Enabling an SDN Approach to Multi-Layer Network Provisioning and Optimization

Chris Liou1, Abhinava Sadasivarao1, Sharfuddin Syed1
Henrique Rodrigues2, Andy Lake2, Chin Guok2, Inder Monga2, Eric Pouyoul2

1Infinera Corporation, 140 Caspian Court, Sunnyvale, CA 94089 USA
2Energy Sciences Network, 1 Cyclotron Road, Berkeley, CA 94720 USA
cliou@infinera.com

Abstract: The collaboration effort in developing a Carrier SDN demonstration is described, demonstrating provisioning and optimization of bandwidth services across a multi-vendor network comprising router and transport networking layers.

1. Introduction
With the recent advances in converged transport networks that integrate OTN switching along with optical super-channel technologies, carriers are now looking to extend the SDN concepts of network programmability and centralized control to multi-vendor, multi-layer IP/Optical networks, and leveraging SDN to facilitate the operationalization as well as optimization of resources within these next-generation networks [1]. While much of the industry focus has been on enabling SDN for packet systems, many of the same concepts are now under study by organizations such as the ONF and OIF, and are being logically extended towards the converged optical transport layer [2,3]. The objectives of Carrier SDN are multifold:

- Enable programmability of the intelligent transport layer and leverage its increasing ability to switch and groom transport bandwidth over optical resources, as well as switch optical capacity
- Virtualize network resources and support a simple abstraction for provisioning bandwidth services
- Simplify, orchestrate, and automate provisioning operations in a multi-vendor, multi-layer environment
- Enable the improvement of overall network resource utilization across multiple network layers
- Speed the delivery of new services and rapidly deliver the bandwidth on-demand to support these services, wherever and whenever needed

This paper describes the collaboration effort in developing a Carrier SDN demonstration, demonstrating provisioning and optimization of bandwidth services across a multi-vendor network comprising router and transport networking layers. In this demonstration, we leverage a common SDN framework and ESnet’s On-demand Secure Circuits & Advance Reservation System (OSCARS) multi-layer provisioning application to coordinate and dynamically re-optimize traffic flows across both network layers, using OpenFlow as the unifying control plane.

2. Multi-Layer Carrier SDN Demonstration Architecture
In the demonstration, an SDN framework architecture was developed that leveraged OpenFlow 1.0 as the control plane wire protocol for provisioning both the transport network as well as the multi-vendor router/switch layer, as illustrated in Figure 1.

Fig 1. Multi-layer SDN Framework Architecture based on OpenFlow and RESTful APIs.
The transport layer comprised Packet Optical Transport Networking (P-OTN) systems equipped with optical super-channels and an OpenFlow enabled virtual transport switch called Open Transport Switch (OTS) that presents a digital abstraction to the SDN Control Layer and supports topology, management, configuration, monitoring, and provisioning. The packet layer was implemented using various routers, including a terabit class OpenFlow enabled router, and a smaller multi-layer OpenFlow switch/router, and utilized a mix of 10GbE and 100GbE interfaces. Floodlight comprised the SDN Controller, providing OpenFlow 1.0 protocol support, and ESnet OSCARS was used as the multi-layer service provisioning system, presenting the centralized view of the multi-layer network.

3. Transport Layer SDN Implementation

For the transport layer, a set of southbound OpenFlow based APIs was defined for supporting the OpenFlow wire protocol as well as a full set of OTS management and configuration functions, including topology management & discovery, lifecycle administration (instantiation, initialization and configuration), multi-tenancy for enabling L1 Optical VPNs, and transport layer data plane provisioning. An information model representing the abstraction of a transport switch was defined and implemented, and each OTS’s managed object instances were persisted using a lightweight SQL-based data store residing natively on the converged optical transport system. The configuration and management API was implemented via a RESTful interface over HTTP, with JSON as the object encoding scheme.

Each OTS instance presents not just an abstraction of the virtualized resources via its information model, but also serves as the mediator between the virtualized transport layer and the physical layer. Logical resources are defined and bound to physical resources by the network administrator during the instantiation and configuration phases. Once the OTS topology is configured, subsequent data path provisioning operations issued to OTS are performed solely based on the virtual network overlay. This virtual network topology was defined using abstracted objects to represent OTS nodes, links and ports.

Data plane provisioning is demonstrated via two distinct southbound APIs, as illustrated in Figure 2:

1. **OpenFlow 1.0 protocol:** OpenFlow 1.0 was employed for transport layer provisioning. The logical port concept was leveraged to represent points of data plane connectivity through OTS, and the ofp_flow_mod request was utilized for affecting data plane changes. Additionally, several other aspects of OpenFlow were implemented, including ofp_hello, ofp_switch_features, ofp_phy_port, ofp_stats_request, ofp_error_msg, and asynchronous updates.

Two different types of transport layer data plane services were implemented within OpenFlow:

a. **Direct mode:** this mode supported nodal-level transport flow control, leveraging the in_port and out_port parameters to uniquely identify 2 local logical ports for cross-connecting. Knowledge of the end-to-end path at the transport layer was centralized and computed in OSCARS.

b. **Implicit mode:** this mode supports domain-level transport flow control, leveraging GMPLS. In this mode, additional information was required to identify the source and destination endpoints of the end-to-end circuit. Other OpenFlow fields that were IP-centric and irrelevant to transport systems were utilized for conveying remote endpoint information.

2. **OTSConfig RESTful API:** the other interface implemented in the demonstration was a RESTful API that supported extended capabilities associated with the GMPLS-based implicit provisioning mode that were not demonstrable via the vanilla OpenFlow control protocol, including extended traffic engineering controls.

![Fig. 2. OTS multi-agent architecture comprising OpenFlow and REST APIs.](image-url)
4. Packet Layer SDN Implementation

The implementation of control of the packet layer leveraged native OpenFlow support on the Brocade and NEC switches. Floodlight was instantiated in a VM running on a Linux platform. The native OpenFlow implementations on the routers provided immediate support for the features needed for the demonstration – OpenFlow handshaking provided the means for Floodlight to initiate communications. For the demonstration, all switch ports were OpenFlow enabled.

As part of the multi-layer demo, connectivity between L2/L3 OpenFlow switches was created on-demand via OSCARS. In scenarios where no L2/L3 connectivity exists, such as at the beginning of the demonstration, OSCARS creates the transport connections first, providing transport bandwidth to support packet flows. OSCARS then engineers the packet flows for supporting the big data applications and implements them via a RESTful API to Floodlight.

For demonstrating dynamic multi-layer optimization of elephant flows, a Traffic Optimization Engine was developed which monitors the flow statistics on the source terabit class router via Floodlight and, based on some decision thresholds, triggers the reoptimization of large flows through OSCARS. The implementation of the reoptimization process includes tearing down the original packet flow, setting up a new 100G transport level circuit using GMPLS-aided implicit mode provisioning, and then reestablishing the elephant flow over this newly created circuit. This could also eventually be done by redirecting packet flows to another port via SDN without tearing down the connection.

5. Results

This prototype Carrier SDN solution successfully demonstrated how SDN can be utilized to address provisioning and optimization challenges of multi-vendor, multi-layer networks, and enabled the demonstration of the following use case scenarios

1. **Transport Layer Network Virtualization and Optical VPN**: demonstrated the dynamic instantiation, configuration, and lifecycle management of OTS and virtual transport network overlays using a RESTful API.
2. **Automated Virtual Network Topology Discovery**: demonstrated centralized topology discovery of the OTS virtual networks via the RESTful OTSConfig API and supplied that information to OSCARS.
3. **On-demand Multi-layer Provisioning**: demonstrated single-step provisioning of bandwidth services spanning the multi-layer network, by orchestrating and triggering provisioning activities at both the packet and supporting optical transport layers.
4. **Flow optimization across a Multi-Layer network**: demonstrated how existing packet flows can be re-optimized based on real-time monitoring of flow characteristics such as total bandwidth and re-provisioned through the network to improve resource utilization as well as economics.
5. **Co-existence of Multiple Provisioning Modes & Interfaces**: demonstrated how both centrally engineered and provisioned transport layer crossconnects as well as GMPLS-enabled end-to-end bandwidth services can simultaneously be supported without conflicts, enabling a more flexible environment for operators seeking to transition towards an SDN architecture while preserving GMPLS capabilities inherent to their transport domains.

References

