MPLS with a Simple OPEN Control Plane

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

Abstract: We propose a new approach to MPLS that uses the standard MPLS data plane and an OpenFlow based simpler and extensible control plane. We demonstrate this approach using a prototype system for MPLS Traffic Engineering.
OCIS codes: (060.4250) Networks; (060.4259) Networks, packet-switched

1. Introduction

MPLS has evolved over 15 years as a solution for ISPs. It is primarily used to perform traffic engineering in IP networks and for offering MPLS based L2 or L3 enterprise VPN services. Providers require traffic engineering for effective use of their network infrastructure, and enterprise VPNs continue to represent one of their most profitable service offerings. MPLS is the preferred solution for such services, mainly because plain-vanilla IP networks are incapable of providing the same, and older ways of providing these services using ATM or Frame-Relay are going away. But as carriers deploy MPLS they find that (1) even though the MPLS *data plane* is meant to be simple, vendors end up supporting MPLS as an additional feature on their very complex, energy hogging, expensive core routers such as Cisco's CRS-1 and Juniper's T-640; and (2) the IP/MPLS *control plane* (Fig. 1a) has become too complex leading to increased cost and fragility.

We propose the use of Open Programmable Extensible Networks (OPEN) to provide these services with the MPLS data plane and the OPEN control plane instead of the IP/MPLS control plane. Briefly, important attributes of OPEN† include (Fig. 1b): separation of data and control planes; flow based datapath where flows (not packets) are the fundamental unit of control; a uniform vendor agnostic interface called OpenFlow [1] between control and data planes for programming and controlling the switches; a logically centralized control plane, realized using a network OS, that constructs and presents a logical map of the entire network to services or control applications on top; and slicing and virtualization of the underlying network. In OPEN a researcher, network administrator, or third party can introduce a new capability by writing a software program on top of the network OS that simply manipulates the logical map of a slice of the network [2]. While some core ideas of OPEN have been proposed separately in the past, their implications together are far-reaching. OPEN promises to reduce the Total Cost of Ownership of various network infrastructures in part by simplifying the datapath switches; it provides greater control to the operator to customize and optimize their networks for the features they need and the services they provide; and finally it allows innovation to take place at a much faster rate, by helping the network evolve more rapidly in software (implemented outside-the-box), with possibly greater diversity of solutions generated by a larger pool of developers.

In this paper, we report on a prototype and demonstration of an OPEN network with an MPLS data plane and OPEN control plane that implements MPLS-TE as a network application (Fig. 1b). We discuss deficiencies of the IP/MPLS control plane with focus on MPLS-TE and suggest how a few new control applications on the network OS can be used to replace all MPLS control plane functionality like distributed signaling and routing. Finally we present implementation details of our prototype system.

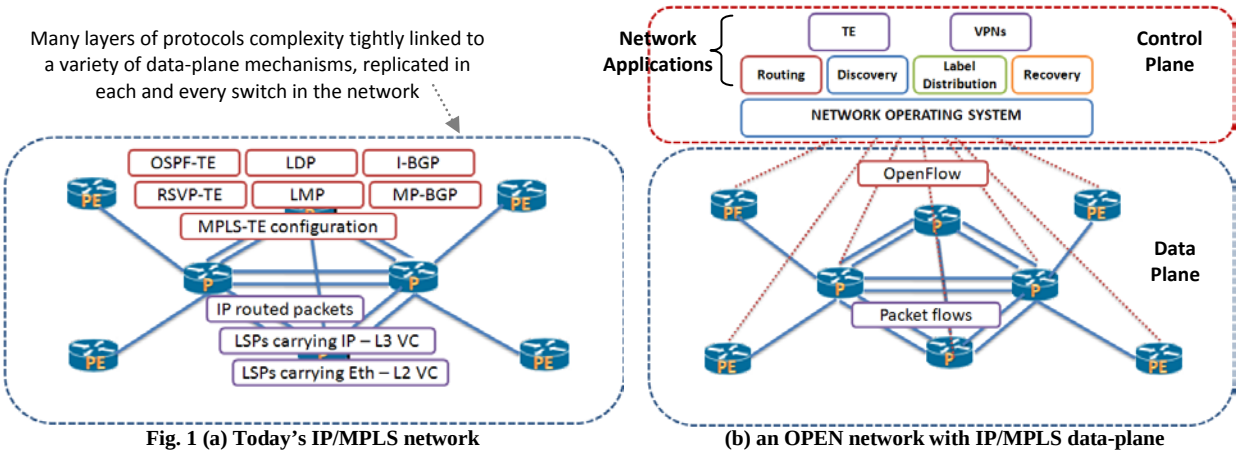

Fig. 1 (a) Today's IP/MPLS network **(b) an OPEN network with IP/MPLS data-plane**

† We have previously referred to such networks as Software Defined Networks (SDN)

2. IP/MPLS Control Plane vs. OPEN Control Plane

In today's packet switched networks, a router is both the control element which makes control decisions on traffic routing, as well as the forwarding element responsible for forwarding packets, and both these functionalities are tightly linked (Fig. 1a). Housing control and data functions in the same box necessarily complicates the equipment, which aside from making routing decisions and switching packets, also needs to have all the intelligence necessary for aggregating and disseminating routing information, using fully distributed routing protocols like OSPF or IS-IS. Furthermore, in IP/MPLS networks, another layer of complexity is added with the need for distributed signaling and label distribution mechanisms like RSVP-TE and LDP. Within a domain, an IP/MPLS network may additionally support a host of other protocols such as iBGP, RIP, LMP, SNMP and MP-BGP together with many more protocols for multicast and IPv6. All of these features contribute to control plane load, increased fragility and increased cost.

Distributed link-state routing protocols are fragile beasts when it comes to stability, where the latter can be defined in terms of convergence time, routing load on processors as well as the number of route flaps. If there are frequent changes in the network state, and update messages are sent frequently (sometimes called churn), then the router CPU may spend a lot of time doing route calculations, leading to ingress queues getting filled and incoming packets getting dropped (including Hellos). Dropping Hellos may in turn cause timeouts and dropping of routing adjacencies, which in-turn leads to even more control packets and greater CPU load. This cascading effect results in large convergence times, routing loops while converging, and in the worst case, total network meltdown. To avoid routing protocol instability, router vendors apply several damping mechanisms that require careful tuning and tweaking to keep things under control. Needless to say, an IP network is operated *very* carefully. In MPLS-TE networks, the same routing protocols are extended to advertize even more information about links, and it gives even more reasons to generate control traffic as link-state changes (with bandwidth reservations). Again more dampening mechanisms like timers and thresholds are employed to keep things stable, at the cost of having stale link-state information in routers. The possibility of having stale information when performing a CSPF (constrained SPF) calculation for a TE-LSP, in-turn necessitates the use of a distributed signaling mechanism like RSVP-TE.

RSVP-TE is a poor choice for a TE signaling protocol because it extends classic RSVP with many features (baggage) not intended for TE. As a protocol, RSVP's soft-state nature requires frequent refreshes to keep state alive, and since it runs directly over IP (instead of TCP), it needs its own reliable delivery mechanism. Both features result in lots of control plane messages and switch CPU load. Furthermore, RSVP has been overburdened with many more features such as hierarchical LSP setup, point-to-multipoint LSP setup, LSP stitching, FRR setup and GMPLS extensions, making an already complex protocol more bloated. In short, we believe the MPLS control plane can benefit from alternate designs.

With OPEN, we propose the use of MPLS forwarding elements as data plane switches that connect to a logically centralized Controller using the OpenFlow protocol. The Controller runs a network-wide OS and a set of network control applications to realize *all* the functionality of existing intra-domain protocols mentioned above, thereby *eliminating their need* in an OPEN architecture. We believe that a new network-feature/application/service-provider-need should *not* equate to a new protocol or an extension of an existing one. Such a process can easily take 5-10 years, by which time the 'need' may change, the resultant protocol may be very different from the original idea, and several vendors may have already produced multiple, non-interworking, 'pre-standard' implementations.

The OpenFlow protocol allows discovery of the network topology by the Controller, and keeping the topology and network-state updated via statistics, status and error messages. OpenFlow also provides mechanisms to manipulate each data-plane switch's flow-table. The network OS based applications can then make all network-control decisions, when presented with the network topology, state and the power to classify traffic into 'flows' and control switch forwarding. The controller and its applications can decide how each flow is forwarded (reactively as new flows start, or proactively in advance), how it is routed, which ones are admitted, where they are replicated, (optionally) the data-rate they receive and more. They can then cache the decision in each data plane's flow table via various types of actions that OpenFlow allows on flows (e.g. forward, multicast, drop, push/swap/pop tags etc.). And so it is the control plane network-applications that determine access control, routing, multicast, load-balancing, tunneling and so on, eliminating the need for fully distributed routing and signaling protocols. Importantly the need for creating a new protocol for every new service/feature is also eliminated – all that is needed is a new application.

The logically-centralized nature of the Controller implies that while the decision making is performed in a centralized manner, the Controller *itself* is distributed over multiple-physical servers for fault-tolerance and performance. Onix is an example of a distributed network OS that claims to have the necessary scale, performance and reliability [3]. We believe that over the next few years, more network OSes with different design tradeoffs will appear for research and commercial use. In the next section, we demonstrate the ease with which, given such a network OS, we can introduce a new network service such as MPLS-TE, into an existing OPEN based IP network.

3. MPLS-TE Prototype System and Demonstration

We *emulated* an OpenFlow-enabled wide-area MPLS data plane in software and prototyped MPLS-TE applications on a network OS to demonstrate and experiment with our approach. The main components of our system are:

- Open vSwitch (OVS [4]): An open-source OpenFlow-enabled software switch, modified to perform MPLS data-plane functionalities.
- Mininet [5]: An open-source tool that creates a network of interconnected software switches on a single PC for emulating networks. Our system runs 12 instances of OVS in Mininet to form the MPLS data-plane.
- NOX [6]: An open-source network OS, modified to work with the MPLS changes to the OpenFlow protocol.
- MPLS API: An API we developed on top of NOX to create, modify and manage LSPs by distributing labels and obtaining network state and statistics via the OpenFlow protocol.
- GUI and traffic-generators: We extended an existing GUI to show network state in real-time (Fig. 2) and used software traffic generators to generate http, voip, and video traffic in our network (color-coded in the GUI).

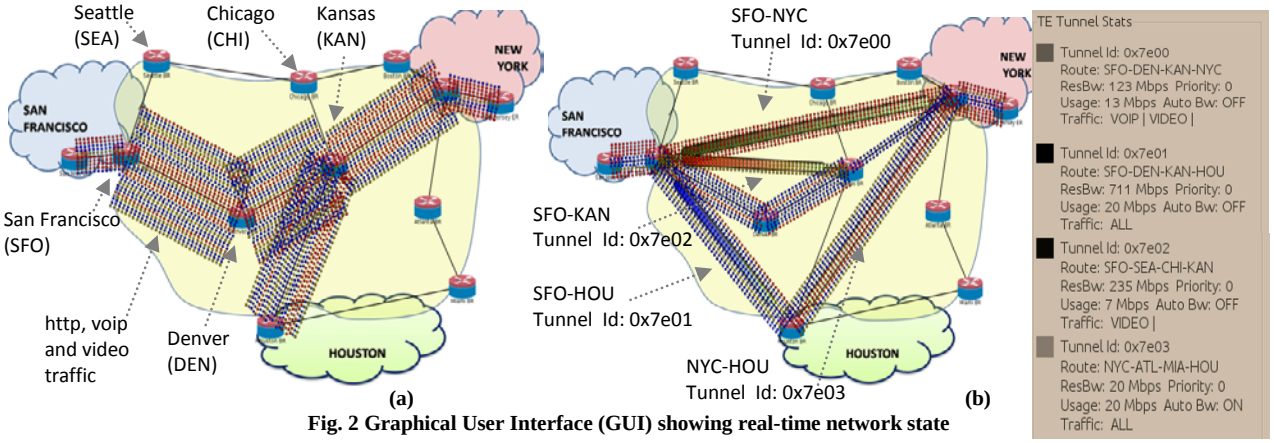

Fig. 2 Graphical User Interface (GUI) showing real-time network state

With such a prototype system, we demonstrated MPLS Traffic Engineering, using regular LSPs in the data plane together with tunnel-like (or TE-LSP) features in the control plane (see Tunnel Stats panel in Fig. 2b). Without TE in Fig. 2a, all traffic from San Francisco (SFO) to Kansas (KAN), New York (NYC) and Houston (HOU) share the SFO-DEN and DEN-KAN links. But with TE, we can reserve bandwidth ('ResBw') resources on links at different priority levels, perform CSPF to find shortest paths for tunnels that meet all constraints, and demonstrate admission-control for new TE-LSPs. In Fig. 2b, the SFO-HOU and SFO-NYC tunnels together reserve nearly all the available bandwidth (1 Gbps) on the links over which they are routed, prompting the SFO-KAN tunnel to be routed over the Seattle-Chicago links. We have shown that incoming traffic can be routed into tunnel or IP links, and traffic that was previously using the regular IP links can be auto-routed into newly created TE-LSPs. The user can specify a tunnel 'Priority' used in CSPF to preempt tunnels of lower priority, as well as the type of 'Traffic' the tunnel can carry (eg. the SFO-KAN tunnel carries only Video traffic). Finally we have shown the auto-bandwidth ('Auto Bw') feature, whereby tunnel ResBw adjusts dynamically to track the 'Usage' of the TE-LSP (eg. for NYC-HOU tunnel).

4. Summary

We proposed the use of the IP/MPLS data plane with an OPEN control plane, and demonstrated a traffic-engineering application in a prototype software system. We aim to replicate this demonstration on OpenFlow enabled switch hardware that supports the MPLS data plane, and demonstrate other MPLS control capabilities like FRR and VPNs. This effort required less than two months of work by two graduate-students with about 3500 lines of code added to NOX, OVS, the TE application and the GUI. More details including reference code and directions to re-create the demo are available in [7]. With this platform, we expect that any new features like FRR or VPNs would require no more than a few hundred lines of code and a few weeks of effort, a feat hard to achieve otherwise.

5. References

[1] OpenFlow protocol: http://www.openflowswitch.org/wp/documents/
[2] N. McKeown, et al., "OpenFlow: Enabling Innovation in Campus Networks", SIGCOMM CCR, Vol. 38, Issue 2, March 2008.
[3] T. Koponen et al., "Onix: A Distributed Control Platform for Large-scale Production Networks", OSDI, October 2010
[4] Open vSwitch information: http://openvswitch.org
[5] B. Lantz et al., "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks", ACM HotNets-IX, October 2010.
[6] N. Gude et al., "NOX: Towards an Operating System for Networks", ACM, SIGCOMM CCR, July 2008.
[7] http://openflowswitch.org/wk/index.php/OPEN-MPLS