing mobile network service, whereas VNO2 is a WiFi service operator, providing internet access to user through unlicensed spectrum. Varied radio link qualities for the two types of networks are realized by implementing more lossy links for the WiFi network. From QoS point of view, VNO1 guarantees better service quality via its dedicated licensed spectrum and high performance servers connected by high capacity network links. As show in Figure 3, users h2, h3, h5, h6 and h8, marked red, belong to VNO1, where h2 and h3 are prime customers and users h1, h4, h7, h9, h10, marked in blue, are served by VNO2. In the wireless data center, connection between the servers and switches are of 1GB capacity, no transmission delay and loss are assumed for these links. Server hosting PGW2 VM (for serving regular clients from VNO1) is connected via 800 MB link having 2% packet loss, while for the WGW VM (to serve delay-tolerant traffic), the link is 600 MB, with 0.5 ms delay and 2 % packet loss. These links are configured in such a manner so as to simulate a differentiated QoS. Connections between switches and between switches and RRHs are of 1GB capacity. The fiber length from WDC to the RRH is 2 km, hence a 0.01 ms of transmission delay is assumed, as typical delay for radio transmission over optical fiber is 5μs/km.

Users of VNO1 have simultaneous access to both the mobile network and the WiFi network. Given the omni presence of WiFi networks in our everyday ICT eco-system, e.g. WiFi networks in campuses, offices, shopping malls, airports, stadiums, etc., it is a reasonable assumption. For the service differentiation evaluation, delay sensitive traffics (e.g. file transfer, video streaming, etc.) from the users of VNO1 directed to PGW1 (default server for data traffic for the UEs of VNO1) are offloaded to the WGW server, that belongs to the VNO2. This helps saving licensed spectrum that can be used for providing

TABLE I
DELAY AND ACHIEVABLE THROUGHPUT FOR STATIC USERS

UE	Regular		Offloading		Load balancing	
	RTTD [min-avg]	Th [MB]	RTTD [min-avg]	Th [MB]	RTTD [min-avg]	Th [MB]
h2, h3	0.2-46	95.2	1.6-11	27.3	0.12-35	95.7
h5, h6, h8	0.36-28	94.7	1.32-12	28.8	0.21-39	79.8
h1-to-SRV	0.36-51	31.2	-	-	-	-
h1-to-WGW	1.45-50	10.3	-	-	-	-

TABLE II
DELAY AND ACHIEVABLE THROUGHPUT FOR MOBILE USERS

UE	Regular		Offloading		Load balancing	
	RTTD [max-avg]	Th [MB]	RTTD [max-avg]	Th [MB]	RTTD [max-avg]	Th [MB]
BH: h2-RRH6	1931-148	94.8	919-48	23.9	775-39	95.7
AH: h2-RRH5	1738-129	95.0	794-44	29	924-47	95.6
BH: h5-RRH1	1989-156	94.9	894-47	27.9	802-43	78.4
AH: h5-RRH2	2457-221	95	518-28	25.9	750-40	79.7
BH:h10-RRH7	2142-172	35.5	-	-	-	-
AH: h10-RRH1	1429-97	34	-	-	-	-

services having tighter QoS requirement, e.g. services producing more delay-sensitive traffic. Also, in case of VNO1, traffic from privileged users (h2, h3) are directed to server (PGW3) capable of providing better QoS from achievable throughput and RTT delay point of view. Table I shows the RTT delay and throughput for different service differentiation cases, when the users are static. RTTD are measured in ms and the throughput in Mbps. The 'Regular' column shows delay and throughput when traffic from users are forwarded to the server 'SRV'. The 'Offloading' is the measure when delay-tolerant traffic from VNO1 is offloaded to VNO2 and the 'Load balancing' shows the result of differentiated services for privileged (h2, h3) and regular (h5, h6, h8) users. Minimum and average delays are shown in the table. For the control information exchange between the controller and switches, the transmission time for the first packet is pretty high which in turn increases the average packet delay, in fact, the long term average delay is lower than the noted average delay in Table I. No offloading or load balancing is assumed for VNO2.

We implemented a random mobility model for the users of VNO1 and VNO2, Table II shows the maximum (for the first packet) and the average packet RTTD, including the average achieved throughput. The delay depends on the connected RRH and the links' qualities to the service nodes. BH stands for 'before handoff' and AH denotes 'after handoff' RRH connections of users. According to the 3GPP standard, preferable delay for LTE voice and video is $<$150 ms and maximum allowable delay is $<$400 ms. From the average RTTD values in Table I and II, we can see that the HetNet cloud architecture is very well able to satisfy these requirements.

V. Challenges

There are certain challenges that need to be addressed for successful realization of a HetNet cloud model, in this section we will briefly discuss some of them.

Balancing between network complexity and flexibility is an important design consideration. A flow-level [4] virtualization allows doing network virtualization at packet flow level but it is not possible to make any PHY and MAC layer modification. On the other hand, virtualizing the radio chain [22] allows to build new wireless protocol by using various PHY and MAC layer processing blocks but it has no provision for modular building of network applications. Hence design compromise is needed between the depth of network virtualization and achievable flexibility for building virtual networks.

VM placement is a critical issue, especially as some wireless applications are very delay sensitive, maintaining very low RTTD is crucial. Hence, in a distributed WDC model, locations of the DCs are very important. Moreover, from green communication point of view, the WDCs should be established in places that have access to renewable energy sources.

While SDN facilitates building **network security** applications, it has its fair share of security holes in the form of, malicious traffic flows, switch vulnerability, compromise of controller or control plane communication channels, etc [23]. Proper measures should be taken to tackle such network vulnerabilities, for example, controller replication, so that, a backup controller might take control of the operation in case the primary controller fails, auto healing mechanism to recover from security attacks.

Standardization of APIs is important for integration and synchronous functioning of different network devices and applications built by various vendors and operators. OpenFlow [13] is the de-facto southbound API maintained by open networking foundation (ONF) consisting of leading industry and academia partners. Similarly, standardization of east-westbound and northbound APIs are also necessary for controller platform integration and facilitating modular application building.

Backward compatibility is significant for any new technology. SDN and cloud computing technology are supposed to be gradually included to the existing production networks. Hence, it is of utmost importance that these networks operate smoothly with the existing networks.

VI. Conclusion

In this paper, we presented the HeNet cloud model that implements virtual wireless networks atop shared substrate of physical infrastructures. We have used a northbound API for building modular network applications, like virtualization, traffic offloading and load balancing, and compose them together to achieve complex network functionalities, e.g. service differentiation in virtual hetnet wireless networks. Emulation results show that, the HetNet cloud can achieve very low RTTD and high data throughput while ensuring service differentiation per user per application basis. Critical technical challenges for realizing such a virtual network model have also been discussed. Investigating the deployment challenges in NFV implementation in the context of HetNet cloud is the subject of our ongoing research.

References

[1] 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.

[2] M. Kobayashi, S. Seetharaman, G. Parulkar, G. Appenzeller, and et al., "Maturing of openflow and software-dened networking through deployments," *Elsevier*, 2013.

[3] 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.

[4] 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.

[5] 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.

[6] K. Pentikousis, Y. Wang, and W. Hu, "MobileFlow: Toward software-defined mobile networks," *IEEE Communications Magazine*, 2013.

[7] L. E. Li, Z. M. Mao, and J. Rexford, "CellSDN: Software-defined cellular networks," *Technical Report, Princeton University*, 2012.

[8] "C-RAN: The road towards green RAN," in *China Mobile Research Institute (CMRI), White Paper, version 2.5*, Oct. 2011.

[9] 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.

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

[11] T. Taleb, M. Corici, C. Parada, and et al., "Ease: Epc as a service to ease mobile core network deployment over cloud," *IEEE Network*, 2015.

[12] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing software defined networks," *USENIX NSDI*, 2013.

[13] 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.

[14] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: A network programming language," in *proc. of the ACM SIGPLAN International Conference on Functional Programming (ICFP)*, Sept. 2011.

[15] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, 2008.

[16] J. Mccauley, "Pox: A python-based openflow controller," *available at: http://www.noxrepo.org/pox/about-pox/*.

[17] "Opendaylight project," *available at: http://www.opendaylight.org/*.

[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] N. Handigol, B. Heller, V. Jeyakuma, and et al., "Reproducible network experiments using container-based emulation," in *proc. of CoNEXT*, 2012.

[22] M. Bansal, J. Mehlman, S. Katti, and P. Levis, "OpenRadio: A programmable wireless dataplane," in *proc. of the HotSDN*, Aug. 2012.

[23] D. kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, A. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, 2015.