CONFIDENTIAL INFORMATION
DESTROY BY SHREDDER ONLY

MPLS Interoperability

SDN

DRAFT

August 2012

Document ID:ITD-10349v1
Status: Draft

TABLE OF CONTENTS

1 Introduction

Networks have become a critical component of all infrastructures in society. However the industry and their designs have not kept pace with ever growing requirements. Networks are built using switches, routers, and other devices that have become exceedingly complex because they implement an ever increasing number of distributed protocols standardized by IETF and use closed and proprietary interfaces within. In this environment, it is too difficult, if not impossible, for network operators, third parties including researchers, and even vendors to innovate. Researchers cannot experiment with their ideas at any scale in a real network setting. Operators cannot customize and optimize networks for their use cases including the application set that is relevant to their business. Even vendors cannot innovate fast enough to meet their customer requirements.

The net result is that

(1) networks continue to have serious known problems with security, robustness, manageability, mobility that have not been successfully addressed;
(2) their capital costs have not been reducing fast enough and operational costs have been growing, putting excessive pressures on network operators;
(3) network operators find it difficult to introduce new revenue generating services on their expensive infrastructures.

Originating from the academic community, SDN is a relatively new concept which has formed multiple approaches all of which serve the common SDN agenda but do so in distinct fashions. Two approaches have risen to prominence with differences in pedigree and implementation making each applicable to different markets but not so much that they cannot be deployed together – creating a hybrid SDN solution.

- OpenFlow
  Born in university research labs, OpenFlow is defined in the recently formed Open Networking Foundation. OpenFlow removes the entire control plane from the network equipment relegating it to a data-plane only role. New mechanisms of network control (discovery, path computation, path setup etc...) are created and hosted on a server/cloud. Although applicable to telco/WAN, early work has focused on data center and campus applications. OpenFlow is a Layer 2 communications protocol that gives access to the forwarding plane of a network switch or router over the network. In simpler terms, OpenFlow allows the path of network packets through the network of switches to be determined by software running on multiple routers (minimum two of them — primary and secondary — has a role of observers). This separation of the control from the forwarding allows for more sophisticated traffic management than is feasible using access control lists (ACLs) and routing protocols

- PCE
  Standardized in the IETF [2], PCE takes an evolutionary approach and migrates only the path computation component of traditional networking devices to a centralized role. Much of the well-established and proven software functions of the control plane are left untouched and remain integrated within the NEs enabling a gradual migration to SDN. PCE has the added benefit of

providing inter-domain networking which is a key application for carrier networks. These attributes make PCE the preferred approach to SDN for telco/WAN environments.

SDN is a new approach to networking and its key attributes include: separation of data and control planes; a uniform vendor-agnostic interface called OpenFlow between control and data planes; a logically centralized control plane; and slicing and virtualization of the underlying network. The logically centralized control plane is realized using a network operating system that constructs and presents a logical map of the entire network to services or control applications implemented on top of it. With SDN, a researcher or network administrator can introduce a new capability by writing a simple software program that manipulates the logical map of a slice of the network. The rest is taken care of by the network operating system.

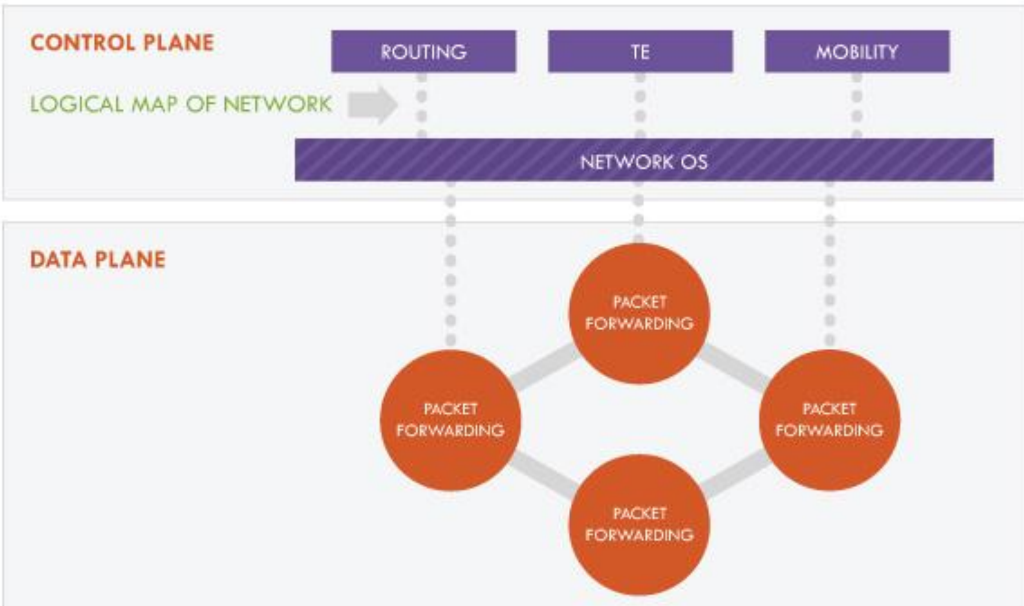

Figure1: Architecture view of SDN

Comparison of OpenFlow feature support across different Openflow version

Requests

	OpenFlow 1.0	OpenFlow 1.1	OpenFlow 1.2
Hello	Yes	No	Yes
Echo Request	Yes	No	Yes
Features Request	Yes	No	Yes
Get Config Request	Yes	No	Yes
Set Config Request	Ye	No	Yes
Packet Out	Yes	Yes	Yes
Port Mod	Yes	Yes	Yes
Flow Mod	Yes	Yes	Yes
Group Mod	N/A	No	No

Table Mod	No	No	No
Queue Get Config Request	N/A	No	No
Role Request	Yes	N/A	No
Barrier Request	Yes	No	Yes
Experimenter Request (Vendor Request)	N/A	No	No

Responses

	OpenFlow 1.0	OpenFlow 1.1	OpenFlow 1.2
Echo Reply	Yes	No	Yes
Features Reply	Yes	Yes	Yes
Get Config Reply	Yes	No	Yes
Queue Get Config Reply	N/A	No	No
Role Reply	N/A	N/A	No
Error	Yes	Yes	Yes

Actions

General Actions

	OpenFlow 1.0	OpenFlow 1.1	OpenFlow 1.2
Output (Forward)	Yes	Yes	Yes
Set-Queue (Enqueue)	Yes	No	No
Drop	Yes	Yes	Yes
Group	N/A	No	No

Push-Tag/Pop-Tag Actions

	OpenFlow 1.0	OpenFlow 1.1	OpenFlow 1.2
Push VLAN Header	N/A[*]	Yes	Yes
Pop VLAN Header	N/A	Yes	Yes
Push MPLS Header	N/A	Yes	Yes
Pop MPLS Header	N/A	Yes	Yes

[*] Similar behaviour may be achieved using the VLAN Id and VLAN priority Modify-Field actions.

Set-Field (Modify-Feild) Actions

	OpenFlow 1.0	OpenFlow 1.1	OpenFlow 1.2
In Port	N/A	N/A	N/A
Physical Port	N/A	N/A	N/A
Metadata	N/A	N/A	N/A
Ethernet Destination	Yes	Yes	Yes
Ethernet Source	Yes	Yes	Yes

Ethernet Type	N/A	No	Yes
VLAN ID	Yes	Yes	Yes
VLAN PCP (VLAN Priority)	Yes	Yes	Yes
Strip VLAN Header	Yes	N/A	N/A
IP DSCP (IPv4 ToS)	Yes	Yes	Yes
IP ECN (IPv4 ECN)	N/A	No	Yes
IP Protocol	N/A	N/A	Yes
IPv4 Source	Yes	Yes	Yes
IPv4 Destination	Yes	Yes	Yes
IPv4 TTL	N/A	No	N/A
Decrement IPv4 TTL	N/A	No	N/A
Transport Source Port	Yes	Yes	N/A[*]
Transport Destination Port	Yes	Yes	N/A[†]
TCP Source Port	N/A	N/A	Yes
TCP Destination Port	N/A	N/A	Yes
UDP Source Port	N/A	N/A	Yes
UDP Destination Port	N/A	N/A	Yes
SCTP Source Port	N/A	N/A	Yes[**]
SCTP Destination Port	N/A	N/A	Yes[**]
ICMPv4 Type	N/A	N/A	Yes
ICMPv4 Code	N/A	N/A	Yes
ARP Opcode	N/A	N/A	Yes
ARP SPA	N/A	N/A	Yes
ARP TPA	N/A	N/A	Yes
ARP SHA	N/A	N/A	Yes
ARP THA	N/A	N/A	Yes
IPv6 Source	N/A	N/A	Yes
IPv6 Destination	N/A	N/A	Yes
IPv6 Flow Label	N/A	N/A	No
ICMPv6 Type	N/A	N/A	No
ICMPv6 Code	N/A	N/A	No
ICMPv6 ND Target	N/A	N/A	No
ICMPv6 ND SLL	N/A	N/A	No
ICMPv6 ND TLL	N/A	N/A	No
MPLS Label	N/A	WIP	Yes
MPLS Traffic Class	N/A	WIP	Yes

[*] Similar behaviour may be achieved using the TCP, UDP and SCTP Source Port actions
[**] Pending completion of Open Flow 1.2
set-field encoder/decoder.

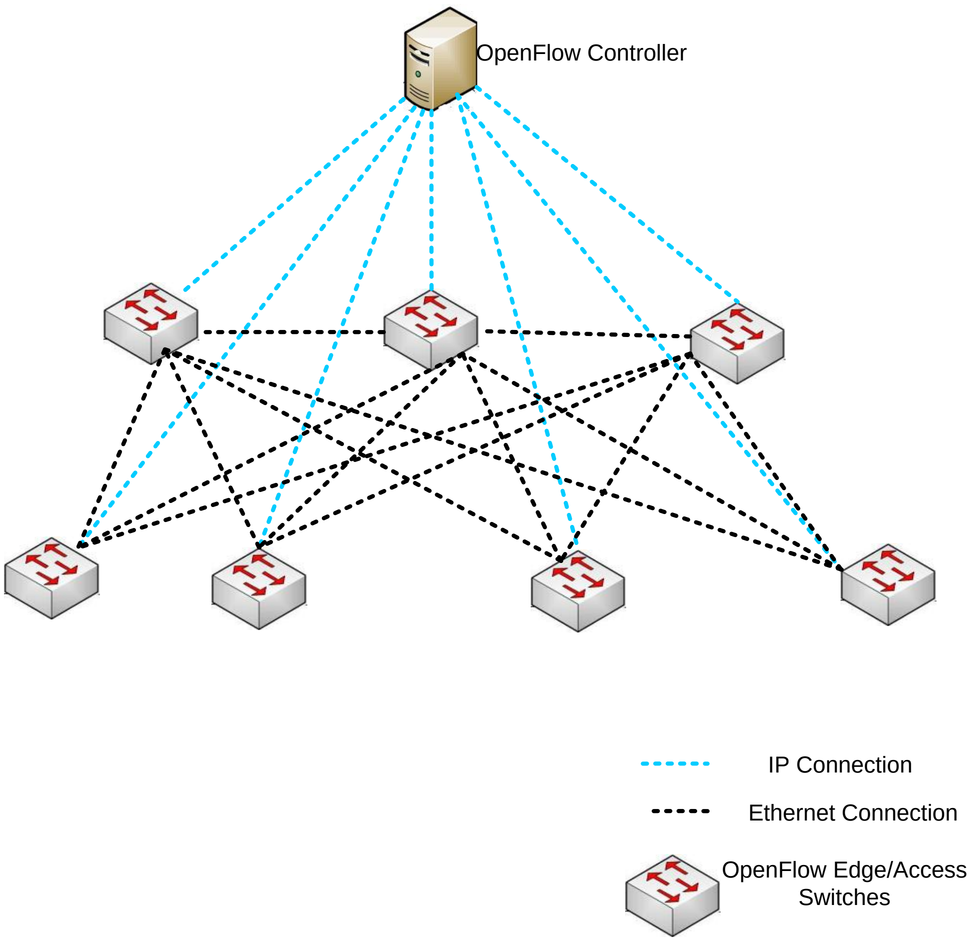

Figure 2: Reference testbed

2 Openflow 1.0

It is important to first perform some basic testing of the OpenFlow 1.0 feature set to ensure functionality.

2.1 Test Case – Network Discovery LLDP

A commonly implemented discovery mechanism uses the propagation of Link Layer Discovery Protocol (LLDP) packets. There is nothing really specific to LLDP, but it was chosen since it is a standard packet used to discover neighbors. The way it works in OpenFlow is as follows:

- A switch establishes a connection to the controller and responds to a feature_request with a feature_reply enabling the controller to learn the datapath_id (unique to each switch)
- The controller will install a flow entirely using the flow_mod that has a rule to match an LLDP packet and send the packet to the controller over the OF Channel via packet_in

- The controller then sends an LLDP packet containing the datapath_id and port number to the switch instructing it to send out all ports
- When the adjacent switches receive the LLDP packet they will send them up to the controller
- Receiving the LLDP packet back enables the controller to learn connectivity between switches and ensure there is a connection in each direction.

2.2 Test Case – Layer 2 Circuit Provisioning

Layer 2 circuit provisioning is a common task of a large enterprise or service provider. This test case focuses on the ability to dynamically-provision point-to-point Layer 2 paths across an OpenFlow network. These endpoints are switch ports, behind which one or more end-hosts are located. Between these endpoints, a primary and secondary (backup) virtual circuit is created.

2.3 Test Case – Layer 3 (IP) Learning with Dynamic Provisioning and Failover

After performing network topology discovery as described in test case 2.1, the Layer 3 (IP) dynamic provisioning application performed learning of the emulated IP hosts by forwarding the ARP to the controller for each host. This Controller was then able to run a Dijkstra-based shortest path algorithm and provision a primary path and an alternate path. Once the path is established it can be verified using test traffic from the emulated hosts. Once in a good, working state, the primary path is failed (several options to cause failure). Upon a failure, traffic should be moved to the alternate path by the controller. The data path is validated with test traffic between the emulated Layer 3 hosts

2.4 Test Case – Load Balancing

There are two ways to perform load balancing on flows.

- The first case is static load balancing.

  Operators can pre-configure routing policy to the Controller (i.e. HTTP traffic is always forwarded via OFSC2 to the destination, but all other traffic to the same destination is not forwarded to OSFC2).

- The second case is Equal Cost Multi-Path (ECMP).

  The Controller dynamically calculates the path-cost of the link and each flow is forwarded to the destination via a path depending on the cost. In this case, hashing with MAC/IP addresses add others are used.

2.5 Test Case – L2 MAC Learning

The Layer 2 MAC learning is one of the more basic test cases that essentially transform the OpenFlow network into an Ethernet switch.

- Once traffic is sent from the emulated hosts, the default switch behavior should be used to send the packet up to the controller.
- The controller use this information to learn where the hosts were located and then instruct the switch to forward the packet out to all ports like the flood behavior of an Ethernet Switch.
- Once the controller learns the location of the source and destination MAC addresses, it could install explicit forwarding entries in each switch.

3 MPLS with SDN

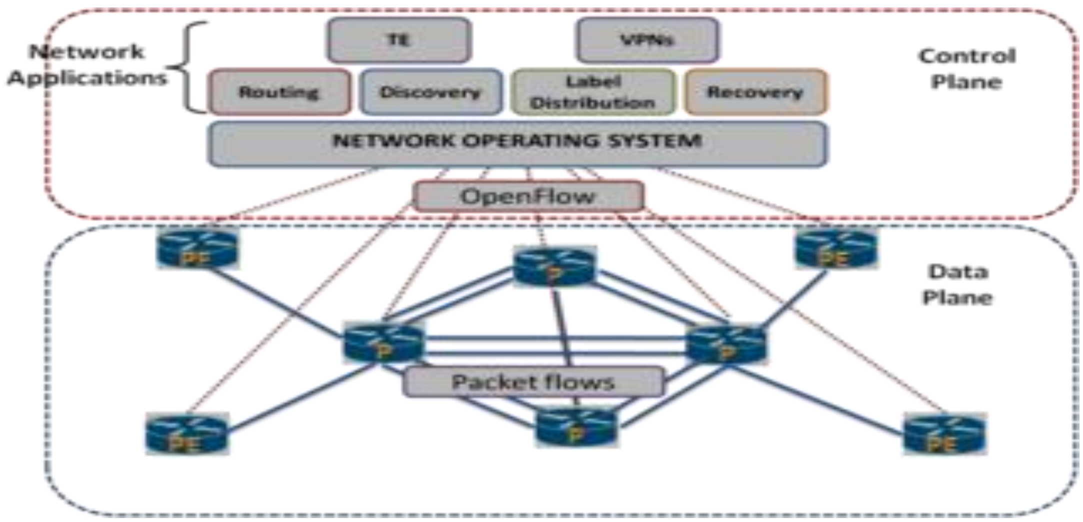

Figure 3: MPLS with SDN reference network

OpenFlow protocol (version 1.1 onwards) gives the ability to push, pop and swap MPLS labels.

3.1 Test case: MPLS Traffic-Engineering based on admission-control of TE-LSPs

3.2 Test case: MPLS L3-VPNs

4 PCE-based SDN

The OpenFlow model is more revolutionary in concept which comes at the cost of requiring a forklift upgrade. With OpenFlow all devices along the path of a flow must support OpenFlow before the path is available, no matter what equipment modification or replacement that may require. A wholesale change of this nature makes it particularly well suited for closed environments such as data centers and campus networks, where IP routing and MPLS signaling are not widely deployed. However, the complete upheaval of control plane function in the WAN, where IP routing and MPLS prevail, is not well suited for incumbent service providers with massive investments in network elements and legacy service requirements.

In contrast, SDN Migration with a PCE-based approach can be gradual and/or partial. With PCE, only the ingress node of a flow needs to be upgraded. Network elements that are not yet upgraded with PCE may still be used in paths and may also continue to function as ingress nodes using their existing path computation function. Moreover, the heavy investments carriers have made in IP routing and MPLS control plane technologies remain intact. For Telco/WAN environments, this approach is significantly less cost, less risk, and less disruptive than OpenFlow making it the preferred approach for SDN.

References

[1]	http://www.openflow.org/wp/documents/
[2]	RFC 5440, "Path Computation Element (PCE) communication Protocol