MPLS-TE and MPLS VPNs with OpenFlow

Ali Reza Sharafat, Saurav Das, Guru Parulkar, and Nick McKeown
Department of Electrical Engineering, Stanford University, Stanford, California 94305, USA
sharafat, sd2, parulkar, nickm@stanford.edu

ABSTRACT
We demonstrate MPLS Traffic Engineering (MPLS-TE) and MPLS-based Virtual Private Networks (MPLS VPNs) using OpenFlow [1] and NOX [6]. The demonstration is the outcome of an engineering experiment to answer the following questions: How hard is it to implement a complex control plane on top of a network controller such as NOX? Does the global vantage point in NOX make the implementation easier than the traditional method of implementing it on every switch, embedded in the data plane?

We implemented every major feature of MPLS-TE and MPLS-VPN in just 2,000 lines of code, compared to much larger lines of code in the more traditional approach, such as Quagga-MPLS. Because NOX maintains a consistent, up-to-date topology map, the MPLS control plane features are quite simple to implement. And its simplicity makes it easy to extend: We have easily added several new features; something a network operator could do to customize their network to meet their customers' needs.

The demo consists of two parts: MPLS-TE services and then MPLS VPN driven by a GUI.

Categories and Subject Descriptors: C.2.1 – Computer Systems Organization [**Computer-Communication Networks**]: Network Architecture and Design

General Terms: Management, Design, Experimentation

Keywords: MPLS, MPLS-TE, VPN, Traffic Engineering, OpenFlow

1. SCIENTIFIC RATIONALE
We claim that while the MPLS data plane is fairly simple, the control planes associated with MPLS-TE and MPLS VPNs are rather complicated. For instance, in a typical traffic engineered MPLS network, one needs to run OSPF, LDP, RSVP-TE, I-BGP, and MP-BGP to name a few protocols. The distributed nature of these protocols results in excessive traffic of update messages when there are frequent changes in the network. This, in turn causes the routers to spend a lot of CPU time recalculating routing information. Hence, CPU message queues may get filled leading to incoming hello messages getting dropped. This leads to false link-state information being distributed throughout the network. The described vicious cycle causes large convergence times for the above protocols, meaning excessive control traffic on the network and stale information on the routers.

In SDN, the Network Operating System (NOS) is responsible for constructing and presenting a logically centralized map of the network. Instead of a set of distributed protocols implemented on each router, we implement these functionalities as simple software modules that work on the network map in NOS. Implementation of these functions on a logical map of the network is very simple. Hence, by pushing the control plane functionality to NOS, we benefit from not only simplicity of implementation, but also the fact that maintaining and updating applications are easy as well. This is because new features are no longer tied to multiple protocols that would normally have to be changed. In fact, with the controller in charge of the control plane, there is no need for any distributed protocol running in the routers as the NOS has complete knowledge of the network.

2. ARCHITECTURE
The architecture of our system is given in Figure 1. Our test-bed consists of several software and physical switches. The software switches are instances of Open vSwitch [2] which are hosted within the Mininet environment [3]. These switches are connected to a network of physical switches. Both software and physical switches support the OpenFlow 1.0 specifications [4] as well as the MPLS related section of the OpenFlow 1.1 specifications [5]. The switches are designed so that they handle the data plane, and not the control plane functionality of MPLS.

With the abovementioned network, we emulate a wide area network for the purpose of our demonstration. All switches are controlled by a single instance of the NOX [6] controller. The MPLS-TE and VPN services are managed via an application that runs in NOX. The control plane and the MPLS features are exclusively handled by NOX. The data plane simply supports the push swap and pop actions. When changes are needed in the data plane, NOX modifies the flow tables in the appropriate switches.

We use multiple GUIs to show the workings of the network and to dynamically interact and modify the TE-LSPs and/or VPNs.

3. DEMO SCENARIOS
The demonstration consists of two parts, the first pertaining to MPLS-TE [7] and the second pertaining to MPLS VPNs.

3.1. MPLS-TE
The first part of the demo is visualized via two GUIs, both showing the topology of the entire physical network. The first GUI displays the IP flows in the network and the second GUI displays the LSPs and the flows routed through them. All the flows and LSPs are color-coded to distinguish between various

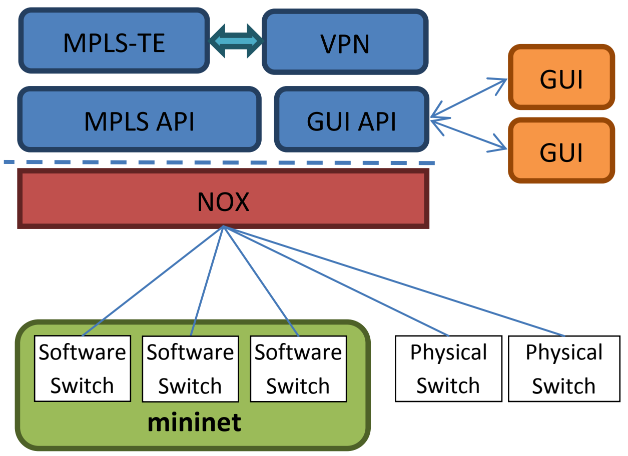

Figure 1. The architecture of the physical network and the controller used in our demonstration.

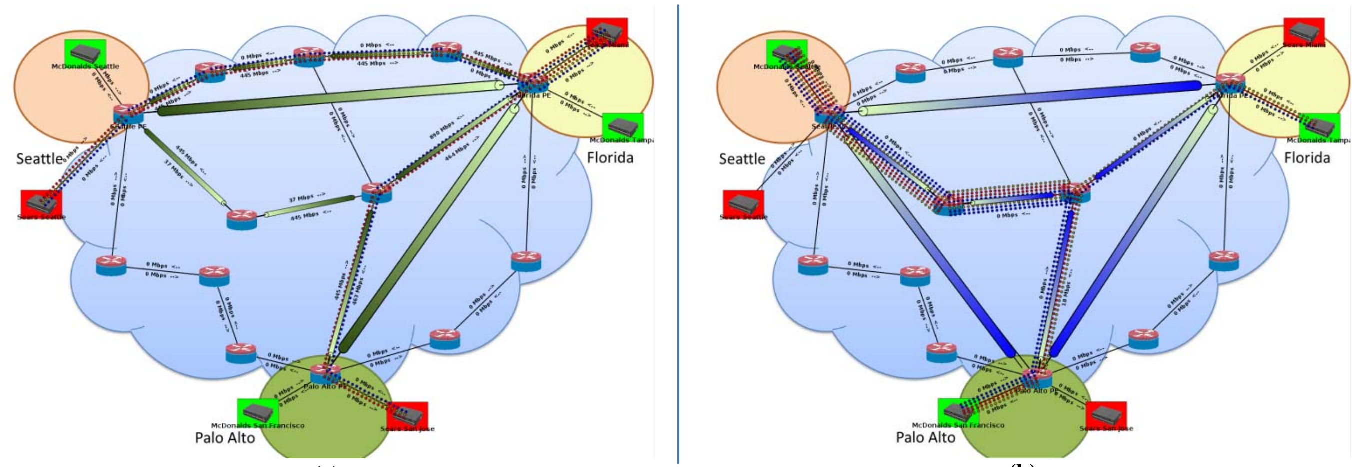

Figure 2. Sample GUIs corresponding to the MPLS VPN demonstration. Two VPNs with their corresponding flows and LSPs are shown in (a) and (b)

types. The demonstration starts with all the flows in the IP plane, and as we step through the demo, we create TE-LSPs and reroute some of the IP flows through the LSPs. By creating TE-LSPs of different characteristics, we demonstrate the following features:

Constrained Shortest Path First (CSPF) The CSPF algorithm allows us to find the shortest path for a TE-LSP that satisfy its bandwidth and priority requirements.

Auto-route When a TE-LSP is created, we automatically reroute any flow whose path goes through the head-end and the tail-end router of the created LSP onto the LSP.

Traffic Aware LSPs We can create TE-LSPs that carry a specific type of traffic. These can include VOIP, HTTP, etc.

Priority LSPs can have different levels of priority. When the reservable bandwidth of a link is fully allocated, we reroute LSPs of lower priority to alternative paths.

Auto-bandwidth The auto-bandwidth feature allows for the bandwidth reservation of a TE-LSP to dynamically adjust to its actual bandwidth usage (as opposed to the bandwidth being statically set at the creation of the LSP).

Interactive Management Users can create custom TE-LSPs from the GUI. That is, the user specifies the head-end and tail-end routers and the characteristics of the desired TE-LSP, which is then created in the network and shown in the GUI.

3.2. MPLS VPNs

The second part of the demo is visualized using multiple GUIs as well. In this case, all GUIs show the physical network, and each GUI pertains to a VPN and its associated flows and LSPs. Screenshots of working versions of this part of the demo are given in Figure 2. Our MPLS-VPN network has a simpler backbone topology compared to the previous section, but with some customer nodes added to the edge of the backbone network.

The backbone network is MPLS-TE enabled and so when LSPs are created to support a VPN, they are accompanied with all the TE features mentioned in Section 3.1.

We configure multiple VPNs with overlapping private IP address spaces. Each VPN comes with different characteristics which enables us to demonstrate the following:

Isolation of VPNs We demonstrate that multiple VPNs can coexist in a single backbone network where different VPNs may have overlapping address spaces. The flows associated with different VPNs are routed through their respective LSPs and do not aggregate with each other.

Custom Topologies The logical topology of each VPN can be specified by the customer. The topology can be anything from the traditional hub-and-spokes to full-mesh.

TE Services Since the backbone network is MPLS-TE enabled, we can readily offer TE services to the customers. We demonstrate the Priority and Auto-bandwidth features of TE. This allows for a high priority VPN to force others to reroute when it requires more bandwidth along its LSPs.

Interactive Management Users can create a new VPN by specifying the connection between the customer and provider routers as well as the topology and other characteristics of the network. After the specifications are given, we create the desired VPN network and display the results in the GUI.

4. REFERENCES

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, April 2008.

[2] Open vSwitch [Online]. Available: http://openvswitch.org/

[3] B. Lantz, B. Heller, and N. McKeown. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. In *ACM SIGCOMM HotNets Workshop*, 2010.

[4] OpenFlow Switch Specification: Version 1.0.0 [Online]. Available: http://www.openflow.org/documents/openflow-spec-v1.0.0.pdf

[5] OpenFlow Switch Specification: Version 1.1.0 Implemented [Online]. Available: http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf

[6] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: Towards and operating system for networks. In *ACM SIGCOMM Computer Communication Review*, July 2008.

[7] S. Das, A. R. Sharafat, G. Parulkar, and N. McKeown MPLS with a Simple OPEN Control Plane. In *Optical Fiber Communication Conference*, March 2011.