Realizing Packet-Optical Integration with SDN and OpenFlow 1.1 Extensions

Meral Shirazipour, Wolfgang John[†], James Kempf, Howard Green and Mallik Tatipamula
Packet Technologies Research
Ericsson Research Silicon Valley
San Jose, CA, USA 95134

Abstract—**This paper discusses the benefits of applying software defined networking (SDN) to circuit based transport networks. It first establishes the need for SDN in the context of transport networks. This paper argues that the use of SDN in the transport layers could be the enabler for both packet-optical integration and improved transport network applications. Then, this paper proposes extensions to OpenFlow 1.1 to achieve control of switches in multi-technology transport layers. The approach presented in this paper is simple, yet it distinguishes itself from similar work by its friendliness with respect to the current transport layer control plane based on generalized multi-protocol label switching (GMPLS). This is important as it will enable an easier and gradual injection of SDN into existing transport networks. This paper is completed with a few use case applications of SDN in transport networks.**

I. Introduction

Software defined networking (SDN) or split architecture is a concept which allows network operators to flexibly manage routers and switches using software running on external servers. Split architecture refers to the decoupling of the control from the forwarding plane in switches and routers [1], [2]. Equipment vendors can leverage on split architecture designs to offer forwarding plane equipment separately from control plane equipment and software. Network operators can leverage on SDN to efficiently offer innovative services with improved quality of experience to end users. Control plane functions will thus see the day as soon as the need surfaces, as opposed to today's tedious process for integrating new control plane functionalities. If applied adequately, SDN/split architecture may also result in CAPEX and OPEX savings for the operators in the long run.

OpenFlow [3], [4] is a standard protocol that can be used for SDN. An alternative to OpenFlow would be IETF's ForCES [1] or any proprietary protocol. However, OpenFlow is currently the only standard, available and widely accepted protocol for SDN. OpenFlow gives a standard API to the forwarding plane which can be used by the remote control plane software to run the network and innovate with new control plane applications.

OpenFlow and SDN functionalities have been studied in the recent years for packet networks, but despite some work in the past years, the question remains open on how to efficiently integrate SDN in circuit switched and optical transport networks.

[†]The author was partially sponsored by the EU FP7 project SPARC (http://www.fp7-sparc.eu).

This article sheds light to this question by first, in Section II, recalling where the need for SDN comes from and discussing if there is a need for SDN-like functionalities at the transport network layers. This section also overviews current relevant work on the application of SDN to transport networks using OpenFlow. Then, Section III presents generalized multi-protocol label switching (GMPLS) as the existing transport layer control plane. Section IV presents the extensions required to OpenFlow 1.1 to support circuit based transport networks with the same benefits offered by a standard GMPLS control plane. Then, in Section V, use case applications of SDN in transport networks are presented. Finally, Section VI concludes this paper with a summary of the discussions and current state of the work. This paper does not give an overview of the OpenFlow protocol but refers the reader to the specification [4].

II. Software Defined Networking

A. Why the need for SDN?

In packet networks, the need for SDN comes mainly from new networking requirements imposed by new services: mobile broadband, video content caches deployed in access networks, cloud computing services and data centers deployed across the networks. There is indeed a shift in traffic trends associated with the larger presence of data centers and content delivery networks (CDNs), as well as inter-data center and inter-CDN communication. These will cause network operators to experience changes in traffic trends, with unpredictable behavior and a wider spread of source-destination end points. This creates the need, not only for new traffic engineering (TE) functions, but also for a framework that allows the on-demand creation and deployment of new TE mechanisms. TE has often been described as a process of putting the traffic where the resources are. Given the new traffic trends and their sophisticated traffic management requirements, the process is now referred to as traffic steering. SDN is the agreed on solution to such requirements.

B. Why the need for SDN in transport networks?

Inevitably, the stress on packet networks caused by the widespread of cloud computing and data center/CDN applications is reflected on the underneath transport network. Circuit based optical transport networks need to become more flexible with respect to the bandwidth demands of the above packet network layers. The proposed solution at the transport layers is the progress towards packet-optical integration. Packet-optical integration is a term given to the tight coupling between the packet network and transport layer equipment. It is a cost effective practice that will allow the packet based routers and switches to operate jointly with the underneath optical network elements. Such practice, especially at the edges of the network, will enable a more reactive approach in dealing with the recent dynamic traffic trends. The integration is about packet switches controlling the optical network switches. The current transport networks (mainly based on SONET/SDH) tend to be too slow to react dynamically to router traffic shifts as the two networks usually have separate control planes. Dynamic control planes such as GMPLS have been developed at the IETF to re-mediate to this problem, but the complexity and distributed nature of a control plane like GMPLS slows down its widespread deployment and does not allow its use for SDN like functionalities (e.g. on-demand creation of timely TE mechanisms in support of traffic steering applications). SDN applied to the transport layers can be the solution to packet-optical integration. However, any SDN solution at the transport layers needs to be gradual, as existing infrastructures cannot be torn down.

C. Recent SDN proposals for transport layers

Two main propositions exist in the literature on how to apply SDN to circuit based transport networks. First, there is the OpenFlow Circuit Switched Addendum v.03 [5] which is an extension to OpenFlow 1.0. It consists of seven additions to the OpenFlow 1.0 specification. The Addendum proposes extensions to accommodate certain transport layer technologies, in particular time division multiplexing (TDM). A basic circuit switched cross-connection table is defined inside the OpenFlow switch. This cross-connect table is to be kept separate from the usual OpenFlow packet *flow table*. The circuit switch flow table has four fields per input and output ports. These include the port, the lambda, the virtual port and the TDM signal and time-slots (starting time slot in the SONET/SDH encoding). Unfortunately, these extensions re-define from scratch the circuit resources used by the TDM and other transport technologies. This adds unnecessary complication to the OpenFlow protocol and compatibility issues with existing control planes (e.g. GMPLS). Second, the European project OFELIA addresses the same issue [6]–[8]. Their approach uses the user-network interface (UNI) and allows the controller to interact with the transport network element's control plane (GMPLS or other). They developed OpenFlow agents that communicate with the network element using Simple Network Management Protocol (SNMP). This is a short term solution in enabling OpenFlow on transport layers and is not a solution for SDN and packet-optical integration.

III. GMPLS as the Current Transport Layer Control plane

A control plane is generally defined as the infrastructure and intelligence responsible for the establishment and maintenance of connections in a network. It usually consists of:

1) the protocols and mechanisms for the diffusion of information (e.g. GMPLS routing protocols);
2) the algorithms and policies for finding the optimal paths between end points (e.g. path computation element (PCE)); and
3) the signaling for the setup and tear down of the connections (e.g. GMPLS signaling protocols).

The control plane interacts with the management plane on one side and with the data plane on the other. The management plane communicates with the control plane with standard protocols such as NetConf. But the control plane usually uses proprietary protocols to interact with the data plane. With SDN and OpenFlow, this will change as the interface between the data plane and the control plane is standardized. In the case of circuit based transport networks, the desired scenario consists of a set of networking elements serving various packet and circuit switching technologies with various granularity layers, all controlled by a unique centralized control plane.

Generalized multiprotocol label switching:

GMPLS [9], [10] is the de-facto control plane for transport networks. It has been developed at the IETF with close supervision by the ITU-T. GMPLS discusses optical and multi-layer networks by differentiating between multi-layer (ML) and multi-region (MR) concepts [11], [12]. A *region* refers to switching technologies. A *layer* refers to granularities inside a switching region. The interfaces on a GMPLS router or node can have one or many of the six defined switching capabilities. The interface can be 1-Packet switch capable (PSC), 2- Layer 2 switch capable (L2SC), 3-Time division multiplex capable (TDM), 4-Lambda switch capable (LSC), 5-Fiber switch capable (FSC), and 6-Data channel switch capable (DCSC) . These six switch types are what GMPLS nomenclature calls regions. An OC3, a VC4 or a VC11 are examples of GMPLS layers in TDM region.

The interface switching capability (ISC) is the interface's ability to forward data of a particular data plane technology, uniquely identified by the six GMPLS switching regions. A node can have a single or multiple switching type capabilities. Moreover, MR switches are denoted as *Simplex* or *Hybrid*. Hybrid switch nodes have at least one interface which supports more than one switching type (region). RFC5339 [13] defines the concept of adjustment or adaptation between regions. Hybrid switches need to further define the interface adjustment capability (IAC) of their links. All these technological details will need to be considered in the OpenFlow based control plane defined in this paper.

To refer to specific transport level technologies, GMPLS standardizes label formats per technology that will be borrowed in the extensions proposed in Section IV. These 32 bit labels are defined as in the following. RFC3471 [10] presents the basic label in GMPLS used to signal a connection request (i.e. label switched path (LSP)). The actual resources are allocated using per technology specific labels. RFC4606 [14] defines SUKLM encodings to represent TDM resources in a single GMPLS label. RFC6205 [15] encodes the wavelength division multiplexing (WDM) labels as described by ITU-T G.694.1 (Dense WDM) and G.694.2 (Coarse WDM) labels. Optical transport network (OTN) technology is being standardized at the IETF with GMPLS labels specifying each OTN layer and granularity. The IETF is currently working on the standardization of GMPLS labels for the flexible grid in WDM networks [16]. As seen below in Section IV, reusing these existing standardized encodings will remove some of the burden of developing a new control plane; however it is important to recognize that many optical issues in GMPLS are still a matter of investigation.

IV. Proposed Transport Layer Extensions to OpenFlow 1.1

This section presents the OpenFlow 1.1 protocol extensions that allow a unified control of ML/MR switches in today's transport networks. The goal is to allow the OpenFlow controller to effectively retrieve the information of the switch's ports, switching technologies, available bandwidth (time slot, wavelength, etc.), and available MR adjustment capabilities and capacities.

OpenFlow usually relies on a centralized control plane (e.g. NOX [17]) which commands the switch's behavior via a secure layer 4 interface. Inside the switch, the interface between the data plane and this channel is implementation specific and out of scope of this paper. It is interesting to note that the OpenFlow protocol already handles two different packet based switching regions (PSC and L2SC), in a non-hierarchical or flat way. This is not a challenge because both regions are packet switched. For the case of optical and circuit based ML/MR switches, a more future proof architecture is desirable when extending OpenFlow since adding new tuples for each transport layer switching technology is not adequate.

Similar to [5], this work proposes a circuit flow table. A circuit flow table has a different use than the current packet flow table; the former will only represent existing connections while the latter serves in a per tuple lookup process. The fundamental difference between circuit switched and packet switched OpenFlow is therefore the fact that the circuit flow table is not used to lookup circuits. The OpenFlow controller is responsible for setting up the circuits' cross-connections in the switch using the OpenFlow protocol and treating messages received from the switch regarding the current state of connections. The circuit cross-connections are established generally in a proactive way, i.e., no packet is forwarded to the controller for circuit flows. However, a packet sent to the controller can trigger the establishment of a new circuit cross-connect (e.g. pre-configured cross-connects, similar to virtual TE-links in GMPLS). The extensions proposed in this section consider hybrid switches with both circuit based and packet based interfaces. This is not be confused with the OpenFlow-hybrid terminology defined in [4].

The proposed OpenFlow extensions partly rely on existing GMPLS features. While GMPLS covers the three control plane roles presented in Section III, the proposed OpenFlow extensions only rely on GMPLS' way of provisioning new

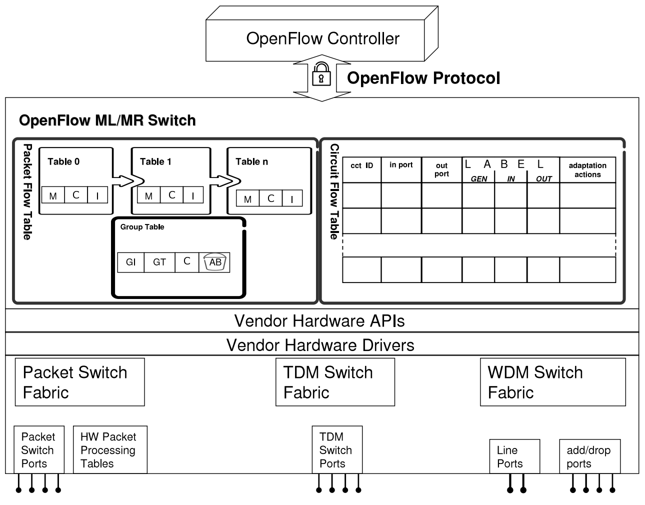

Fig. 1. Possible OpenFlow multi-layer/multi-region switch architecture

CCT ID	*in port*	*out port*	*Gen Label (encoding,ST,G-PID)*	*label in (e.g. TDM/WDM)*	*label out (e.g. TDM/WDM)*	*adaptation actions*

Fig. 2. Circuit flow table entry

connections with the standardized label encodings (role 3). There is no need for diffusion of information (role 1, e.g. TE-link advertisements performed by GMPLS routing protocols), as the information is usually centralized in the OpenFlow controller. The problem of how to gather this information is out of scope of this paper. Finally, TE and PCE applications are also centralized in the OpenFlow controller (role 2).

The proposed node architecture of an OpenFlow ML/MR switch is shown in Fig. 1. Packet flow tables (left side of Fig. 1) are consulted on the fly for each packet to determine its forwarding and required actions. Circuit flows (right side of Fig. 1) represent existing physical circuits established by the switch. The circuit IDs serve as virtual ports to other flows. A circuit ID is a virtual port to which incoming packet flows can be forwarded. Other circuit flows can also point to a circuit ID and hence form circuit hierarchies (the equivalent to GMPLS LSP nesting [12]). The circuit flows do not affect the on the fly processing of packets. The proposed architecture is an extension to the OpenFlow 1.1 specification, the left side of Fig. 1 is therefore left unmodified.

Fig. 2 shows that each entry in the circuit flow table consists of a set of circuit identifiers and descriptive fields. Again, the circuit table is just an internal representation of existing cross-connects inside the switch and a new entry is added each time the controller signals the establishment of a new circuit. For the case of bidirectional circuits, the circuit will occupy two entries in the circuit flow table, as resources (In/Out Labels) may not be symmetrical in both directions. The different circuit flow table fields are described below.

- Circuit Identifier (CCT ID): a 32 bit unsigned integer represents the circuit flow and also corresponds to a virtual port to which other flows can be forwarded.
- In Port/Out Port: a 32 bit unsigned integer represents the incoming/outgoing port number between which the

circuit cross-connects have been programmed.

- General Label: a 32 bit unsigned integer represents the information required to fully characterize the cross connect as part of an end-to-end circuit. This label thus represents the encoding, switch type and payload ID, using GMPLS standard encodings [10]. The OpenFlow extension separates these three values as in the following:
  - *Encoding:* an 8 bit unsigned integer that designates the encoding type of the connection. For example, for a packet over SONET circuit flow, the encoding is SONET. The GMPLS compliant encodings are defined in an enumeration named `ofp_cct_encoding`.
  - *Switch Type:* an 8 bit unsigned integer that designates the switching type (region) used on the link. This is particularly important for hybrid switches which may have interfaces supporting more than one region. The GMPLS compliant switch type encodings are defined in an enumeration named `ofp_swtype`.
  - *G-PID:* a 16 bit unsigned integer that designates the payload of the client using the circuit, as defined in RFC3471 [10]. The GMPLS compliant encodings are defined in an enumeration named `ofp_cct_gpid`.
- In Label/Out Label: a 32 bit unsigned integer represents the incoming/outgoing label following GMPLS standardized labels per technology (TDM, WDM, etc.).
- Adaptation actions: the adaptation of the signal from the input towards the output port. This field can also be used in the future for specific technology related actions (e.g. related to optical technologies).

The establishment of a new circuit flow (and hence its addition to the circuit flow table) must carry enough information to allow the switch to program its cross-connections. To be able to signal the new circuit flow cross-connect, the controller first needs to know the features of the switch, its ports, and the available resources. The information stored in the circuit flow table comes from the controller and is sufficient for the switch to establish the circuit connection. To this end, and for fast circuit restoration applications, the controller needs to keep an updated view of the switch's resources and state. It may also require residing in close proximity to avoid communication delays. The control plane design optimization is out of scope of this paper.

V. SDN Use Cases for Transport Networks

This section presents use cases specific to the application of SDN and split architecture in a ML/MR transport network. Some of these applications are possible even without the SDN/split architecture, but are made easier and more efficient with it. For each use case, the benefits of the SDN/split architecture remain the same and are mainly shown around:

1) Centralized control (new TE mechanisms, optimized routing and traffic/bandwidth steering possibilities).
2) Ability to quickly and independently scale and evolve data plane and control plane functionalities (i.e., data plane's or control plane's ML/MR functionality to evolve or just scale disjointedly).

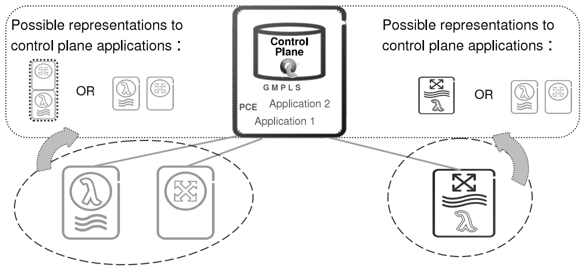
Possible representations to control plane applications :
OR
Control Plane
GMPLS
PCE
Application 2
Application 1
Possible representations to control plane applications :
OR

Fig. 3. Control plane views provided to packet-optical integrated applications

The first point can generally be achieved without split architecture at perhaps higher cost or less flexibility. However the second point usually requires a split architecture design.

A. Smooth migration towards packet-optical integration

The proposed extensions to OpenFlow can be used to smooth the migration process towards ML/MR, fully packet-optical integrated nodes. Packet-optical integration is not only a physical combination of the two domains but is also about the routers and switches to control the optical switches (i.e., unified control plane). There is a need for a gradual migration towards a fully packet-optical integrated network. Fully packet-optical integrated nodes will for example consist of routers with reconfigurable optical add-drop multiplexers (ROADMs) in the same box. Today's transport networks separate these two functionalities in separate boxes. Often, the control planes completely separate the packet layers from the transport optical layer, and at best offer an overlay model (e.g. GMPLS overlay model). The integration of packet-optical domains is desired for rapid and flexible traffic steering. The migration (node replacement) may take some time and require detailed planning. OpenFlow based split architecture can ease this process by defining migration applications that leverage on the data plane abstraction given by OpenFlow. The network applications will not have to worry about the underlying hardware changes as OpenFlow will provide a common control plane view of the various network elements. Fig. 3 presents how a controller application can have the desired view for both separated and physically integrated packet-optical functionalities. In both cases, and during the transition, the benefits of packet-optical integration are experienced. This method by itself can be considered as a partial packet-optical integration. Note that the mechanism to migrate to a fully SDN based transport network is incorporated in the SDN philosophy itself. Moreover, this method can be used for the integration of any other type of technology into an existing network; the integration will be programmed through SDN.

B. Bandwidth steering in support of traffic steering

The previous use case presented a smooth migration towards packet-optical integrated technology. Packet-optical integration is required because traffic is becoming increasingly distributed in addition to the bandwidth requirements varying

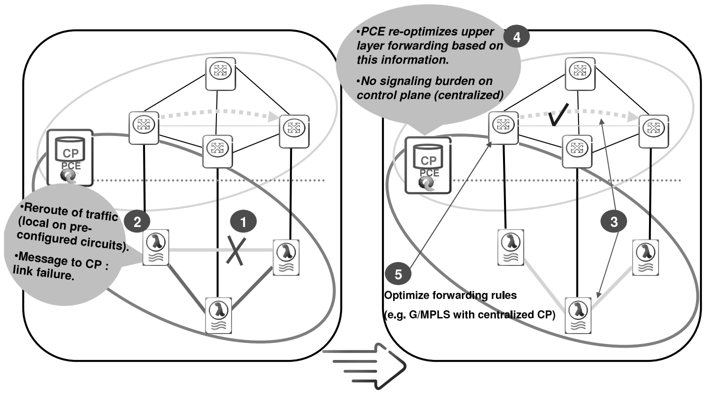
•PCE re-optimizes upper layer forwarding based on this information.
•No signaling burden on control plane (centralized)
4
CP
PCE
•Reroute of traffic (local on pre-configured circuits).
•Message to CP : link failure.
2
1
3
5
Optimize forwarding rules
(e.g. G/MPLS with centralized CP)

Fig. 4. Example optimized restoration as a result of flexible control plane

both in size and in direction. All these will require various-sized traffic trunks to be quickly and dynamically steered. A SDN/split architecture based ML/MR control plane is well suited for the development of such steering functionalities, with fast and cost effective (software based) operator customization possibilities (e.g. efficiently putting the transport layer at the service of the packet layers). A ML/MR OpenFlow, or centralized ML/MR control, can orchestrate the traffic steering with the underneath transport layer bandwidth steering functions.

C. More efficient multi-layer/multi-region recovery and restoration

Typically in a ML/MR network a single failure in a server layer may induce a cascade of failures in client layers. With a proper splitting of centralized/end-to-end versus distributed/local failure recovery techniques, OpenFlow can offer the best of both worlds when dealing with restoration scenarios. Like with G/MPLS, fast restoration paths can be preconfigured in advance. Fig. 4 shows an example. Upon failure, point (1) in Fig. 4, the protected services can be re-routed in a timely fashion, (2)-(3). Failure detection mechanisms are yet to be incorporated into the OpenFlow architecture. At the same time (2), the centralized controller can be notified about the failure. Then, the controllers PCE application can re-optimize the rest of the traffic (4), and update the forwarding behavior of the network elements (5). This last point usually puts a signaling burden on dynamic control planes (e.g. GMPLS). The split architecture and its flexible SDN based controller alleviate this problem.

VI. Conclusion

This paper showed the relevance of applying SDN concepts to transport layer networks. Transport networks are ML/MR and thus composed of multiple technologies and granularity layers. The proposed solution builds on top of the OpenFlow 1.1 specification, and with minor additions, allows a ML/MR switch to be controlled from a distant controller. The extensions are complete as they are based on the GMPLS standards which, under the supervision of IETF and ITU-T, fully cover existing technologies and can easily extend to cover upcoming ones. This allows the OpenFlow extensions to not only be inherently compatible with GMPLS control planes, but to also inherit the extensibility feature of GMPLS (i.e., simply define new label encodings each time a new technology surfaces). This paper also presented a few use case applications of SDN for ML/MR transport networks, all aligned with the current trend for packet-optical integration. Current work consists in the alignment of the proposed extensions with new versions of OpenFlow currently in development at the Open Networking Foundation (ONF).

References

[1] IETF Working group, “Forwarding and control element separation (forces) framework,” http://www.ietf.org/html.charters/forcescharter.html/.

[2] European Commission Seventh Framework Programme (FP7), “Split architecture carrier grade networks (sparc),” http://www.fp7-sparc.eu/, 2010.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling innovation in campus networks,” *SIGCOMM’08, ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 69–74, April 2008.

[4] OpenFlow Switch Consortium, “OpenFlow version 1.1.0 Switch Specification,” http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf, February 2011.

[5] S. Das, “Extensions to openflow protocol in support circuit switching, addendum to openflow protocol specification (v1.0),” http://www.openflow.org/wk/index.php/PAC.C, June 2010.

[6] European Commission Seventh Framework Programme (FP7), “Openflow in europe: Linking infrastructure and applications,” http://www.fp7-ofelia.eu/, 2011.

[7] S. Azodolmolky, R. Nejabati, E. Escalona, R. Jayakumar, N. Efstathiou, and D. Simeonidou, “Integrated OpenFlowGMPLS Control Plane: An Overlay Model for Software Defined Packet over Optical Networks,” *ECOC’11, 37th European Conference and Exhibition on Optical Communication*, pp. 1–3, September 2011.

[8] D. Staessens, S. Sharma, D. Colle, M. Pickavet, and P. Demeester, “Software defined networking: Meeting carrier grade requirements,” *LANMAN’11, 18th IEEE Workshop on Local and Metropolitan Area Networks*, pp. 1–6, October 2011.

[9] E. Mannie, *RFC3945: Generalized Multi-Protocol Label Switching (GMPLS) Architecture*, IETF, October 2004.

[10] L. Berger, *RFC3471: Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description*, IETF, January 2003.

[11] I. Bryskin and A. Farrel, *RFC4397: A Lexicography for the Interpretation of Generalized Multiprotocol Label Switching (GMPLS) Terminology within the Context of the ITU-T’s Automatically Switched Optical Network (ASON) Architecture*, IETF, February 2006.

[12] K. Shiomoto, D. Papadimitriou, J.-L. LeRoux, M. Vigoureux, and D. Brungard, *RFC5212: Requirements for GMPLS-Based Multi-Region and Multi-Layer Networks (MRN/MLN)*, IETF, July 2008.

[13] I. Bryskin and A. Farrel, *RFC5339: Evaluation of Existing GMPLS Protocols against Multi-Layer and Multi-Region Networks (MLN/MRN)*, IETF, September 2008.

[14] E. Mannie and D. Papadimitriou, *RFC4606: Generalized Multi-Protocol Label Switching (GMPLS) Extensions for Synchronous Optical Network (SONET) and Synchronous Digital Hierarchy (SDH) Control*, IETF, August 2006.

[15] T. Otani and D. Li, *RFC6205: Generalized Labels for Lambda-Switch-Capable (LSC) Label Switching Routers*, IETF, March 2011.

[16] D. King, A. Farrel, Y. Li, F. Zhang, and R. Casellas, *Generalized Labels for the Flexi-Grid in Lambda-Switch-Capable (LSC) Label Switching Routers*, IETF, October 2011, Individual Draft: draft-farrkingel-ccamp-flexigrid-lambda-label-01, Work in progress.

[17] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, “NOX: towards an operating system for networks,” *SIGCOMM’08, ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 105–110, July 2008.