MPLS based Hybridization in SDN

Yash Sinha, Siddharth Bhatia, Virendra S Shekhawat
Department of Computer Science & Information Systems
Birla Institute of Technology & Science, Pilani Rajasthan, India

G S S Chalapathi
Department of Electrical and Electronics Engineering
Birla Institute of Technology & Science, Pilani Rajasthan, India

***Abstract*—The new paradigm of Software Defined Networking (SDN) although has great potential to address the complex problems presented by enterprise networks, it has its own deployment and scalability issues. Further, a full SDN deployment has its own business and economic challenges. A smooth transition from legacy networks to SDN (disruption free, accommodating budget constraints, with progressive improvement in network management) requires a hybrid networking model as an inevitable intermediate step; that allows heterogeneous paradigms to function together while the full transition is realized in phases. Therefore, the need of the hour is to develop an incremental deployment strategy that caters to the needs of the organization. We present here a class-based hybrid SDN model for Multi Protocol Label Switching (MPLS) networks. We discuss the model, design, components, their interactions, advantages and drawbacks. We also present an implementation and evaluation of a prototype.**

In legacy networks, MPLS architecture closely resembles SDN paradigm in terms of separation of control and data planes, flow-abstraction etc. Moreover, ISPs have preferred MPLS over the years due to benefits of virtual private networks and traffic engineering. The central idea is to partition traffic using forwarding equivalence classes at the ingress router, the rules of which can be updated via a centralized controller using OpenFlow. Therefore, we aim to use the standard MPLS data-plane together with a control-plane based on OpenFlow to come up with a systematic incremental deployment methodology as well as a hybrid operation model

Keywords— SDN; MPLS; Hybrid-SDN; Traffic Engineering

I. INTRODUCTION

Transition from legacy networks (the term legacy network is referred here for the networking paradigm that was used before the advent of SDN) to full SDN poses challenges of budget constraints, disruption of services and lack of confidence among administrators [1]. Unless full transition takes place, benefits of SDN cannot be realized completely, rather during the transition phase it becomes more prone to failures and disruptions.

But combining SDN and legacy architectures in hybrid SDN models has the potential to sum their benefits while mitigating their respective challenges [2]. For example, certain legacy protocols and techniques can solve some difficulties in SDN. Automated configuration of SDN switches has been achieved by extension of DHCP to DHCP-SDN [3] and automatic bootstrapping of OpenFlow networks in minimal time for in-band controllers has been accomplished by DHCP, OF-Config (based on NETCONF [4]) and ARP protocol [5]. Also, ONF has made NETCONF as mandatory for configuration of OpenFlow enabled devices [6]. The need of the hour is to come up with a well-designed hybrid SDN model to assimilate the benefits of many of the similar works in a working, deployable, heterogeneous paradigm.

There are number of reasons, why MPLS networks are very much suitable for hybridization. Time-tested maturity and close resemblance with the SDN paradigm in terms of separation of control and data plane mechanisms and flow abstraction are some of the important factors considered to choose MPLS networks. We have discussed them in detail in Section III.

Further due to various recommendations of the researchers [7-8], to place the intelligence at the edge, we have placed the SDN switches at the edge. This enables simplicity of separation of forwarding and control and services at the core and at the edge. The various advantages of this approach are discussed in Section IV.

We present here a hybrid model where MPLS and SDN paradigms coexist and coordinate together to enable smooth transition. In particular, we use the standard MPLS data-plane together with a simple and extensible control-plane based on OpenFlow and SDN to come up with, a hybrid operation model. The traffic is partitioned using forwarding equivalence classes at the ingress router, which can be updated via a centralized controller using OpenFlow. We present the design, components, their interactions, the advantages and drawbacks of the model. We implement and evaluate a prototype implementation also.

II. RELATED WORK

Saurav Das et al. [9] demonstrate a prototype for MPLS based traffic engineering that uses the standard MPLS data plane and an OpenFlow based simpler and extensible control plane. But this is limited to pure OpenFlow networks only.

Although there has been little work in the area of MPLS networks for hybridization, [10] proposed an implementation of various for more efficient utilization of network resources with a reconfigurable centralized controller, which turns off certain network elements.

978-1-5386-2855-3/17/$31.00 ©2017 IEEE

Various models for hybrid SDN have been proposed by the researchers [2], which are categorized on the basis of respective roles of legacy network management system and SDN. Tradeoff analysis suggests that the combination of centralized and distributed paradigms can provide mutual benefits, as highlighted with examples in the introduction.

Topology-based hybrid SDN: Panopticon [1] involves dividing network into cell blocks, placing SDN switches at the edge of a cell block and computing spanning tree from a port to SDN switch. It explicitly leverages waypoints (similar to gateways) to control traffic and thus realize the SDN abstraction. Further, it isolates end-hosts in the legacy network using VLAN and restricts their traffic to traverse strategically upgraded SDN switches. One of the drawbacks of this approach is that it can apply rules only to the traffic passing through the SDN switch. This solution is constrained by maximum possible number of VLANs (i.e. 4096) and flow table entries. Further, whenever there is a little change in network topology, the whole network has to be setup manually, that needs to be automated.

Hong et al. [11] focus on Traffic Engineering (TE) and real-time attainment of global network topology view which consists of both legacy and SDN devices. It captures changes in the topology in real time by parsing messages flooded by legacy devices (like OSPF link-state advertisements). Putting SDN devices at edges is good for QoS and security purposes but this may not work well for TE and failure recovery application.

Integrated Hybrid SDN: Past efforts on Routing Control Platforms (RCPs) [12] and the ongoing efforts within Open Daylight Project (ODL) such as SNMP4SDN can be considered as examples of this hybrid model.

Automated SDN deployment methods [3] focus on automating SDN switch installation in hybrid networks to minimize risk of errors in manual configuration and reduce operational costs. A new protocol DHCP SDN has been designed to connect a new SDN/non SDN switch to an existing network with automatic configuration. It does not specify the details of any control that one may exercise on legacy switches. Security issues and network loop issues are yet to be explored.

The literature review reveals that service-based hybrid SDN and class-based hybrid SDN models have not been attempted yet. The following research challenges to be addressed for realizing such models:

- Active cooperation between architectural elements of different paradigms requires a (partial) control-plane redesign.
- Multiple paradigms running together can hamper the development of networks due to added complexity.

III. WHY MPLS NETWORKS

In MPLS networks, the inherent separation of control and data planes and flow-abstraction closely resembles the SDN paradigm, thus, makes it a very good candidate for designing a hybrid model.

Secondly, the data plane mechanisms remain the same simple push, swap and pop operations even when changes to the legacy network are required. Otherwise, the creation of a new service necessarily involves changes in protocols or the creation of an entirely new protocol. Both options are lengthy and time consuming. MPLS mechanisms have the potential to provide a simplified hardware [7].

Thirdly, MPLS has evolved over the twenty years and has stood the test of time. This can help the network fallback to the legacy mechanisms, thus reducing widespread service disruption and build confidence and expertise of the network operators while the full transition happens.

Fourthly, enterprise networks that are already using MPLS enabled data plane will find it economically cheaper to reuse the switches and routers while enabling the benefits of SDN at the same time. This can also be an incentive for initial adoption of this hybrid model among others [2].

IV. OUR APPROACH

A. Aim

We aim to use the standard MPLS data-plane together with a control-plane based on OpenFlow to come up with a class based hybrid operation model. Further, we wish to build a prototype of the model to leverage traffic engineering in a hybrid network.

B. Design

1) The Control Plane: It consists of an SDN controller that implements control mechanisms of MPLS also. It uses the OpenFlow protocol to communicate and install forwarding entries, MPLS labels etc. for the SDN switches. For the legacy MPLS switches, it implements the Label Distribution Protocol (LDP) to distribute labels. There can be other ways to enable controller-switch communication which will be discussed in later sections. Other required functionalities such as topology discovery, discovering traffic engineered paths, allocation of resources etc. are also implemented.

2) The Data Plane: All core switches are MPLS switches whereas the edge consists of SDN switches (with dual stack i.e., they support MPLS also). The SDN switches being in direct contact with the controller, act as “Label Edge Router” (LER) which assign a label to the incoming packet. Packets are forwarded along a "Label Switch Path” (LSP) where each "Label Switch Router” (LSR) makes forwarding decisions based solely on the contents of the label, which is like an indirect way to enforce the controller’s decision. At each hop, the MPLS router strips off the existing label and applies a new label which tells the next hop how to forward the packet.

3) Functionality: As soon as the network starts, the controller starts services like networking monitoring,

topology discovery, and resource allocation. It gets the policies that need to be implemented from the network administrator and discovers traffic engineered paths to fulfil them. The centralized controller having a global view of the topology and better programmability can outperform MPLS's ways of path discovery. It generates the labels and distributes them to the SDN and MPLS switches using appropriate protocols.

When a packet arrives at the edge, the labels are assigned. An SDN switch, which is efficient in reading and manipulating the packet headers (header rewriting, forwarding to controller for unmatched packets etc.) performs better than an MPLS switch in assigning a proper label to the packet. Further, label push/pop is carried out under indirect supervision of the controller. Thus the packet is routed better within an ISP or enterprise.

C. Advantages

The specific advantages of this hybridization model are:

- Reuse of existing MPLS infrastructure, thus reducing investment cost.
- Benefits of SDN with less disruption of services, hence an incentive for SDN adoption.
- Separation of forwarding for edge traffic and internal traffic enables better resource allocation and separation of services at edge (i.e., mobility, access control, billing) and at core (i.e., efficient packet forwarding)
- Separation of control enables to focus on different problems and goals for the edge (i.e., security, mobility, access control etc.) and core (i.e., network utilization, efficient packet processing).
- The controller can discover paths more efficiently with the help of global view of the network, network programmability, and active monitoring.
- With OpenFlow, we can readily provide all the services that MPLS networks provide. But more importantly, we can make changes to existing services or create new ones by simply changing the networking applications that run on the top of network operating system. New capabilities are no longer tied to protocol extensions that need to be implemented in each and every router in the network.
- OpenFlow doesn't need to change either, for all it gives is control over the simple push, swap and pop data plane operations which remain the same.
- The controller successfully installs rules for both SDN and legacy devices.
- The two paradigms can exert their control in a cooperating manner; we need not to switch off decentralized mechanisms to generate routes in the legacy paradigm.

  There can be two entries for the same flow which would create the problem. Use of different protocols helps in resolving conflicts. It provides a conflict free separation of the two control paradigms. For example, if we use OSPF packets injected by the controller to reroute packets then control plane in the switch may overwrite it in the subsequent cycle. This will not happen in the proposed model due to separate routing rules used for non-MPLS and MPLS traffic.

  Better optimizations are possible with the help of controller, many of which are impossible with the legacy paradigm. For example, routing traffic based on dynamic delay and bandwidth, incorporating administrator policies such as preferring a premium user over others.

D. Drawbacks

Besides, above mentioned advantages of the proposed network model, following can be treated as its limitations:

- Not every network supports MPLS.
- SDN devices at the edge is a must. This might mean a lot of investment for some organizations.
- Hybrid deployment may hamper other services such as use of firewalls and the network may not fallback to legacy mechanisms properly in case of failure.

E. Discussion

We discuss here how services are implemented at the controller and some of the assumptions.

*1) Topology Discovery:*The controller knows the details of the topology such as placement of the hosts, switches, link capacities and delays dynamically. Hosts can be discovered by tapping DHCP/ARP requests from the topology to the controller. Similarly, legacy switches can be identified using SNMP requests/traps. Topology discovery in a hybrid network is a separate area of research, and there can be many ways to do it.

2) SDN at Edge and Legacy at Core Network: We propose to keep the core network made of legacy switches whereas edge network of SDN compatible nodes. There have been a lot of recommendations by researchers to place intelligence of the network at the edges [7].

3) Controller Switch Communication: The controller communicates with the legacy switches via some mechanism for various purposes such as label distribution. The controller may inject packets into the network of legacy protocols like LDP. ClosedFlow [13] provides a promising idea of making legacy switches SDN compatible but there are other ways such as ACLs, CLI etc. which can be used to programmatically install routes in the switch/router using

the controller. For the prototype, we employed OVSDB protocol.

4) Path Discovery: The controller is supposed to generate optimal routes for a given network state and satisfy the policies specified by the network administrator. Although this is tricky, but there can be different ways to fallback if there is no path available to support the desired quality of service. This is yet another area of research. For the prototype, we have used SAMCRA (Self-Adaptive Multiple Constraints Routing Algorithm) [14] to calculate the shortest path as per desired constraints.

V. Implementation of the Prototype

A. Open Source Software

We use Mininet [15], a rapid prototyping emulator for emulating network elements such as hosts, links and switches on a single machine. We use Open vSwitch [16] for the Linux kernel space to emulate OpenFlow compatible switches as well as legacy switches along with the Ryu controller [17].

B. Network Topology

We have used SDN controller, SDN compatible switches and non-SDN switches in the network topology. All switches support MPLS and Spanning Tree Protocol (STP). We assume that initially STP is used to route packets and leverage MPLS to enable SDN control. The controller uses OpenFlow to push MPLS entries in the SDN switches whereas it uses OVSDB protocol to control the legacy switches as shown in Fig 1.

We deploy a simple switching hub application in the Ryu controller. The controller implements a simple switching hub, which learns the MAC addresses of connected hosts in a MAC address table, enables SDN switches to transfer packets to the corresponding port for connected hosts, and performs flooding for unknown hosts.

It has been assumed that the controller has the knowledge of the topology in advance and can take optimal routing decisions for the given traffic flow. The SAMCRA [14] algorithm is used for this purpose. The topology consists of a mesh of legacy switches with SDN switches placed at the edges. The controller injects rules to push and pop relevant labels at the switches in the path. As soon as a relevant flow is identified at the ingress switch, that is rerouted using MPLS.

C. Controller-Switch Communication

We use OpenFlow 1.3 as the communication protocol between the controller and the SDN enabled switches. The controller uses classes like “OFPInstructionGotoTable”, “OFPActionPushMpls”, “OFPActionSetField” and “OFPActionOutput” to create a new flow table, push MPLS eth_type, set MPLS label value and output to a desired port respectively. The hybrid model control is shown in Fig. 2.

Similarly, for non SDN switches we add the controller’s IP address as the remote manager which uses “ovs-vsctl” and “ovs-ofctl” commands to access the “db_server” associated the switch and installs entries for pushing and popping out labels.

D. Expected Consequence

If a premium user wants to download a big file, such requests can be routed through a network path having high bandwidth availability using this model. This ensures a better user experience with desired QoS. When network condition change (due to congestion, link failure, etc.), flows can be re-provisioned automatically on a new optimal network path.

E. Issues in Emulation

Some of the critical issues are listed below that we faced while using unstable MPLS code in the Open vSwitch (OvS).

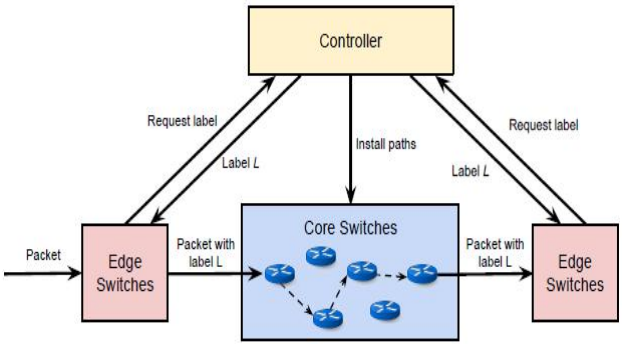

Fig. 1 Flow rules installation

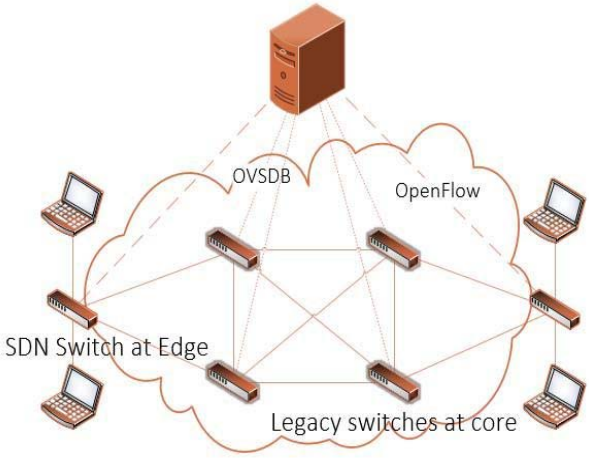

Fig. 2 The control on hybrid model

1) After an MPLS header is pushed, packets are not forwarded: If the switch is used in kernel mode, it does not forward a packet after it pushes an MPLS header on the top of it. This problem does not occur with the user-space version. However, user space version gives inferior network performance as more number of switches are emulated in the network topology.

2) MPLS header stacks are limited to size of three labels only: The MPLS stack does support processing of up-to three headers only. This was reported by the research community earlier also [18]. Thus we have used a small network topology for which three labels would suffice for traffic engineering. Therefore, datasets such as LBNL etc. could not be used to emulate a real-network topology. This constraint restricts the performance analysis to a limited scope. Further, if we are using version 2.3, which is the latest version available in Ubuntu 14 (Trusty Tahr) via apt-get manager, only one label is supported. Otherwise, we get a OFPT error as follows:

```
OFPT_FLOW_MOD (OF1.3) (xid=0x2):
(***truncated  to 64 bytes from 104***)
 00000000  04 0e 00 68 00  00 00 02-00 00 00 00
00 00  00 00|... h...........      |
 00000010  00 00 00 00 00  00 00 00-01 00 00 00
00 00  80 00|                      |
 00000020  ff ff ff  ff ff ff ff ff-ff ff ff ff
00 00  00 00|                      |
 00000030  00 01 00 17 80  00 00 04-00 00 00 02
80 00  0a 02|                      |
OFPT_ERROR (OF1.3) (xid=0x2):
OFPBAC_BAD_ARGUMENT
```

3) Fields cannot be matched beyond the MPLS header: If there is a packet with MPLS header, one cannot match other fields (such as ip_src, ip_dst) of higher layers. Although the SDN controller is supposed to be able to separate traffic into classes more than what legacy devices could do, but this bug limits the controller's ability to divert traffic even for a layer as low as layer 3 which is very crucial in network management. This problem is also mentioned in [19].

4) Ports in STP_BLOCK State Fully Blocked: When we run the STP in a layer 2 topology to remove any loops, certain ports are designated as STP_BLOCK so as to have a tree like topology that spans over all nodes of the topology. As the controller can intelligently use these ports to route traffic without getting into loop, it is specifically mentioned in the OpenFlow 1.0.0 specification [20] that "Note that forward actions that specify the outgoing port or OFP_ALL ignore the port status set by the STP". But currently flow entries cannot cause a packet out from a port that is in STP_BLOCK state. This crippled our emulated topology to be loop free. The solution that seems to work is that if a host is attached to the switch, none of the ports are sent to STP_BLOCK state. Therefore, we attach dummy hosts to core switches in the network.

5) Limited Support for LDP in Quagga: Due to the limitations of OvS switches, we tried the Quagga routing suite [21] but due to its incompatibility with recent kernel versions of Ubuntu 14 (Trusty Tahr) and 16 (Xenial Xerus) we postponed its deployment.

VI. EVALUATION

A. Emulation

The network topology used is as shown in Fig. 3. We have used loop free topology to avoid the STP_BLOCK state. The responsibility lies on the SDN switch at the edge to distribute traffic intelligently between the two paths based on link bandwidth, delays, and desired QoS to improve the user experience at the user end of the network.

The least hop path (shown in Fig. 3) is generally preferred between server and the end user, but it is often either choked due to congestion or offers limited bandwidth. Therefore, it is emulated as to be limited by low bandwidth. The longer unused path in Fig. 3 is largely unused but offers more available bandwidth. Consequently, this is the path being provisioned by the controller. The controller adds rules to the flow table to push and pop MPLS labels to reroute the traffic.

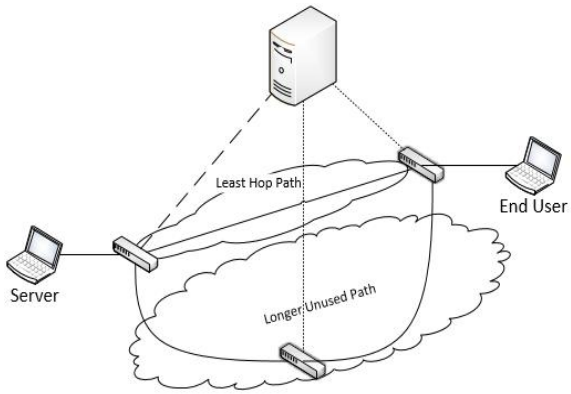

Fig. 3 Emulated topology

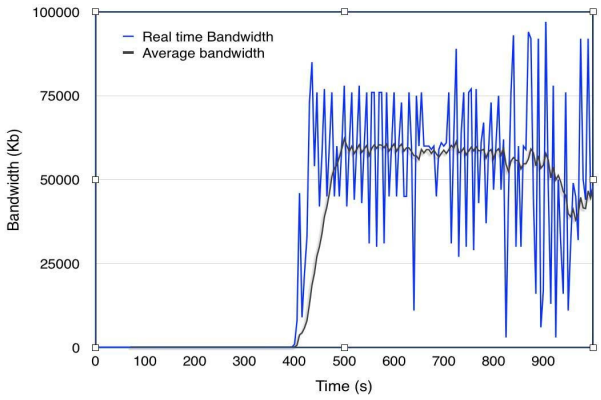

Fig. 4 Data transfer rate available at the end host

B. *Available bandwidth*

When the controller changes the path, the data transfer rate for the end host increases by a huge margin. The data transfer rate available at end host is plotted in Fig. 4. Using tcpdump and wireshark, we are able to verify that the packets in the network are rerouted with a path of better bandwidth. We use iperf to measure the recent available bandwidth.

VII. CONCLUSION

A hybrid network model which encompasses both SDN and MPLS based functionality is presented and experimentally evaluated with the help of a prototype implementation. The prototype implementation of the hybrid networking model gives the proof of concept for the co-existence of MPLS and SDN based network elements in the network. These switches can be configured using a centralized controller to exercise control and push configurations.

This model is able to accomplish conflict-free separation of centralized and decentralized controls. Also it has been proven that a centralized controller is able to provision traffic flows better as compared to decentralized based approaches using MPLS. Therefore, this model provides a better alternative for the smooth transition of a legacy network to a SDN enabled network.

VIII. FUTURE WORK

This prototyped model can further be extended and scaled out for large network topology with multiple traffic flows. This is going to be achieved in both ways; through a experimental test-bed setup and through emulation. on a topology that resembles a real network scenario. Control layer mechanisms of the MPLS protocol can supplement to the controller to provide topology information in dynamic manner.

References

[1] D. Levin, M. Canini, S. Schmid, F. Schaffert, and A. Feldmann, “Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks,” USENIX Annual Technical Conference (USENIX ATC 14), pp. 333–345, 2014.

[2] S. Vissicchio, L. Vanbever, and O. Bonaventure, “Opportunities and research challenges of hybrid software defined networks,” ACM SIGCOMM Comput. Commun. Rev., vol. 44, no. 2, pp. 70–75, 2014.

[3] R. Katiyar, P. Pawar, A. Gupta, and K. Kataoka, “Auto-Configuration of SDN Switches in SDN/Non-SDN Hybrid Network,” in Proceedings of the Asian Internet Engineering Conference, pp. 48–53, 2015.

[4] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, “Network configuration protocol (NETCONF),” Internet Eng. Task Force, RFC, vol. 6241, 2011.

[5] S. S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, “Automatic bootstrapping of OpenFlow networks,” in Local & Metropolitan Area Networks (LANMAN), 19th IEEE Workshop, pp. 1–6, 2013.

[6] N. Mckeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, and S. Louis, “OpenFlow: enabling innovation in campus networks,” ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, 2008.

[7] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, “Fabric: a retrospective on evolving SDN”. In Proceedings of the first workshop on Hot topics in software defined networks, pp. 85-90, August 2012.

[8] A. Manzalini, R. Saracco, “Software networks at the edge: A shift of paradigm,” Future Networks and Service (SDN4FNS), IEEE SDN, pp. 1-6, 2013.

[9] S. Das, A. R. Sharafat, G. Parulkar, and N. McKeown, “MPLS with a simple OPEN control plane,” in Optical Fiber Communication Conference and Exposition (OFC/NFOEC), and the National Fiber Optic Engineers Conference, 2011.

[10] N. R. P. Katov, N. Anton, Albena Mihovska, “Towards Internet of Services – SDN-enabled IMS Architecture for IoT Integration,” in Wireless Telecommunications Symposium (WTS), IEEE, pp. 1–8, 2015.

[11] S. B. Z. M. M. David Ke Hong Yadi May, D. K. Hong, Y. Ma, S. Banerjee, and Z. M. Mao, “Incremental Deployment of SDN in Hybrid Enterprise and ISP Networks,” in Proceedings of the 1st {ACM} {SIGCOMM} Symposium on Software Defined Networking Research, {SOSR}, Santa Clara, California, USA, 2016.

[12] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, “Design and implementation of a routing control platform,” in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-vol. 2, pp. 15–28, 2005.

[13] R. Hand, and E. Keller, “ClosedFlow: OpenFlow-like control over proprietary devices,” In Proceedings of the third workshop on Hot topics in software defined networking, pp. 7-12, August 2014.

[14] S. C. Ceballos, "Improving SAMCRA: A QoS Routing Algorithm, using the Look-Ahead Property," Electrical Engineering, 2002.

[15] "Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet," [Online], Available: http://mininet.org/

[16] "Open vSwitch," [Online]. Available: http://openvswitch.org/.

[17] "osrg/ryu: Ryu component-based software defined networking framework," [Online]. Available: https://github.com/osrg/ryu/

[18] "ovs/FAQ#L1238" [Online], Available: https://github.com/openvswitch/ovs/blob/c288d303b513f5874ea50c8845e719837ee47a23/FAQ#L1238

[19] "ovs/FAQ#L1243" [Online], Available: https://github.com/openvswitch/ovs/blob/c450371ec02341fe9ac0534e3727b7532235ccc3/FAQ#L1243

[20] "OpenFlow Switch Specification" [Online], Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf

[21] "Quagga Software Routing Suite," [Online], Available: http://www.nongnu.org/quagga/