

# Open Transport Switch - A Software Defined Networking Architecture for Transport Networks

Abhinava Sadasivarao\*

Sharfuddin Syed\*

Ping Pan\*

Chris Liou\*

Andy Lake†

Chin Guok†

Inder Monga†

\*Infinera Corporation  
Sunnyvale, CA 94089

{asadasivarao, ssyed, ppan, cliou}@infinera.com

†Energy Sciences Network  
Berkeley, CA 94720

{andy, chin, inder}@es.net

## ABSTRACT

There have been a lot of proposals to unify the control and management of packet and circuit networks but none have been deployed widely. In this paper, we propose a simple programmable architecture that abstracts a core transport node into a programmable virtual switch, that meshes well with the software-defined network paradigm while leveraging the OpenFlow protocol paradigm for control. A demonstration use-case of a OpenFlow-enabled optical virtual switch implementation managing a small optical transport network for big-data applications is described. With appropriate extensions to OpenFlow, we discuss how the programmability and flexibility SDN brings to packet-optical backbone networks will be substantial in solving some of the complex multi-vendor, multi-layer, multi-domain issues service providers face today.

## Categories and Subject Descriptors

C.2.1 [Computer-Communication Networks]: Network Architecture and Design—*Circuit-switching networks*; C.2.3 [Computer-Communication Networks]: Network Operations—*Network management*

## General Terms

Design, Standardization

## Keywords

transport networks, optical networks, virtualization

## 1. INTRODUCTION

Significant advances in optical technologies, bit rates and deployment of Optical Transport Network (OTN) protocols have enabled transport networks to provide flexible multiplexing and switching functions in addition to basic data transport and survivability. In ad-

dition, transport network elements are being supplemented with more intelligent set of features for flexible management. The growth in traffic volumes, changing traffic profiles and types of applications has prompted service providers to rethink not only how to engineer their IP and optical backbone transport optimally, but also to ease their operational and management overhead.

In the Internet core, traditionally, the design approach has been to place all the network functions within the IP layer (routing, signaling, protection) and use static optical trunks interconnecting these L2/L3 devices. This hop-by-hop architecture of packet processing and forwarding can be optimized significantly by taking advantage of the dynamic transport capabilities offered by the state-of-the-art optical network. In addition, service providers typically manage their L3 networks and transport layer operations independently. In this multi-layer setup, provisioning bandwidth involves multiple steps: creating necessary interfaces and forwarding entries in the L3 devices and provisioning circuits in the transport networks, closing the end-to-end path. The distributed nature of the protocols needs UNI and NNI signaling to happen before the actual datapath is complete. This approach adds complexities to the transport control plane mechanisms (GMPLS [5]/MPLS [4]/MPLS-TP [6]).

The latest trends in application delivery architectures, like cloud computing and consolidated data-centers, are aggregating the user traffic and also creating large flows between such data-centers for data and state synchronization. The need for cost and performance optimization as well as creation of new network services relevant to the above applications is driving the requirements for dynamic multi-layer and multi-domain networking.

Multi-layer optimization, with applications such as dynamic router bypass, does not only have technology drivers, but also influences CapEx economics. Even though the advantages of such approaches are well understood and protocols created by the vendor community, the complexity of protocols, vendor interoperability and lack of management tools has prevented these from being deployed.

Software-Defined Networking (SDN), that decouples the data plane from control plane, has been discussed recently [2] as a viable and simple approach to provide the required functionality. The approach promises meeting the manageability, flexibility, and evolvability requirements in large service provider networks. Although, much of SDN efforts today are concentrated on networks at Layer 2 and above. Many vendors have added OpenFlow capabilities to their Gigabit Ethernet switches. There have also been efforts in building hardware architectures [11] and switch fabrics for efficient OpenFlow enabled network devices [1]. OpenFlow based enterprise wireless network management has also been proposed [13]. All these are Ethernet/IP centric.

In this paper we propose a virtual abstraction of the transport element, Open Transport Switch (OTS), that integrates within a SDN framework and offers simple OpenFlow protocol based control of the packet-optical cross-connect and bandwidth allocation capability of the optical element. In addition, we showcase a prototype implementation of this abstraction and deployment at a test network in Long Island. We show SDN as a viable approach for building wide-area packet-optical networks.

## 2. ARCHITECTURE

The central approach *abstracts* the interface between packet and circuit layers leading to *virtualization* of the transport layer. Let us consider (Fig. 1) - a common scenario found in service provider networks. The network is segmented into various layers each running their own control plane for routing and signaling. Each layer may have equipment from different vendors. Multi-layer integration becomes a challenge as 1) GMPLS protocols for multi-layer require UNI relationship which hides each layer's topology (Fig. 2) 2) Multi-vendor implementations of GMPLS protocols and path-finding are fairly different with interoperability at a least common denominator of functionality 3) Different EMS/NMS systems are ultimately used to manually manage each vendor, leading to a static, pre-planned network solution.

On the other hand, the applications at the edges of these networks require high-bandwidth paths for ex-

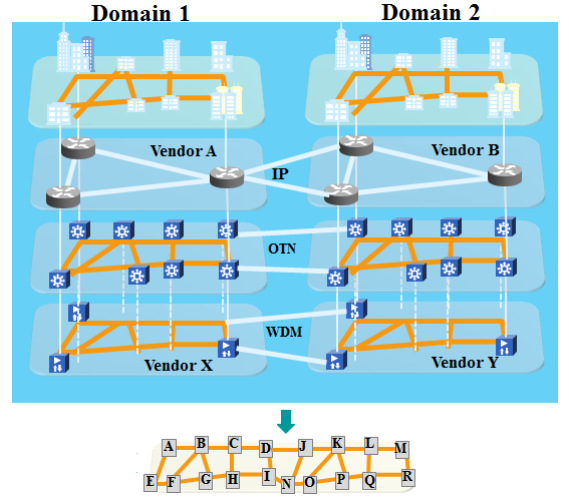


Figure 1: Multi-Domain, Multi-Layer

change of data, for example data center interconnects. These connections require connectivity and varying amounts of bandwidth, irrespective of the protocols used to transport the information. The underlying transport infrastructure could be packet/MPLS, OTN or MPLS-TP. If the resources viz. ports, links and bandwidth can be virtualized with generic abstractions, the applications would need to program this **virtual overlay network** of devices interconnected by links (Fig. 1). The network then truly becomes open, programmable and flexible.

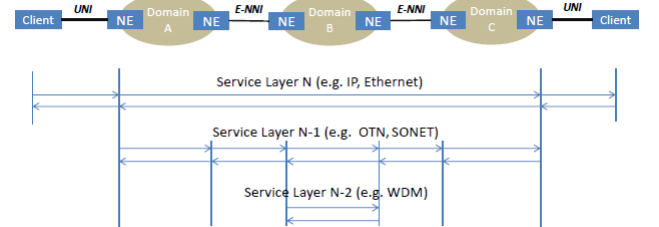
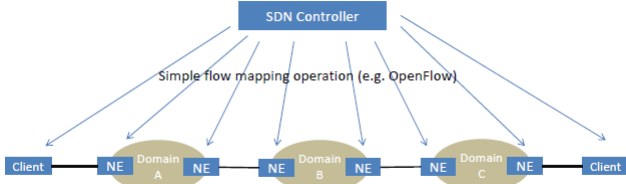


Figure 2: Service Provider Transport Network

*Open Transport Switch (OTS)* is an OpenFlow [12] enabled light weight, virtual switch that represents a transport network element (NE). Applications can now talk to an SDN controller to request provisioning of cross-connects or aggregation of packet interfaces into optical trunks with the required capacity and relevant QoS parameters, if needed. This gives service providers the ability to create a unified view of the network (Fig. 3). These abstractions remove the need for control plane interactions at packet-optical layer transitions. The SDN Controller offers the virtualized topology to smart applications enabling them to perform optimal path computation, provisioning and monitoring based on their constraints. Applications not interested in their

own path-computation can request the bandwidth capacity and QoS, outsourcing the end-to-end path computation to a specialized carrier controller/application that can match the request for the end-to-end path across multiple domains/layers to meet the requested SLA.



**Figure 3: SDN Enabled Transport Network**

Fig. 4 shows the building blocks of OTS. Assuming the Controller can be reached at a well known location, the OTS agent can advertise the port, link and other resource information to the Controller as part of the handshaking. The *Discovery Agent* is responsible for discovery and registration of SDN-controlled resources. It appropriately notifies the Controller dynamically as and when the NE and/or the Network state changes (for example, link up/down). In OpenFlow, this typically happens via the switch `OFPT_FEATURES_REQUEST`, `OFPT_FEATURES_REPLY` and other related `Modify State` messages [12]. How the discovery agent retrieves this information from the NE is upto the implementation or via proprietary vendor interfaces.

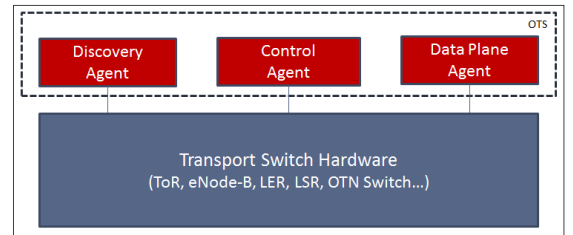
The *Control Agent* is responsible for monitoring and propagating notifications and alarms to the Controller, allowing network admins to monitor performance, faults and alarms in the network. These include change notifications for any new equipment/facilities provisioned/de-provisioned. Loss-of-light, Loss-of-sync, Loss-of-signal are some examples of alarms. Faults could range from link failures to equipment failures. (Note that some of equipment related alarms could be reported by both the Control and Discovery agents). This way, the controller's state is asynchronously (or synchronously) kept consistent with the state of the underlying network. The *Dataplane Agent* is responsible to program the NE datapath to create/update/release circuits/LSP. The datapath entities could be Time slots, Cross connects (XCON) or MPLS labels. This programs the underlying network infrastructure and helps complete the datapath. The controller sends appropriate OpenFlow messages (similar to `OFPT_FLOW_MOD` message). Again, how the Data-plane Agent programs the particular NE database/forwarding tables could be through vendor specific interfaces.

The northbound interface from OTS to the Controller

is OpenFlow 1.0.0 [12]. Given that OTS is virtualizing transport NE, much of the Ethernet centric OpenFlow messages are not used. With addition of extensions (see sections 2.1 & 3), the Controller can send requests to OTS to provision/release transport circuits.

OTS being a virtual switch has multiple advantages associated:

- OTS is stateless or very minimally stateful: All the alarms, stat counters, forwarding table entries are stored in the NE database and could be retrieved by OTS. OTS need not maintain these managed objects. This also allows for OTS to be light on resources.
- OTS is lightweight and portable: Given that most of the state is maintained by the NE, if the southbound interface from the OTS agent to the NE is flexible enough (implementation and vendor specific), the OTS agent could be made to run on a standalone server or EMS or any other machine which can communicate and maintain an active OpenFlow session with the Controller.
- OTS Southbound Interface: The southbound interface from the OTS agent to the NE could also be standard interfaces which allows OTS implementations to be multi-vendor ready. For example, TL-1 [14] is a widely used network element management protocol amongst RBOCs. Other examples include NETCONF [7]
- Multiple OTS agents could be run on the same NE. These different instances can be given a restricted view of the ports/wavelengths present on the NE and manage their respective resources only (See section 3.2)



**Figure 4: OTS Building Blocks**

## 2.1 OpenFlow Extensions

OpenFlow [12] is very L2/L3 centric as of today. Given that we want to control optical transport equipments, the protocol currently has no understanding of circuit switching constructs like time-slots or cross-connects (XCON). We are proposing extensions to OpenFlow by adding messages that allow provisioning/release of

circuits. Given that we are virtualizing the network, we use opaque, MPLS-style labels to represent links (sequence of ingress/egress ports). We also indicate the style of circuit that needs to be setup (see section 2.2). Along with these, the message includes service rate and latency parameters along with provisioning actions (ADD\_XCON and REM\_XCON). For now, we assume the type of service/traffic to be GbE. However, this could be OC-192, OTU3, Fibre Channel and so on.

```

struct ofp_id {
    // Host ID - DCN IP Address of the Node
    uint32_t node;

    // Flow ID maintained by the Controller
    uint32_t label;
};

struct ofp_xconn {
    struct ofp_header header; // OFPT_VENDOR
    uint32_t vendor; // Vendor ID

    uint8_t pad[4];

    struct ofp_id src; // Source of the flow
    struct ofp_id dst; // Destination of the flow

    uint32_t rate; // Rate of service (Mbps)
    uint8_t latency; // Latency - 0 to 255
    uint8_t style; // Implicit = 1 Explicit = 2

    // Unidirectional = 1 Bidirectional = 2
    uint8_t directional;

    uint8_t pad_extra[1];

    // ADD_XCONN = 0xFF REM_XCONN = 0xFE
    struct ofp_action_header actions[0];
};
OFP_ASSERT(sizeof(struct ofp_xconn) == 40);

```

## 2.2 Modes of Operation

We already described how SDN for transport can provide an alternative to interworking UNI/NNI protocols associated with distributed routing and signaling. Integrating OTS into today's large service provider transport networks may become a very complex exercise (we are infact trying to make transport networks more flexible and manageable!). Taking this into account, we propose two modes of operation to allow smooth integration of, and transition to transport SDN.

### 2.2.1 Explicit Mode

Fig. 5 depicts *Explicit Mode*. In this mode, the Controller has the knowledge of every NE in a particular domain. After optima/constrained path computation, provisioning a circuit becomes an exercise of the Controller programming all the transport devices along the path in a hop-by-hop manner across single or multiple transport domains.

### 2.2.2 Implicit Mode

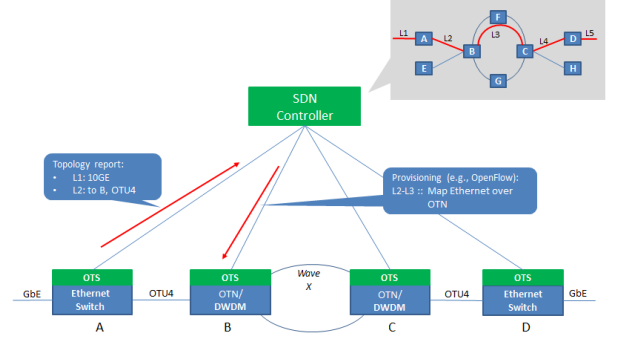


Figure 5: Transport SDN Explicit Mode

Fig. 6 depicts *Implicit Mode*. In this mode, the Controller is aware of only the edge nodes in every transport domain (Ethernet/MPLS/OTN). Within the domain, the existing routing and signaling control plane can be used to setup path. The Controller would send provisioning request, specifying the source and destination to the SDN-aware nodes at the edges of the network. The source node will then trigger MPLS/GMPLS control plane to setup the circuit. Controller being aware of NE type and capabilities, *stitches* these segments across multiple domains to form an end-to-end circuit. Implicit mode adds great flexibility in gradually incorporating OTS architecture into existing transport networks. Without disrupting current deployments, service providers may choose to continue using intra-domain control plane while still being SDN aware. From a Controller's perspective, this edge-to-edge intra-domain path appears as a single network fabric of a given capacity. Service providers depending on the necessary management effort, can gradually make all the NEs SDN capable, moving to an explicit deployment model.

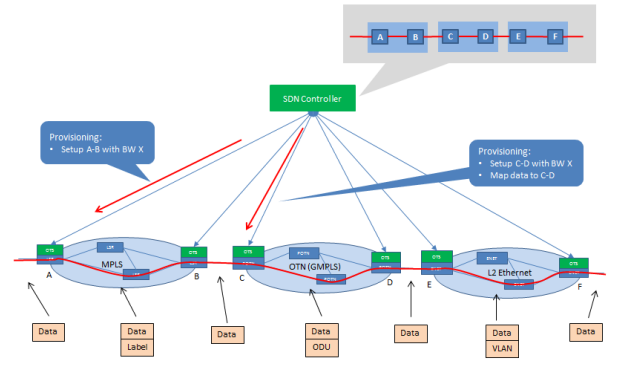


Figure 6: Transport SDN Implicit Mode

Note that these are not the only two feasible models in Transport SDN. In a given network, it is possible to have SDN circuits created with mix of implicit and explicit modes. This prototype demonstrates the larger

concept.

### 3. IMPLEMENTATION

Section 2 described the building blocks of OTS. The prototype OTS implementation only has the Dataplane agent functionality built in. Rest of the subsystems will be integrated in the future.

#### 3.1 Controller

*On-Demand Secure Circuits and Advanced Reservation System* (OSCARS) [3] is a provisioning system developed by Energy Sciences Network (ESnet). It provides multi-domain, high-bandwidth, virtual circuits that guarantee end-to-end network data transfer performance. Today, OSCARS virtual circuits carry about fifty percent of ESnet’s annual 60 petabytes of traffic, supporting large scale sciences such as High Energy Physics, Climate, Computational Astrophysics, and Biological and Environmental Research.

The OSCARS system in this instance acts as a typical SDN controller, albeit with several specific extensions (see 2.1) to leverage the transport OpenFlow capabilities of the OTS. The initial handshake between OSCARS and OTS involves the characteristic exchange of `OFPT_HELLO` followed by `OFPT_FEATURES_REQUEST` and `OFPT_FEATURES_REPLY` messages [12]. However, OpenFlow extension messages as described above are sent for optical specific information.

#### 3.2 OTS Agent - Virtual Switch

The prototype OTS agent implementation is coupled with the Infinera DTN [9], which embodies the optical transport NE. The DTN has fully flexible OTN, SONET/SDH and Ethernet add/drop capabilities with OTN [10] line side wavelength Optical Carrier Groups (OCG), and includes an embedded GMPLS control plane for end-to-end routing and provisioning. As part of the management suite, the DTN provides TL-1 interface access to the NE for troubleshooting, alarms and event notifications, and equipment, fault and performance monitoring. The OTS agent leverages this TL-1 interface to communicate southbound with the NE, while exposing a northbound OpenFlow interface. This affords the flexibility of running the OTS instance outside the NE, as well as managing an NE of a different vendor.

To determine the tributary ports and wavelengths to be used, a simple manually edited configuration file is referenced. Given that these are TDM circuits, the incoming traffic payload is digitally wrapped/containerized into OTN optical channels and transported. There is no header or label lookup done on incoming traffic, and therefore the configuration file itself is sufficient as a simple flow table. In addition, the configuration file

can specify “slices” of an NE that can be virtualized into separate tributary port/line side wavelength resources controlled by distinct OTS agents. This allows service providers to sell wholesale bandwidth to multiple third party/tier-2 providers, who can only control their portion of the equipment/resources.

### 4. RESULTS AND OBSERVATIONS

#### 4.1 Network Setup

We made use of ESnet’s Long Island Metropolitan Area Network (LIMAN) to demonstrate SDN controlling the transport optical backbone through OSCARS and OTS. Fig. 7 shows the setup. DTN nodes A and B are SDN aware and two OTS agents were managing each of these NEs. The nodes connected to ESnet’s production network via DWDM fibers. There also exists a direct fiber connectivity between the two nodes. There are two hosts with 40G Ethernet interfaces connected to each of these nodes. We setup two 40G circuits between node A and B, one over the direct (one-hop) fiber link and another over the production node. The fiber path traversed by the circuit is transparent to the end hosts. The end hosts will see a direct one-hop IP Link connecting each other.

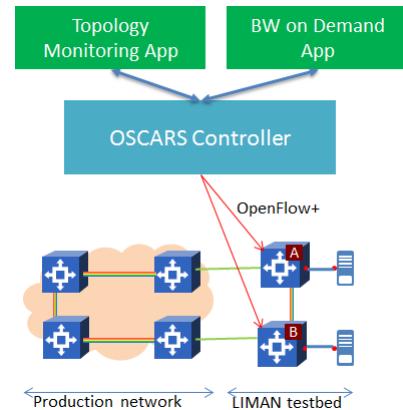


Figure 7: Network Setup

#### 4.2 Measurements

The measurements done were to reserve a 40G circuit from Node A to B (Fig. 7). This only includes the time taken by the Controller to compute the path and to send the `ofp_xconn` message. The time can be further improvised subject to making use of faster processing platform for the Controller. This metric was specifically used as we attempt to highlight the time it takes to setup the path using SDN Controller which helps to contrast with the time taken in case of a non-SDN approach. Further, the amount of time it takes to setup a circuit at the NE level remains the same irrespective of SDN or non-SDN approach.

Mode	Min	Max	Mean	Std. Dev
Implicit	2	7	2.84	0.98
Explicit	2	5	2.95	0.87

**Table 1: Circuit path computation latencies (s)**

Given a fairly simple topology, the maximum latency observed is for the first circuit setup request. OpenFlow session needs to be established with the OTS agents and hence the higher latency. Once the session is active, the time involved is for the Controller to compute path based on the knowledge it has about the topology. Since this is a prototype, most of the topology and node/link information is statically configured. The OTS Discovery agent is responsible to provide the Controller with the necessary topology and network resource information. This is part of the next phase of work.

## 5. SCOPE FOR FUTURE WORK

There are several additions to OTS that could provide full network virtualization capabilities. From an implementation perspective, we wish to fully integrate the Monitoring and the Discovery agents into OTS for fault-/alarm propagation and port/link discovery respectively. Currently, the ports, optical channels and links are hand-configured through a configuration file for this prototype implementation. But we would need a dedicated info model similar to Open vSwitch Database [8] to house these configuration information and advertise it to the Controller. This allows the Controller to discover the complete topology depending on the mode of operation (Implicit/Explicit). JSON encoded data could be used to exchange these topology extracts between OTS and the Controller.

From the point of view of standardization, many important functions that are inherent to core transport networks have to be factored in. For example, protection and restoration. Typically, these are part of the control plane (MPLS FRR or GMPLS restoration). Thorough investigations need to happen to determine if these have to be incorporated into OpenFlow. Or, these could continue to be a part of the underlying control plane as in the case of Implicit mode. Further, if domain specific parameters (like optical impairments, OSNR etc) are needed, these need not be a part of the protocol itself. Instead, a management interface like OFConfig or NETCONF can be used to extract these.

## 6. CONCLUSION

We have demonstrated a prototype virtual transport switch for optical long haul networks. OTS can be extended to a packet switched IP/MPLS (PSN) core backbone too. The idea of extending Software Defined Networking concepts to transport provides compelling

technical and economical advantages to large service providers to efficiently engineer, manage and evolve their networks. Network virtualization through OTS allows building an overlay network that applications can program to meet their specific bandwidth requirements irrespective of what underlying layers (L1/L2/L3) or technologies (OTN/MPLS/IP) are used. With addition of appropriate extensions to OpenFlow, the transport infrastructure can be made more open and programmable which allows multi-layer, multi-domain and multi-vendor optimizations in both core and metro networks. Efforts are already underway within Open Networking Foundation (ONF) to allow standardization of transport extensions to OpenFlow (NewTransport WG). We hope HotSDN acts as a catalyst and provides a forum for the discussion of these ideas.

## 7. REFERENCES

- [1] CASADO, M., KOPONEN, T., SHENKER, S., AND TOOTOONCHIAN, A. Fabric: a retrospective on evolving SDN. In *Proc. of HotSDN* (2012).
- [2] DAS, S., PARULKAR, G., AND MCKEOWN, N. SDN Based Unified Control Architecture. In *Photonics Conference (IPC), 2012 IEEE* (2012), pp. 778–779.
- [3] GUOK, C., ROBERTSON, D., CHANIOTAKISY, E., THOMPSON, M., JOHNSTON, W., AND TIERNEY, B. A User Driven Dynamic Circuit Network Implementation. In *GLOBECOM Workshops, 2008 IEEE* (2008), pp. 1–5.
- [4] IETF. RFC3031: Multiprotocol Label Switching Architecture, January 2001.
- [5] IETF. Generalized Multi-Protocol Label Switching (GMPLS) Architecture, July 2004.
- [6] IETF. RFC6215: A Framework for MPLS in Transport Networks, July 2010.
- [7] IETF. RFC6241: Network Configuration Protocol (NETCONF), June 2011.
- [8] IETF. ovsdb-draft-00: The Open vSwitch Database Management Protocol, August 2012.
- [9] INFINERA. DTN <http://www.infinera.com/products/dtn.html>.
- [10] ITU. G.709: Interfaces for Optical Transport Network, Feb 2012.
- [11] MOGUL, J. C., AND CONGDON, P. Hey, you darned counters!: get off my ASIC! In *Proc. of HotSDN* (2012).
- [12] PFAFF, B. OpenFlow Switch Specifications 1.0.0, Dec 2009.
- [13] SURESH, L., SCHULZ-ZANDER, J., MERZ, R., FELDMANN, A., AND VAZAO, T. Towards programmable enterprise WLANS with Odin. In *Proc. of HotSDN* (2012).
- [14] TELCORDIA. GR-199-CORE: TL1 Memory Administration Messages, Dec 2005.