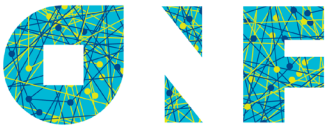

OPEN NETWORKING
FOUNDATION

Migration Use Cases and Methods

Migration Working Group

Disclaimer

Contents

1 Introduction

This document provides a set of high-level guidelines, methods and recommendations to migrate network service from a traditional network, i.e. the starting network, to a Software-Defined Network (SDN) based on the OpenFlow™ Standard, i.e. the target network. Traditional networks in scope include WAN/service provider/carrier networks, datacenter networks, enterprise networks and campus networks with diverse use cases.

This document describes a framework for migration methods exemplified by a set of target networks. This framework includes the description of target network core requirements, starting network migration requirements, phased migration requirements, and finally validation requirements to ensure a complete and successful network migration.

The remainder of this document is organized as follows:

Sections 2, 3 and 4 describe real-world examples of OpenFlow-based SDN migration use cases and deployments covering different network domains. These examples are: Google Inter-Datacenter WAN, NTT Provider Edge, and Stanford Campus Network use cases.

Section 5 provides an overview of the scope and objectives of the ONF Migration Working Group's charter, and defines a terminology and a structure for SDN migration use cases.

Section 6 summarizes best practices and lessons learned from the migration use cases and approaches covered in this document.

Section 7 provides a brief conclusion with suggestions for future work.

This is a living document that will continue to adopt new use cases and document the migration requirements defined within the ONF Migration Working Group's charter. Each use case shall document a list of characteristics described in Appendix A.

Appendix B provides a list of references for further reading.

2 Google Inter-Datacenter WAN Use Case

Google's global user based services (Google Web Search, Google+, Gmail, YouTube, Google Maps, etc.) require significant amount of data to be moved from one region to another, making these applications and services very WAN-intensive. Google concluded that the delivery of such services would not be scalable with the current technologies due to their non-linear complexity in management and configuration. As a result, Google has decided to use SDN for managing WAN as a fabric as opposed to a collection of boxes. A full description of their experience is available in [11]; this use case description excerpts portions to focus on the migration experience.

2.1 Starting Network

2.1.1 General Description

Google's WAN is organized as two backbones – an Internet facing (I-scale) network that carries user traffic and an internal (G-scale) network that carries traffic between datacenters. These two backbones have very different requirements and traffic characteristics. It is the G-scale network in which Google has deployed an OpenFlow powered Software-defined Networking (SDN) solution, which makes the starting network.

Google's datacenter WAN (see Figure 1) exhibits some unique characteristics. For one, Google controls the applications, servers, and the LANs all the way to the edge of the network. Additionally, the most bandwidth-intensive applications perform large-scale data copies from one site to another. These applications benefit most from high levels of average bandwidth and can adapt their transmission rate based on available capacity. They can similarly defer to higher priority interactive applications during periods of failure or resource constraint. Google also anticipated no more than a few dozen datacenter deployments, making central control of bandwidth feasible.

Figure 1 – Google WAN Network

2.1.2 Operational Mode

B4, Google's SDN-powered WAN, provides connectivity among datacenters (see Figure 1). Traffic includes asynchronous data copies, index pushes for interactive serving systems, and end user data replication for availability. Well over 90% of internal application traffic runs across this network.

Google maintains two separate networks because they have different requirements. For example, the user-facing networking connects with a range of gear and providers, and hence must support a wide range of protocols. Further, its physical topology will necessarily be denser than a network connecting a modest number of datacenters. Finally, in delivering content to end users, it must support the highest levels of availability.

B4, in contrast, is Google internal in scope. While there are thousands of individual applications runing across B4 target network, Google categorizes them into three basic classes:

1. user data copies (e.g., email, documents, audio/video files) to remote datacenters for availability/durability
2. remote storage access for computation over inherently distributed data sources
3. large-scale data push synchronizing state across multiple datacenters

These three traffic classes are ordered in increasing volume, decreasing latency sensitivity, and decreasing overall priority. For example, the user-data represents the lowest volume on B4, is the most latency sensitive, and is of the highest priority.

2.1.3 Deployed Equipment

In the B4 Starting Network (SN) WAN routers consist of high-end, specialized equipment that place a premium on high availability.

B4 was built with a three-layer architecture: switch hardware, a site controller layer comprised of network control systems hosting OpenFlow controllers and network control applications, and a global control layer. The switch hardware was custom built from multiple merchant networking chips in a two-stage Clos topology with a copper backplane. Because they are able to use endpoint management to manage the load on this network, these switches avoid deep buffers without expensive packet drops. Similarly, they avoid large forwarding tables because the topology of the inter-datacenter WAN did not require them.

2.1.4 Management Tools

Google developed new monitoring systems for the deployment of B4 [12].

2.1.5 Network Capacity

Network capacity for the B4 has not been made public information by Google.

2.1.6 Pre-Migration Assessment

A number of B4’s characteristics led to the design approach:

- Elastic bandwidth demands: the majority of Google's datacenter traffic involves synchronizing large data sets across sites. These applications benefit from as much

bandwidth as they can get but can tolerate periodic failures with temporary bandwidth reductions.

- Moderate number of sites: While B4 must scale among multiple dimensions, targeting the datacenter deployments meant that the total number of WAN sites would be a few dozens.
- End application control: Google controls both the applications and the site networks connected to B4. Hence, it can enforce relative application priorities and control bursts at the network edge, rather than through over provisioning or complex functionality in B4.
- Cost sensitivity: B4's capacity targets and growth rate led to unsustainable cost projections. The traditional approach of provisioning WAN links at 30-40% (or 2-3x the cost of a fully utilized WAN) to protect against failures and packet loss, combined with prevailing per-port router cost, would make the network prohibitively expensive.

2.1.7 Services Assessment

Google's WAN is among the largest in the Internet, delivering a range of search, video, cloud computing, and enterprise services and applications to users across the globe. These services run across a combination of datacenters spread across the world, and edge deployments for cacheable content.

2.2 Target Software-Defined Network

2.2.1 Objectives

Google adopted Software-Defined Networking (SDN) architecture for its datacenter WAN interconnect to deploy routing and traffic engineering protocols customized to its unique requirements. The SDN design focuses on:

1. Accepting failures as inevitable and common events, whose effects should be exposed to end applications;
2. Switching hardware that exports a simple interface to program forwarding table entries under central control. Network protocols could then run on servers housing a variety of standard and custom protocols.

The expectation was that deploying novel routing, scheduling, monitoring, and management functionality and protocols would be both simpler and result in a more efficient network. Google's WAN (B4), using SDN principles and OpenFlow protocol, manages individual switches. Another objective for the target network was to simultaneously support standard routing protocols and centralized Traffic Engineering (TE) as one of the SDN's earlier applications. With TE, Google can:

1. Leverage control at the network edge to adjudicate among competing demands during resource constraint;
2. Use multipath forwarding/tunneling to leverage available network capacity according to application priority;
3. Dynamically reallocate bandwidth in the face of link/switch failures or shifting application demands.

These features would allow many B4 links to run at near 100% utilization and all links to average 70% utilization over long time periods, corresponding to 2-3x efficiency improvements relative to standard practice.

One of the objectives of the SDN migration is to have a dedicated, software-based control plane running on commodity servers, and the opportunity to reason about global state, yielding vastly simplified coordination and orchestration for both planned and unplanned network changes. SDN will also leverage the raw speed of commodity servers and along with OpenFlow will decouple software and hardware evolution. The control plane software becomes simpler and evolves more quickly; data plane hardware evolves based on programmability and performance. Separating hardware from software helps facilitate customization of routing and monitoring protocols to meet the target network requirements.

Low cost routers built from merchant silicon was another objective, as the network should scale up with growth. Links are utilized at 100% to increase efficient use of long haul transport resources. Centralized traffic engineering is another objective for the network. It uses multipath forwarding to balance application demands across available capacity in response to failures and changing application demands

2.2.2 SDN Architecture

The SDN architecture can be logically organized in three layers (see Figure 2). The Network has multiple WAN sites, each with a number of server clusters. Within each site, the switch hardware layer primarily forwards traffic and does not run complex control software, and the site controller layer consists of Network Control Servers (NCS) hosting both OpenFlow controllers (OFC) and Network Control Applications (NCAs). These servers enable distributed routing and central traffic engineering as a routing overlay. OFCs maintain network state based on NCA directives and switch events and instruct switches to set forwarding table entries based on this changing network state. For fault tolerance of individual servers and control processes, a per-site instance of Paxos elects one of multiple available software replicas (placed on different physical servers) as the primary instance.

The global layer consists of logically centralized applications (e.g. an SDN Gateway and a central TE server) that enable the central control of the entire network via the site-level NCAs. The SDN Gateway abstracts details of OpenFlow and switch hardware from the central TE server. Global layer applications across multiple WAN sites are replicated with separate leader election to set the primary. Each server cluster in the network is a logical “Autonomous System” (AS) with a set of IP prefixes. Each cluster contains a set of BGP routers that peer with the switches at each WAN site. Even before introducing SDN, the WAN as a single AS providing transit among clusters running traditional BGP/ISIS network protocols was in production. Google chose BGP because of its isolation properties between domains and operator familiarity with the protocol. The SDN-based network then had to support existing distributed routing protocols, both for interoperability with the non-SDN WAN implementation, and to enable a gradual rollout.

Google considered a number of options for integrating existing routing protocols with centralized traffic engineering. In an aggressive approach, they would have built one integrated, centralized service combining routing (e.g., ISIS functionality) and traffic engineering. They instead chose to

deploy routing and traffic engineering as independent services, with the standard routing service deployed initially and central TE subsequently deployed as an overlay. This separation delivers a number of benefits. It allowed Google to focus initial work on building SDN infrastructure, e.g., the OFC and Agent, routing, etc. Moreover, since Google initially deployed their network with no new externally visible functionality such as TE, it gave time to develop and debug the SDN architecture before trying to implement new features such as TE.

Perhaps most importantly, Google layered traffic engineering on top of baseline routing protocols using prioritized switch forwarding table entries. This isolation gave their network a "big red button", faced with any critical issues in traffic engineering, they could disable the service and fall back to shortest path forwarding. This fault recovery mechanism has proven invaluable.

Each WAN site consists of multiple switches with potentially hundreds of individual ports linking to remote sites. To scale, the TE abstracts each site into a single node with a single edge of given capacity to each remote site. To achieve this topology abstraction, all traffic crossing a site-to-site edge must be evenly distributed across its entire constituent links. The routers employ a custom variant of Equal Cost Multipath (ECMP) hashing to achieve the necessary load balancing.

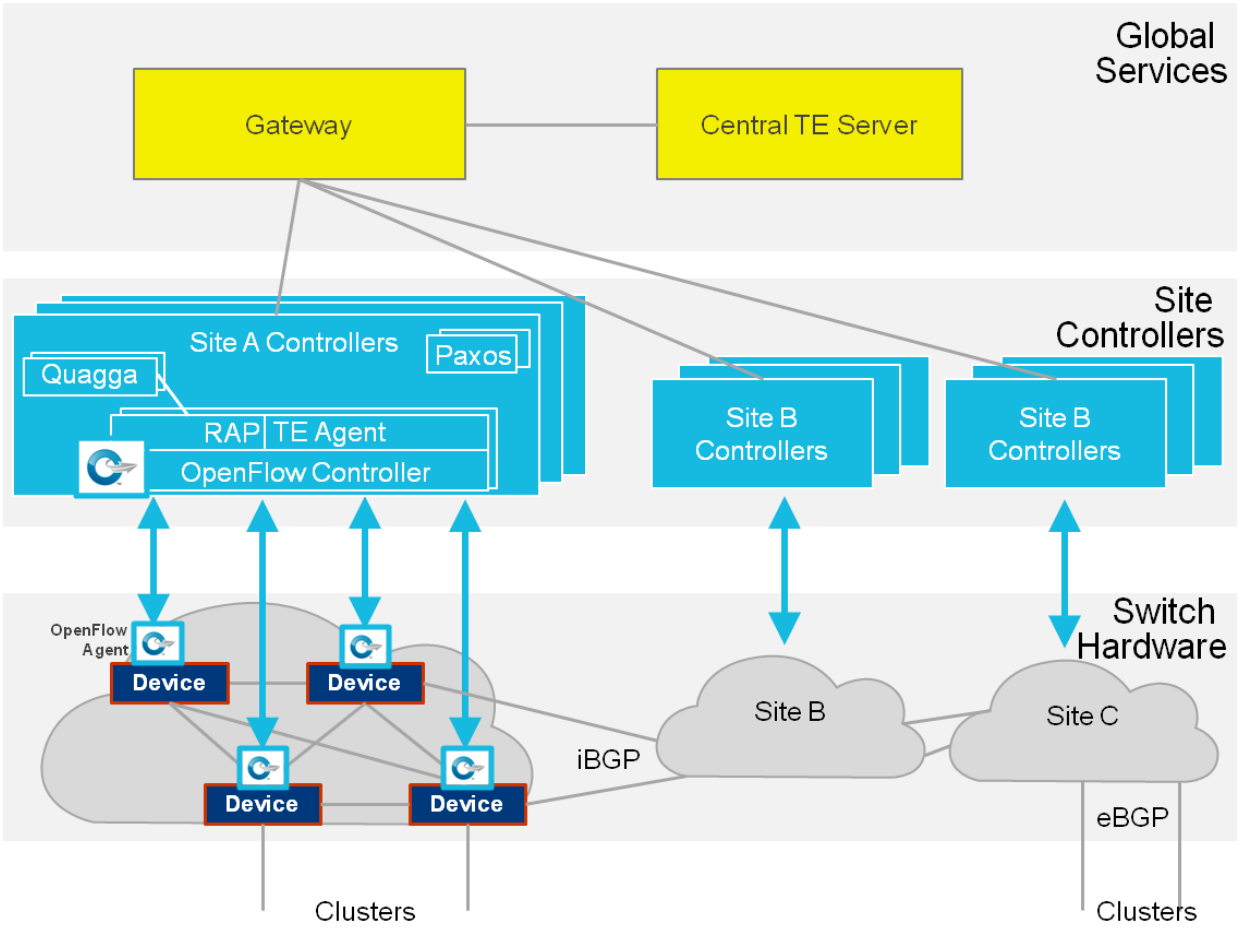

Figure 2 – SDN Architecture

2.2.3 Migration Approach

The migration procedure includes the integration of the target network with the legacy routing, which was to provide a gradual path for enabling OpenFlow in the production network. Google viewed BGP integration as a step toward deploying new protocols customized to the requirements of, for instance, a private WAN setting.

Google's SDN migration path moved in stages from a fully distributed monolithic control and data plane hardware architecture to a physically decentralized (though logically centralized) control plane architecture. The hybrid migration for the Google B4 network proceeded in three general stages.

1. **Starting Network**: In the initial stage, the network connected Datacenters through legacy nodes using E/IBGP and ISIS routing. Cluster Border routers interfaced the Datacenters to the network (See Figure 3).

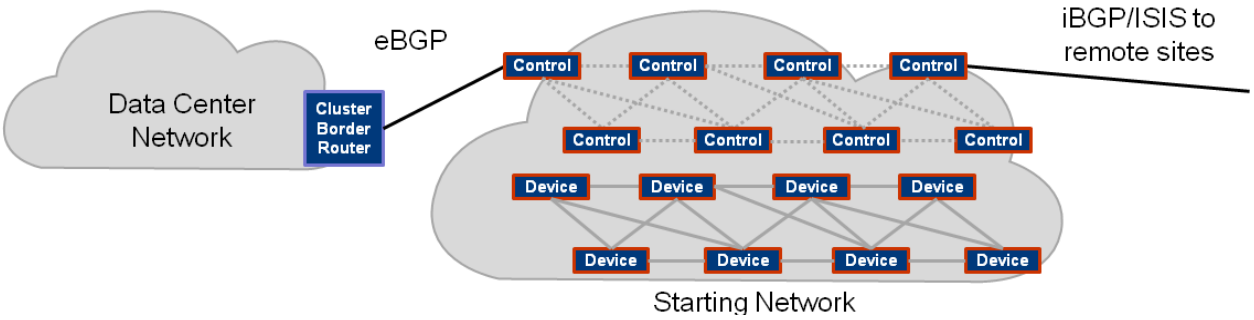

Figure 3 – B4 Starting Network

2. **Phased Deployment** using a mixed network: In this phase, a subset of the nodes in the network were OpenFlow-enabled and controlled by the logically centralized controller utilizing Paxos, OpenFlow controller, and Quagga (See Figure 4).

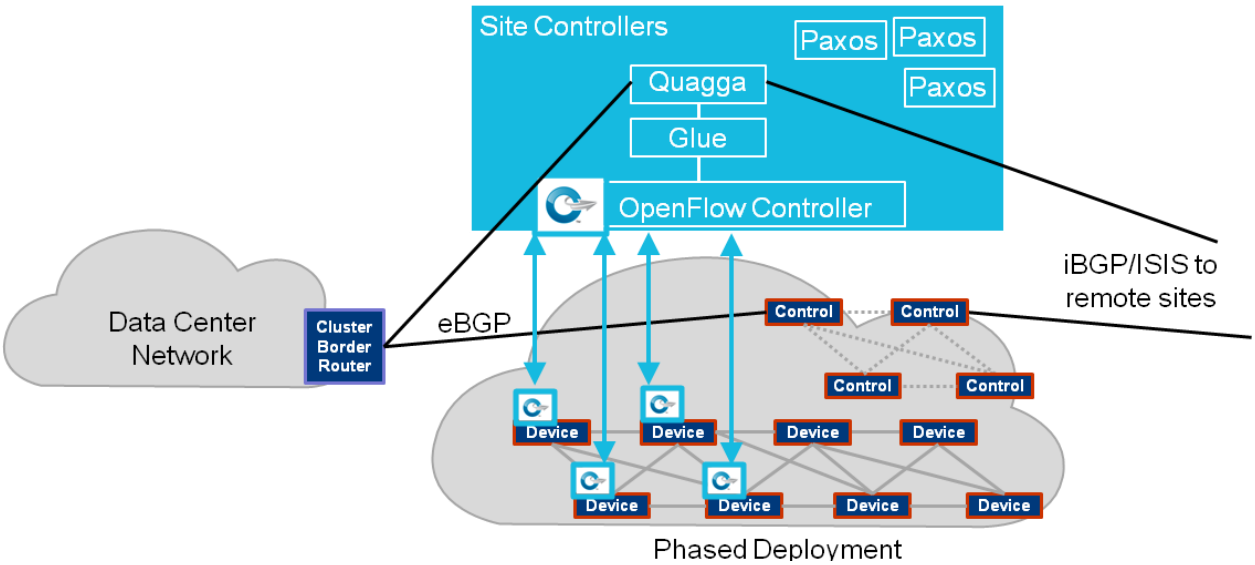

Figure 4 – B4 Phased Deployment using a Mixed Network

3. **Targe Network**: All nodes were OpenFlow enabled in the final phase. In the target network, the controller controls the entire network. There is no direct correspondence between the Datacenter and the network. The controller has also TE server that guides the Traffic Engineering in the network (See Figure 5).

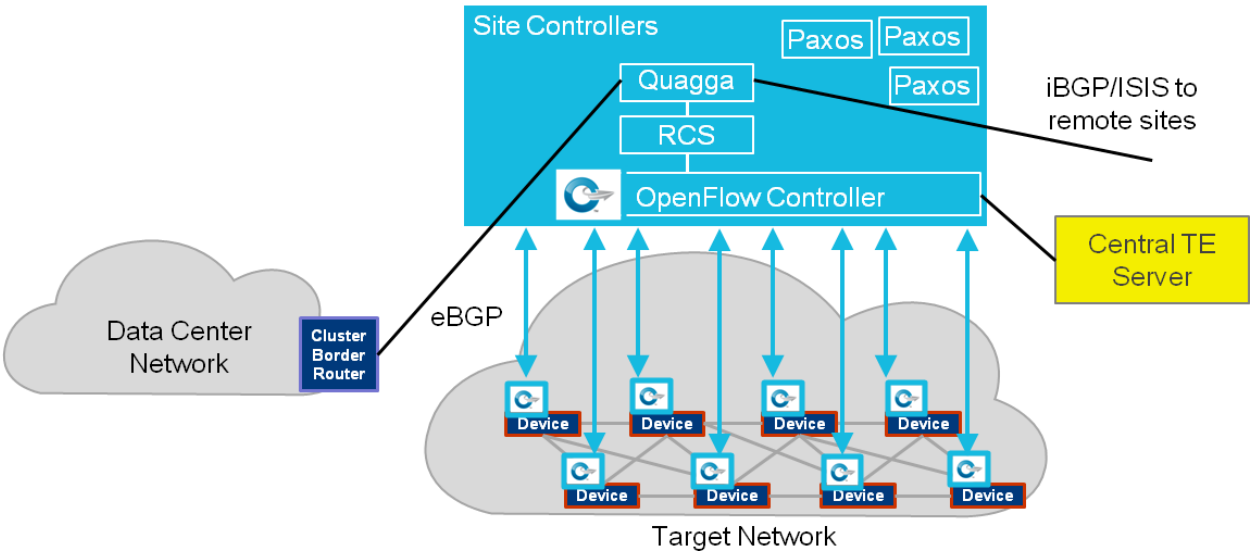

Figure 5 – B4 Target Network

2.2.4 Dependencies

The SDN-based Network had to support existing distributed routing protocols, both for interoperability with the non-SDN WAN implementation, and to enable a gradual rollout.

2.2.5 Target Dependencies

- **Dependencies among Ops**: To avoid packet drops, not all ops can be issued simultaneously. For example, Google must configure a Tunnel at all affected sites before configuring the corresponding Tunnel Group (TG) and Flow Group (FG) [12]. Similarly, a Tunnel cannot be deleted before removing all referencing entries. Figure 6 shows two example dependencies ("schedules"). Figure 6 (a) shows the creation of TG_1 with two associated Tunnels T_1 and T_2 for the A →B FG_1. Figure 6 (b) shows the removal of T_2 from TG_1.

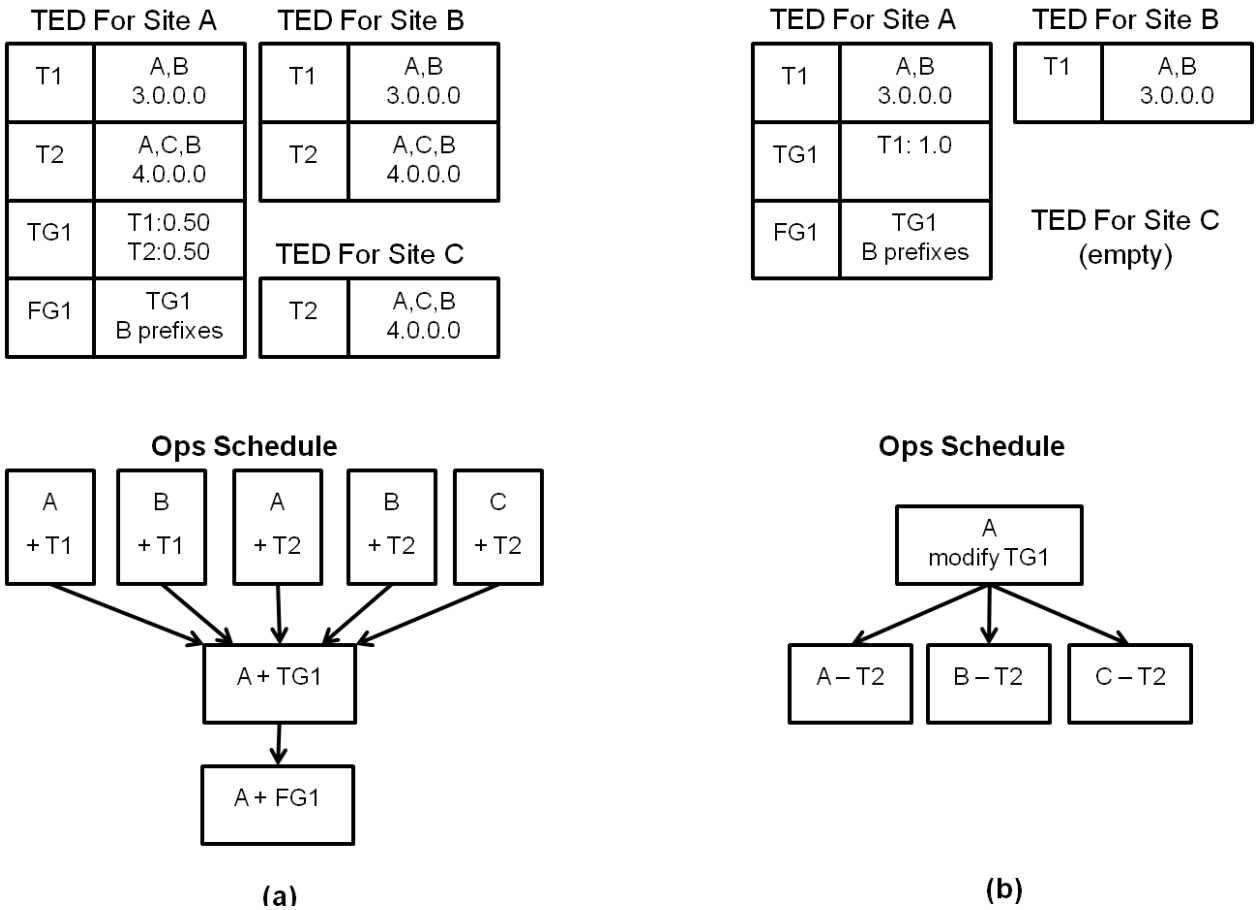

Figure 6 – Dependencies among Ops

- **Synchronizing the Traffic Engineering Database (TED) between TE and Open Flow Controller (OFC)**: Computing diffs requires a common TED view between the TE master and the OFC. A TE session between the master TE server and the master OFC supports this synchronization. Google generates a unique identifier for the TE session based on mastership and process IDs for both end points. At the start of the session, both endpoints synchronize their TED view. This functionality also allows one source to recover the TED from the other in case of restarts. TE also periodically synchronizes TED state to a persistent store to handle simultaneous failures. The Session ID allows for rejecting any op not part of the current session, e.g., during a TE mastership flap.
- **Ordering issues**: Consider the scenario where TE issues a TG op (TG_1) to use two tunnels with T_1:T_2 split 0.5:0.5. A few milliseconds later, it creates TG_2 with a 1:0 split as a result of failure in T_2. Network delays/reordering means that the TG_1 op can arrive at the OFC after the TG_2 op. Site-specific sequence IDs were attached to TEops to enforce ordering among operations. The OFC maintains the highest session sequence ID and rejects ops with smaller sequence IDs. TE server retries any rejected ops after a timeout.
- **TE op failures**: A TE op can fail because of RPC failures, OFC rejection, or failure to program a hardware device. Hence, Google tracks a [Dirty|Clean] bit for each TED entry.

Upon issuing a TE op, TE marks the corresponding TED entry dirty. Dirty entries are cleaned upon receiving acknowledgment from the OFC. Otherwise, TE retries the operation after a timeout. The dirty bit persists across restarts and is part of TED. When computing diffs, any dirty TED entry is automatically replayed. This is safe because TE ops are idempotent by design. There are some additional challenges when a TE session cannot be established, e.g., because of control plane or software failure. In such situations, TE may not have an accurate view of the TED for that site. In the current design, Google continues to assume the last known state for that site and force fail new ops to this site. Force fail ensures the any additional dependent ops are not issued.

2.2.6 Tool Requirements

This information has not yet been disclosed publicly by Google.

2.2.7 GAP Analysis

- **OpenFlow protocol**: The OpenFlow protocol is in its infancy and is bare bones. However, as deployment shows, it is good enough for many network applications.
- **Fault tolerant OpenFlow controllers**: To provide fault tolerance, multiple OpenFlow controllers must be provisioned. This requires handling master election and partitions between the controllers.
- **Partitioning functionality**: It is not very clear what functionality should reside in the network devices and what should reside in external controllers. Configuration of functionality resident in the network device remains an open question.
- **Flow programming**: For large networks, programming of individual flows can take a long time.

2.2.8 Migration Procedures

The migration procedure includes the integration of the target network with the legacy routing, which was to provide a gradual path for enabling OpenFlow in the production network. Google viewed BGP integration as a step toward deploying new protocols customized to the requirements of, for instance, a private WAN setting.

2.2.9 Post-Migration Acceptance

The target network has been in deployment for three years. It carries more traffic than Google's public facing WAN, and has a higher growth rate. It is among the first and the largest SDN/OpenFlow deployments. The target network scales to meet application bandwidth demands more efficiently than would otherwise be possible, supports rapid deployment and iteration of novel control functionality such as TE, and enables tight integration with end applications for adaptive behavior in response to failures or changing communication patterns. SDN is of course not a panacea. Google reported a large-scale outage on this target network by pointing out the challenges in both SDN and large-scale network management [12].

During post-migration evaluation, the main points are:

1. Topology aggregation significantly reduces path churn and system load;
2. Even with topology aggregation, edge removals happen multiple times a day;
3. WAN links are susceptible to frequent port flaps and benefit from dynamic centralized management.

2.2.10 Services Acceptance

Google initiates TG ops after every algorithm iteration and runs the TE algorithm instantaneously for each topology change and periodically to account for demand changes. The growth in TG operations comes from adding new network sites. There was a drop in failures in May (Month 5) and Nov (Month 11) following optimizations resulting from their outage experience (see Figure 12 of [12]).

There were experiments conducted to evaluate the impact of failure events on network traffic. They observed traffic between two sites and measured the duration of any packet loss after six types of events: a single link failure, an encap switch failure and separately the failure of its neighboring transit router, an OFC failover, a TE server failover, and disabling/enabling TE.

Table 1 summarizes the results. A single link failure leads to traffic loss for only a few milliseconds, since the affected switches quickly prune their ECMP groups that include the impaired link. An encap switch failure results in multiple such ECMP pruning operations at the neighboring switches for convergence, thus taking a few milliseconds longer. In contrast, the failure of a transit switch that is a neighbor to an encap switch requires a much longer convergence time (3.3 seconds). This is primarily because the neighboring encap switch has to update its multipath table entries for potentially several tunnels that were traversing the failed switch, and each such operation is typically slow (currently 100ms).

By design, OFC and TE server failure/restart are all hitless. That is, absent concurrent additional failures during failover, failures of these software components do not cause any loss of data-plane traffic. Upon disabling TE, traffic falls back to the lower-priority forwarding rules established by the baseline routing protocol.

Failure Type	Packet Loss (ms)
Single Link	4
Encap Switch	10
Transit Switch Neighboring an Encap Switch	3300
OFC	0
TE Server	0
TE Disable/Enable	0

Table 1 – Traffic Loss Time per Failure type

3 NTT Provider Edge Use Case

This use case documents NTT's migration of BGP to OpenFlow.

In the traditional BGP deployment models, the Provider Edge (PE) router maintains numerous BGP adjacencies as well as large number of BGP routes/paths for multiple address families such as IPv4, IPv6, VPNv4 and VPNv6 etc. In addition, to meet customer service level agreements, the PE may be configured with aggressive BGP session or Bidirectional Forwarding Detection (BFD) timers. Handling BGP state machine, processing BGP updates as per configured policies and calculating best paths for each address-family puts a heavy load on the router. Additionaly, by definition, service changes are quite frequent on the PEs to provision new customers or update customer policies. Because of the limited resources, including CPU and memory, as well as the proprietary nature of the operating system (OS), service acceleration and innovation is dependant on vendor implementation.

BGP free edge defines a new paradigm of simplifying the eBGP routing (control plane load) on the PE routers. In this deployment model, a PE router is converted into a forwarding/transport node to handle data plane traffic whereas BGP control plane function is offloaded to a separate external entity. The external control plane entity leverages OpenFlow/SDN to program the forwarding entries for the data plane traffic on the PE router. Here are some of the motivations behind BGP free edge:

- Simplified and low-cost routing edge architecture with centralized BGP policy management.
- Accelerated deployment of new edge services enabled by control plane and data plane decoupling (OpenFlow) for BGP (carrier service).
- Better control of traffic patterns in the core.
- Flexibility to calculate customized BGP best paths for not only each ingress point, but also on a per customer basis.
- Reduction in BGP Wave Effect – Helps Internet stability, by elimination of leaking intra-AS path transitions.
- Easy BGP monitoring and reporting tools interface. There is no need to collect all BGP "raw" feeds (aka original intention of BGP Monitoring Protocol (BMP)) from all border routers. An API from BGP Route Controller (RC) will enable viewing of the entire BGP in an Autonomous System.

3.1 Starting Network

Figure 7 illustrates a typical IP/MPLS network with PE routers with different types of BGP peerings, including internet providers as well as VPN customers.

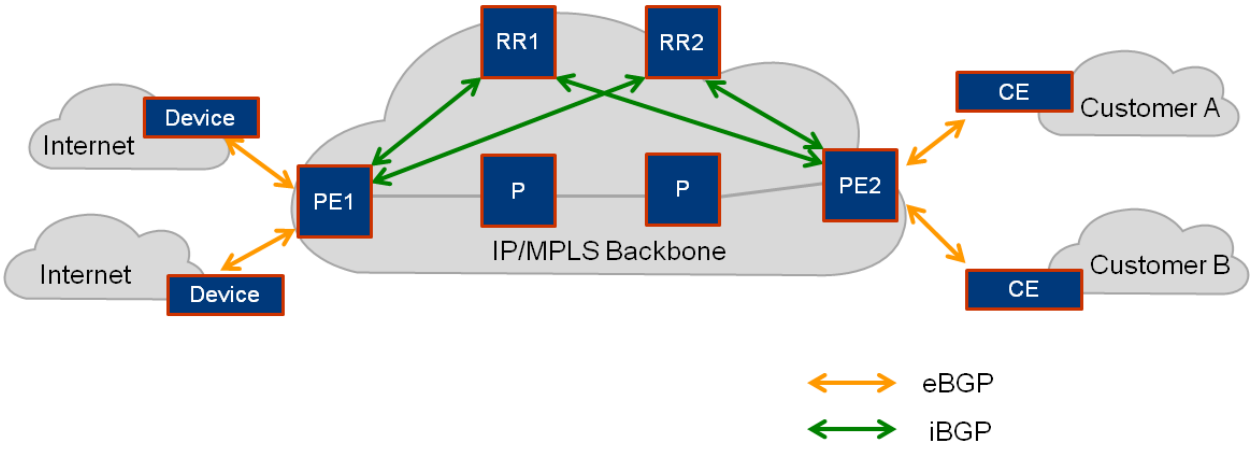

Figure 7 – Current BGP Deployment Model

3.1.1 General Description

In the starting network, the PE router runs BGP with external BGP speaking peers. In a typical service provider environment, it is not uncommon for a PE router to maintain 500K+ Internet and/or L3VPN routes. Besides external peerings, the PE router also maintains internal peering sessions, typically with dual Route Reflectors (RR) as depicted in Figure 7. Typically, all BGP sessions as well as policies are configured manually using vendor specific CLI.

3.1.2 Operational Mode

It is the responsibility of the PE router to negotiate BGP session parameters with both internal and external BGP speakers and maintain BGP and TCP state machine for each neighbor session. The PE router typically peers with external neighbors using the connected subnet address of the neighbor while for iBGP; neighbor relationship is established using loopback addresses. For next-hop reachability, the PE router also runs an Interior Gateway Protocol (IGP). Each PE router may receive more than one copy of the same prefix from different external peers or route-reflectors. As a result, it calculates the best path for installation in the routing/forwarding table while maintaining multiple copies of prefixes in the BGP database.

3.1.3 Deployed Equipment

The edge node can be any OpenFlow capable switch. NTT used an OpenFlow switch on the edge based on the Open vSwitch (OVS) and home-grown OpenFlow controller running over Ryu.

3.1.4 Redundancy Model

Redundancy model for BGP deployment varies depending on the customer connectivity to the PE router as well as the type of peering, i.e. eBGP vs iBGP. For example, in case of single-homed customers, the resiliency against PE router failure is implemented via BGP Graceful Restart (GR) or Non-Stop Routing (NSR) capabilities while for dual/multi-homed customers, the resiliency against primary path/PE router failure is achieved using the alternate path to a second PE. For GR/ NSR capability, redundant route/management processor modules are assumed to be present in the system. Note that, from the control plane point of view, only one TCP/BGP state is maintained per session. The BGP routing information or TCP state (only in case of

NSR) is copied on the standby management module. For iBGP redundancy, a PE router typically peers with a pair of route-reflectors to ensure that if one RR fails, the other still can exchange routing information between PE routers.

3.1.5 Management Tools

Management tools include SNMP, BMP, and home grown BGP monitoring tools.

3.1.6 Network Capacity

Typical PE routers can handle 1000s of eBGP peers and hundreds of thousands of routes/paths. In addition, PE routers have a mix of high and low speed physical and logical interfaces for customer and core connectivity to handle data plane traffic. Migration to BGP free edge relieves the BGP control plane load from the PE router. However, the data plane capacity requirements expected from the edge devices do not change when migrating to the new model.

3.1.7 Upgrade Requirements

The edge device must be OpenFlow capable, i.e. include an OpenFlow Agent. OpenFlow and BGP route controllers must be available in the network. The BGP route controller must be able to communicate with the OpenFlow controller as well as with the BGP router reflector if one exists in the network.

3.1.8 Availability

The edge device and controllers must implement high availability features including within the OS and redundancy at the hardware and protocol level.

3.1.9 Problems and Challenges

Following are some of the challenges with the current deployment model:

- The data plane and BGP control plane are tightly coupled.
- It is hard to keep up with BGP control plane changes or additional features on vendor specific OS and platforms.
- Scaling puts an extra load on the edge router's control plane, which can lead to failures.
- BGP scaling is limited by the CPU/Memory resources available on the PE router.
- BGP configuration, management, monitoring and troubleshooting ardifficult and complex especially for large-scale deployments.
- The network operator spends a significant amount of time creating/maintaining BGP peering sessions and policies manually.

3.1.10 Pre-Migration Assessment

The solution requires moving BGP control plane onto a commodity X86 server and using an OpenFlow agent to configure the forwarding plane. Pre-migration steps include:

- Ensuring that the BGP Free Edge solution and the platform being used for the BGP control plane offload will be able to support required scale and future growth.
- Verifying the OpenFlow Controller and OpenFlow Agent run compatible version of protocols.
- Verifying the BGP route controller will be capable of handling BGP process.

- Ensuring that appropriate APIs, scripts, and other operational tools are compatible with the SDN based deployment.
- Optionally, ensuring the BGP peer creation and activation can be automated.
- Ensuring that proper training is provided to Network Operations Center (NOC) staff.

3.2 Target Software-Defined Network

3.2.1 Objectives

The objective of BGP-free edge is to simplify routing by transforming an edge device into a transport node configured by an OpenFlow Agent by separating control and data planes while storing and processing the BGP routing information on a single, or cluster of, compute device(s). Such a transformation of edge devices must ensure that SLAs as well as BGP operation remains transparent to customer.

3.2.2 SDN Architecture

In the BGP-free edge model, the remote BGP peers (e.g. Custer Edge devices) are connected to the PE devices as before. However, the BGP sessions are not handled or terminated by the PE. One of the pre-requisite for the PE is that it must be OpenFlow capable. An OpenFlow controller pre-programs default flows on the PE that would send all BGP control plane traffic from all the internal and external peers to the BGP route controller running BGP as depicted in the Figure 8.

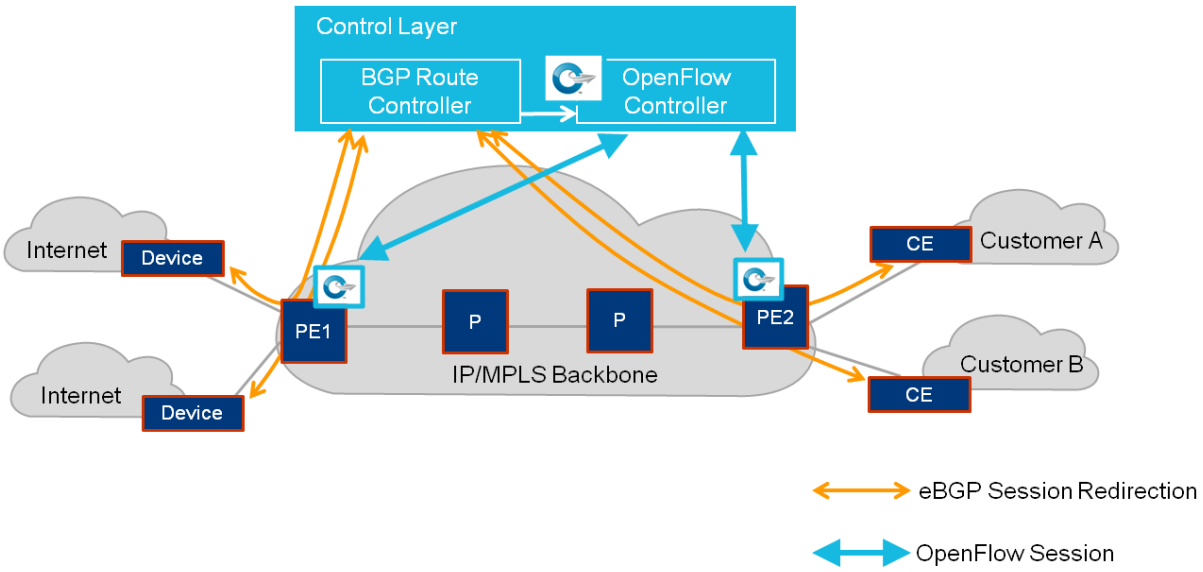

Figure 8: BGP Free Edge- SDN Architecture

The role of BGP route controller is similar to a route reflector for eBGP neighbors. Incoming BGP sessions are recognized (using proactive match rule) by the OpenFlow agent and forwarded to BGP route controllers in the network. The BGP route controller manages BGP sessions with the peering routers, however no end customer or internet traffic flows to/from the peers into the network until the flow table is programmed by the OpenFlow Agent at the edge device.

When setting up a BGP session, CE routers don't see the difference between the BGP route controller and the edge device with respect to configuration. In other words, remote peers continue to use the same IP addresses on the edge node e.g. A', B' and C' as depicted in Figure 9 to establish the BGP peering relationship.

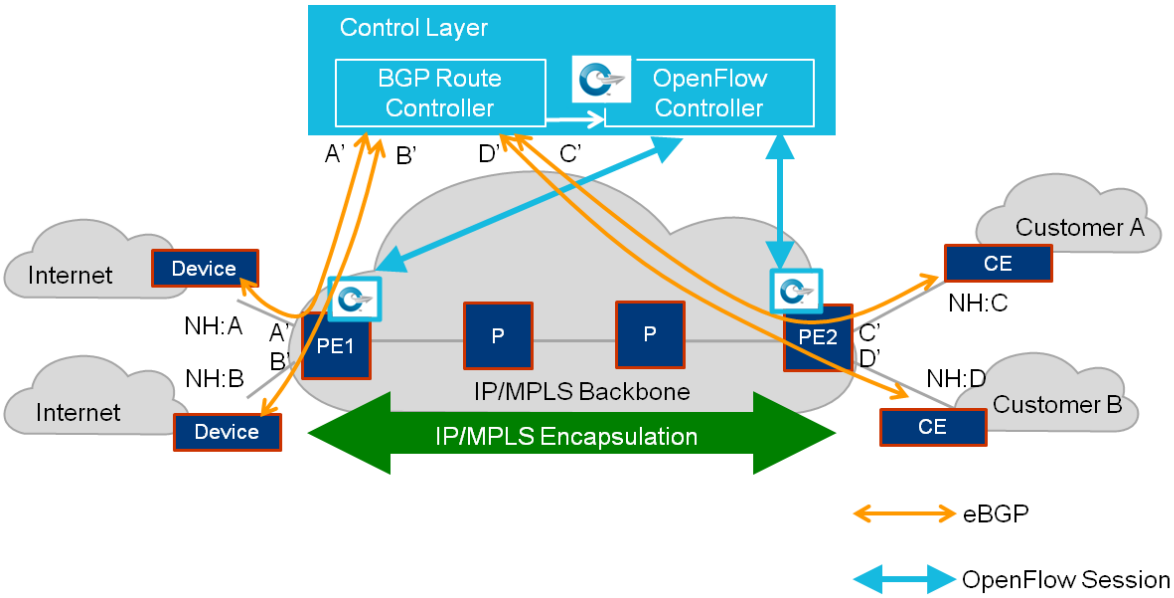

Figure 9: BGP Peering between customers and BGP route controller

The OpenFlow Agent redirects the BGP session to the BGP route controller without changing anything in the packet, using the default OpenFlow match entry. From a physical connectivity point of view, the BGP route controller may or may not be directly connected to the OpenFlow Agent and it may service multiple OpenFlow Agents. For each BGP session, the BGP route controller must be configured with a VLAN and the corresponding IP address, i.e. A', B' and C' etc, which is configured on the device.

To populate the forwarding plane, the OpenFlow Controller needs a copy of the BGP routing information. It gets the routing information from the BGP route controller and builds a flow table, which is then pushed to the OpenFlow Agent. Once flows are programmed successfully, traffic can be forwarded through the edge device without running BGP on it. Any changes i.e updates/withdraws processed by the route controller are reflected in the OpenFlow controller, which, in turn, updates the flow table through the OpenFlow Agent accordingly.

From a deployment perspective, one BGP route controller may be be sufficient per PE. However, it can become a single point of failure. Alternatively, redundant BGP route controllers can be deployed per single or multiple PEs , assuming there is no additional requirement on the remote peers with respect to configuration. In other words, remote peers must continue to peer with BGP route controller as if they have a single BGP session with the PE router. To address the redundancy aspect, NTT use case introduces a concept of lightweight BGP/TCP shim as illustrated in Figure 10.

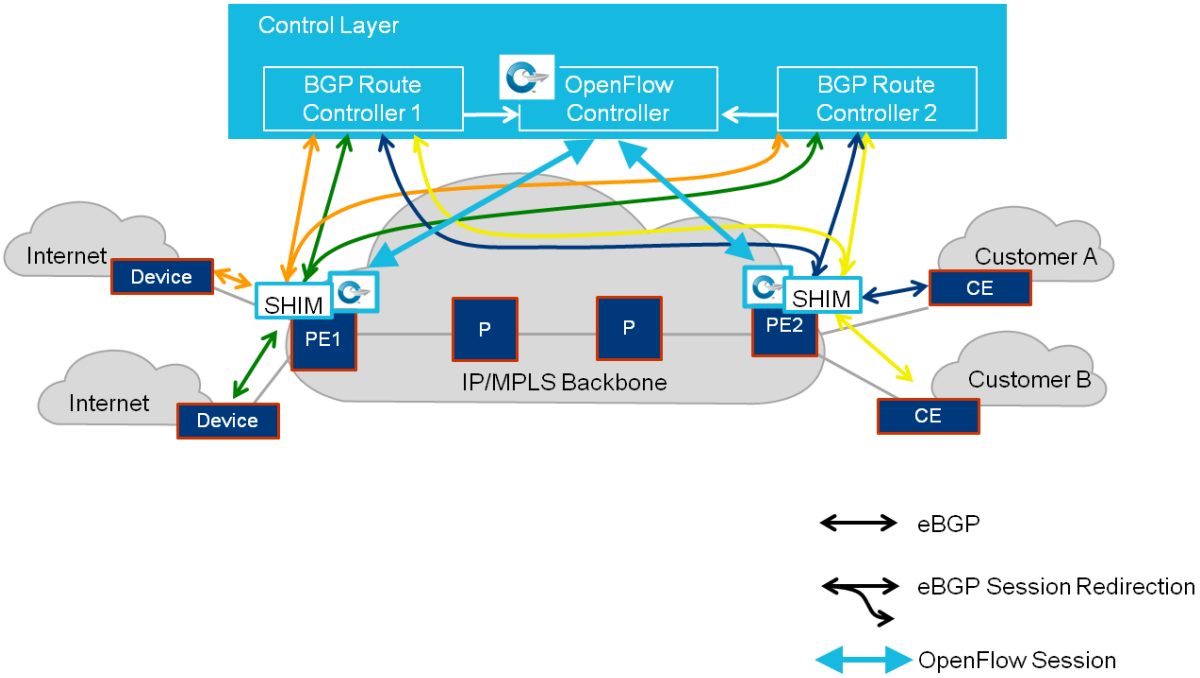

Figure 10: BGP Route Controller with SHIM at the Edge Node

In this case, the incoming eBGP TCP sessions from remote peers are recognized (using pre-programmed flow) by OpenFlow Agent as previously shown with the difference that TCP sessions are now terminated by local shim on the edge device and relayed to redundant BGP Route Controllers in the network. From a logical operation point of view, this is quite similar to the concept of router reflector except that no BGP state is maintained on the edge device.

Figure 11 illustrates the how the OpenFlow Agent handles BGP TCP session via the shim. For every customer BGP session, two sessions are created for redundancy and relayed to each of the BGP route controllers. Optionally, the shim can handle additional processes, such as BFD, BGP route-refresh and BMP, for local processing.

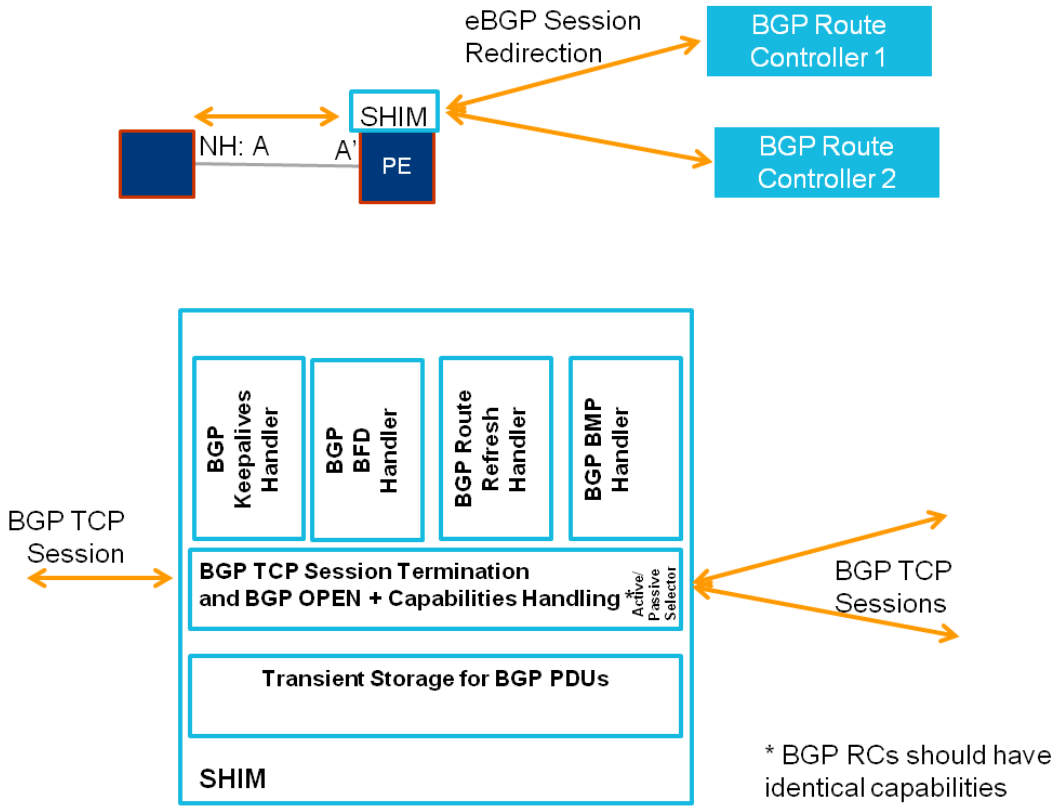

Figure 11: BGP/TCP SHIM for BGP route controller redundancy

The following options describe the BGP state synchronization upon Active/Passive switchover of BGP route controller.

Option 1: Use of BGP Enhanced Route Refresh Mechanism

Option 2: Use of GR:

- Active shim client goes down
- Shim triggers GR procedure on eBGP peers by artificially performing session bounce
- Shim elects new active shim client and switches over to it
- Shim forces complete route refresh to be received from the newly elected active shim client
- Upon reception of End-of-RIB (EOR), all eBGP speakers purge all paths which differ from those received from previously active shim client

Option 3: No GR

- This case is the same as today when no GR/NSR is configured.
- Shim can run NSR when rundundant CPUs/RPs are available

Ideally, it is desirable that peering routers as well as BGP Route Controller support BGP enhanced route refresh or GR for data plane non-disruptive switchover between active and passive BGP Route Controllers.

3.2.3 Migration Approach

There are a few approaches to migrating from the traditional BGP speaking edge router to the BGP-free paradigm depending on the type of device being deployed.

- Greenfield Deployment- This is perhaps the easiest migration model and there is no migration per se since it is a Greenfield deployment. It assumes that all the edge devices are OpenFlow capable and will be deployed BGP-free with BGP terminated at the route controller.
- Mixed or Ships in the Night Deployment – This migration approach assumes that a new BGP-free Edge router is deployed and it will co-exist with other traditional BGP speaking routers. The new BGP-free edge devices and the traditional devices will need to exchange routing information between each other via the BGP route controller. H
- Hybrid Network Deployment – In this case, both legacy and OpenFlow devices can coexist. In addition, the edge device is assumed to be capable of running BGP as well as OpenFlow protocol. The edge device continues to run BGP while BGP sessions and corresponding policies are offloaded to the BGP route controller gradually. This requires careful planning and a lot more resources during the transition stage especially since edge device has to maintain the regular forwarding and OpenFlow forwarding tables along with BGP table.

3.2.4 Dependencies

One of the main requirements is to make the solution resilient and this has a dependency on the shim support on the edge device.

3.2.5 General Considerations

As mentioned earlier, in the current deployment, BGP handles a large number of routes and typically the convergence requirements upon router reload or BGP process restart creates abrupt and heavy control plane traffic. The rate at which a typical PE router handles the BGP incoming or outgoing updates is paced but the programming of the FIB table after the routes have been learned is quite fast. Dynamics of FIB download rate from OpenFlow Controller to OpenFlow Agent could be very slow which is typically not a challenge in traditional BGP deployments. To be fair, the issue is not necessarily related to the BGP-free Edge solution but has more to do with OpenFlow technology. There are some alternative solutions that can eliminate the unnecessary programming of a large number of forwarding enteries by using FIB aggregation/suppression techniques. Schemes such as simple-va [14] can be leveraged to eliminate redundant state in the RIB and FIBs of network elements with overlapping prefix suppression and other processing done on route controller.

3.2.6 Target Dependencies

The base solution is dependant on the availability of BGP route controller that can handle all the BGP functionality transparently and can exchange routing information with the remote peers. The edge device must be OpenFlow capable. In addition, the BGP route controller must be able to communicate with the OpenFlow Controller. The strictest requirement here is the ability of the edge device to support shim for redundancy purposes.

3.2.7 Tool Requirements

SDN Openflow Management tools can be leveraged. In the NTT proof-of-concept, no specific tools were used besides the home grown Ryu-based controller.

3.2.8 GAP Analysis

Currently there is no standard way to support shim on the edge device for better resiliency and redundancy. In addition, the communication between the BGP route controller and OpenFlow controller is not well defined.

3.2.9 Migration Procedures

In order to migrate the starting network to BGP-free edge, the following are some high level steps that can be followed for mixed or ships in the night migration approaches. Prior to the migration of the actual BGP session, an OpenFlow Agent, a BGP route controller and an OpenFlow Controller must be deployed first. In addition, any existing RR can be leveraged to exchange the routing information between the BGP Network and OpenFlow controllers. Note that, during the migration, each BGP session will temporarily flap on the legacy PE router and it will be re-established with the BGP route controller. Therefore, to minimize the disruption, it is recommended to migrate the BGP sessions one by one or in small batches.

1. Configure an iBGP session between RR and BGP route controller so that BGP route controller can learn routes from the entire network.
2. Configure BGP between the OpenFlow controller and the BGP route controller.
3. Program a default flow entry in the OpenFlow controller to initially forward traffic for the matching OpenFlow entry to BGP route controller. Alternatively, TCP port 179 can be programmed to match and forward only BGP traffic to the BGP route controller.
4. Before the migration, BGP path information for a random sample of prefixes should be captured. This will help in validating accurate BGP path information after migration.
5. Configure a VLAN per customer and configure a corresponding BGP session on the BGP network controller.
6. Once the session is established, decommission the session on the legacy router.
7. BGP route controller runs BGP best path selection algorithm and passes the best paths to OpenFlow Controller which in turn programs the OpenFlow Agent.
8. Once the forwarding table is programmed, control traffic continues to be forwarded to the BGP route controller while data traffic now follows the path through the OpenFlow and non-OpenFlow enabled devices along the way to the destination.
9. Repeat the above steps to migrate the rest of the BGP sessions and additional PE routers.

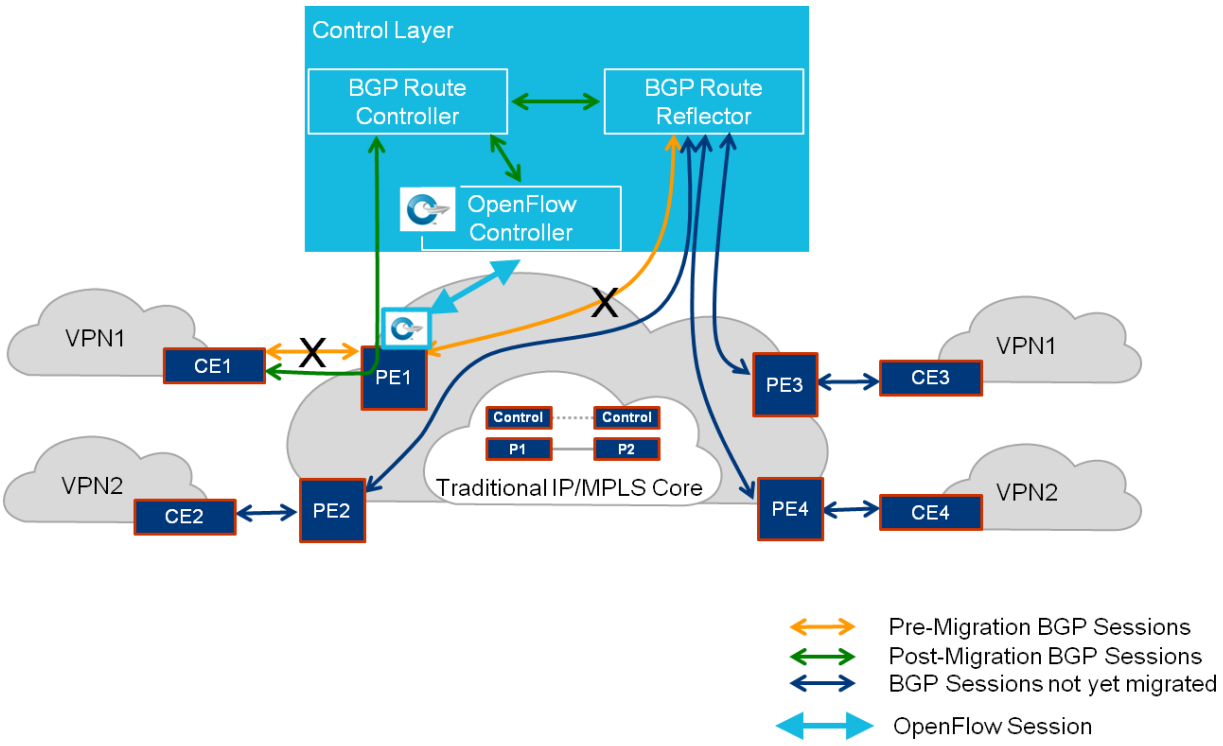

Figure 12 - BGP Free Edge Migration Procedure

3.2.10 Post-Migration Acceptance

All the BGP sessions on the BGP Network Controller should be up. Ensure that the BGP Network Controller receives and sends all the expected BGP routes with proper next hops from the customers as well as from RR and selects the correct BGP best paths. As a benchmark, compare the BGP output of select prefixes with the sample output captured in step 4 during migration procedure. This will ensure that BGP routes are learned accurately.

3.2.11 Services Acceptance

Any existing Internet or VPN services should function normally. A random sample of prefixes in the Internet as well as for select customers can be used to validate the service continuity. Appropriate troubleshooting steps such as ping and trace routes can be employed to check the connectivity.

3.3 Options

Only the mixed environment or ships in the night model of migration approach is discussed here. The greenfield or hybrid mode deployments are also two other options for migration. From the migration procedures point of view, the majority of the steps would remain the same except for a few subtle differences that are not documented at this time.

3.4 References

Additional details can be found in [14].

3.5 Conclusion

BGP-free edge solution provides a simple yet elegant solution to offload BGP from the PE router. In this case, BGP processing such as origin and security signature validation and crypto processing is moved from each PE router to a central control plane. As a result there is no more opex and capex to upgrade CPU and memory network wide for control plane growth. Overall, BGP-free edge provides the flexibility of deploying new services and helps overcome the limitations of scaling and other challenges with current deployments.

4 Stanford Campus Network Use Case

A part of the Stanford campus network was successfully migrated to support OpenFlow in 2010. Initially the migration targeted select type of users (wireless), then expanded to opt-in selected wired users in 3A wing of William Gates building, and then expanded to multiple islands across 2 buildings.

4.1 Starting Network

4.1.1 General Description

There were 2 buildings in which Stanford deployed OpenFlow-enabled infrastructure, for handling both wired and wireless users:

- **William Gates CS building**: The initial network was run by legacy HP Procurve switches deployed in 2 closets in 3A wing of William Gates building and 6 closets of CIS/CIX building. In William Gates building, there were VLAN and associated /24 IP subnet allocated based on research group. The topology of the network was as shown Figure 13 in which the blue circle represents the target subnet to migrate. Originally there was no OpenFlow support in the hardware. The specific group that was switched to OpenFlow used VLAN 74 and had 25 wired users with an aggregate uplink bandwidth of 2Gbps.

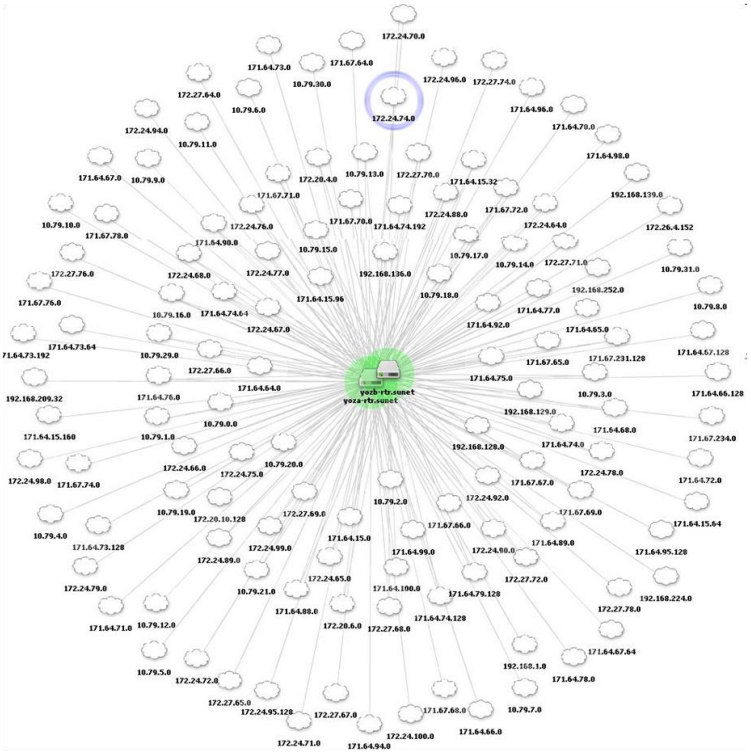

Figure 13 – OpenFlow-ready Gates Building

- **Paul Allen CIS building**: This building also features HP Procurve switches deployed in 6 closets spanning 4 floors (including the basement level). Originally there was no OpenFlow support in the switches. The specific product VLAN that was OpenFlow-enabled is VLAN 98 and it had over 50 workstations in classrooms.

There was (and continues to be) no redundancy at the closet level. All closet switches, however, connected to two distribution switches in the basement. The basement distribution switches in turn connected to a redundant pair of (Cisco-based) core routers of the campus. All switches run the spanning tree protocol to avoid loops. The legacy network was managed by Zenoss open-source software, along with CLI based configurations.

The migration was planned to provide better visibility into the network traffic and allow network experimentation to which users can choose to opt-in for. The goal was, thus, to migrate selected VLANs and users to OpenFlow control. This allowed for a clear path of staged deployment within the existing campus network.

Following are the main requirements to ensure when migrating to the target network:

a) Network availability should be greater than 99.9%;
b) There should be a fail-safe scheme to revert the network back to legacy mode;
c) Network performance should be close to the legacy networks' performance;
d) User-experience should not be affected in any way.

4.1.2 Monitoring Infrastructure

Critical to the deployment is a robust network and traffic monitoring scheme. The monitoring infrastructure collects information in 2 planes:

1) **Control plane**: Most controllers archive flow-level information based on incoming packet_in and flow_exp messages. That information can be queried using Representational State Transfer (REST) or other API. The main statistics we collect are for the *flow_arrival_rate* and *active_flows*.
2) **Data plane**: Stanford relies on some dedicated monitor nodes running *ping* and *wget* between each other to collect information about the *switch_cpu_utilization*, *flow_setup_time*, *RTT*, *wget_delay*, and *loss_rate*. These were collected before and after OpenFlow migration.

Figure 14 summarizes the OpenFlow monitoring infrastructure used.

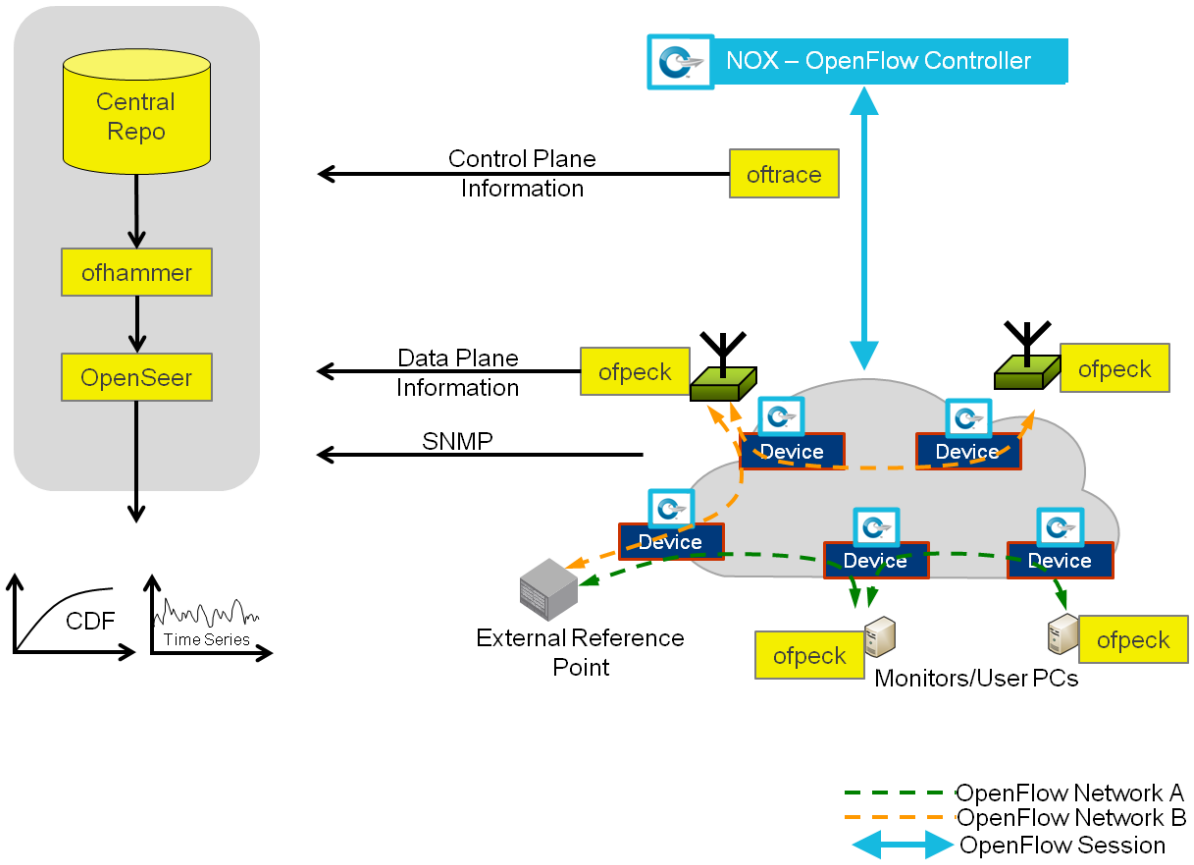

Figure 14 – Monitoring Infrastructure for the OpenFlow Network

Most of the tools Stanford used are standard ones like *ping*, *tcpdump*, and *wget.* All other tools used for compiling the data are listed in [1]. In select cases, special tools were deployed to do the monitoring based on availability of probes and availability of API. For instance, NOX controller does not expose an API for querying the control plane statistics. In that case, a *tcpdump* was performed to archive the controller communication, then that packet capture was parsed using *oftrace* to reveal the statistics needed. Secondly, in the event no spare probes were available, user machines were used as probes by running *ofpeck* on them and having them report the data plane statistics directly to the *mysql* database.

4.2 Target Software-Defined Network

The target for the Stanford deployment was to create a large OpenFlow network that is safely used for both production and experimental traffic.

4.2.1 Objectives

The main objectives of the Stanford OpenFlow deployment were the following:

- Motivate the need for SDN through innovative experiments;
- Understand and verify the new SDN technology;
- Contribute back to the OpenFlow specification and community.

Exploration and understanding were essential at the following 3 layers:

1. **Data plane layer**: Several vendors were building new OpenFlow-enabled switches or adding OpenFlow support to their existing switches. A platform for experimenting with these switches was needed. The OpenFlow deployment serves this purpose. Some of the outcome from the deployment includes the certification of the stability of each switching hardware device, and the analysis of their performance limits.
2. **Controller platform**: Similar to the data plane, Stanford wanted to understand the capabilities, performance limits and stability of controller platforms released by open-source and commercial vendors. The deployment served this purpose as well.
3. **Applications or Innovation**: The driving force for SDN is the need to innovate in the network and break free from its ossification. For that purpose, several experimenters built innovative experiments or applications to showcase the motivating need and the potential of the new architecture. Videos of experiments are available in [2]. To allow these applications to be realistic, they needed to be run in a working production network. This was achieved using FlowVisor-based slicing that allowed creating a coexisting slice of the network to be used for the purpose of experimentation.

4.2.2 SDN Architecture

The final network spanned these two buildings with varying coverage and each being operated as a separate island:

- **William Gates CS building**: The production-use network in the 3A wing of William Gates Building at Stanford University was OpenFlow-enabled. In the network, six 48-port 1GE OpenFlow Ethernet switches, 30 WiFi APs, and 1 WiMAX base-station were deployed. The testbed includes the following devices:
    - OpenFlow-enabled switches from HP (ProCurve 5406ZL), NEC (IP8800), Toroki (LS4810) and Pronto (3240 and 3290). VLAN configurations on each switch were used to isolate the legacy non-OpenFlow network from the OpenFlow network.
    - WiFi APs based on the ALIX PCEngine boxes with dual 802.11g interfaces. The APs ran the Linux-based software reference switch from the OpenFlow website, and were powered by passive power over Ethernet to reduce the cabling needed for our deployment.
    - WiMAX base-station built by NEC.

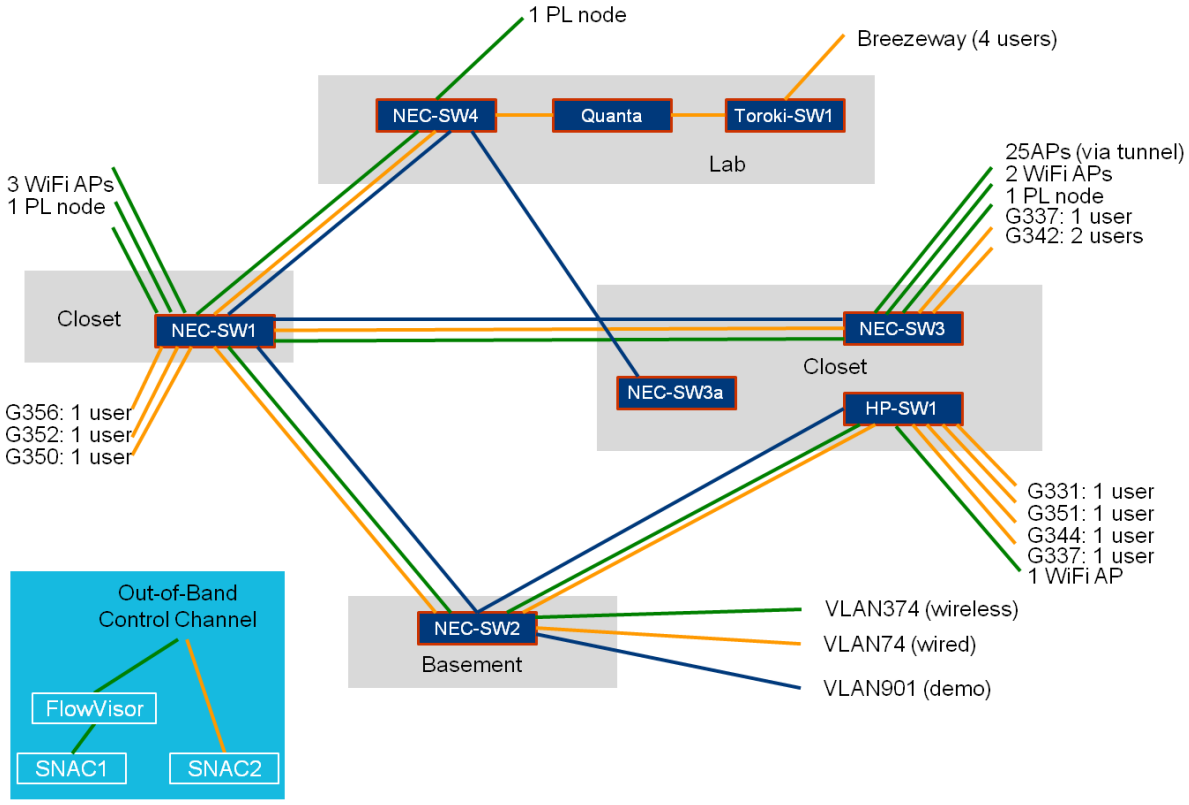

Figure 15 – OpenFlow-enabled 3A Wing of William Gates Building

- **Paul Allen CIS building**: One VLAN (viz., VLAN 98) spanning the entire building was OpenFlow-enabled. The network had six 48-port 1GE OpenFlow Ethernet switches, and 14 WiFi APs. It connected servers in class rooms and wireless users to the Internet. The testbed, illustrated in Figure 16 included the following devices:
    - OpenFlow-enabled switches from HP (ProCurve 5406ZL and ProCurve 5412ZL), with one VLAN being OpenFlow enabled.
    - WiFi APs based on the ALIX PCEngine boxes with dual 802.11g interfaces.

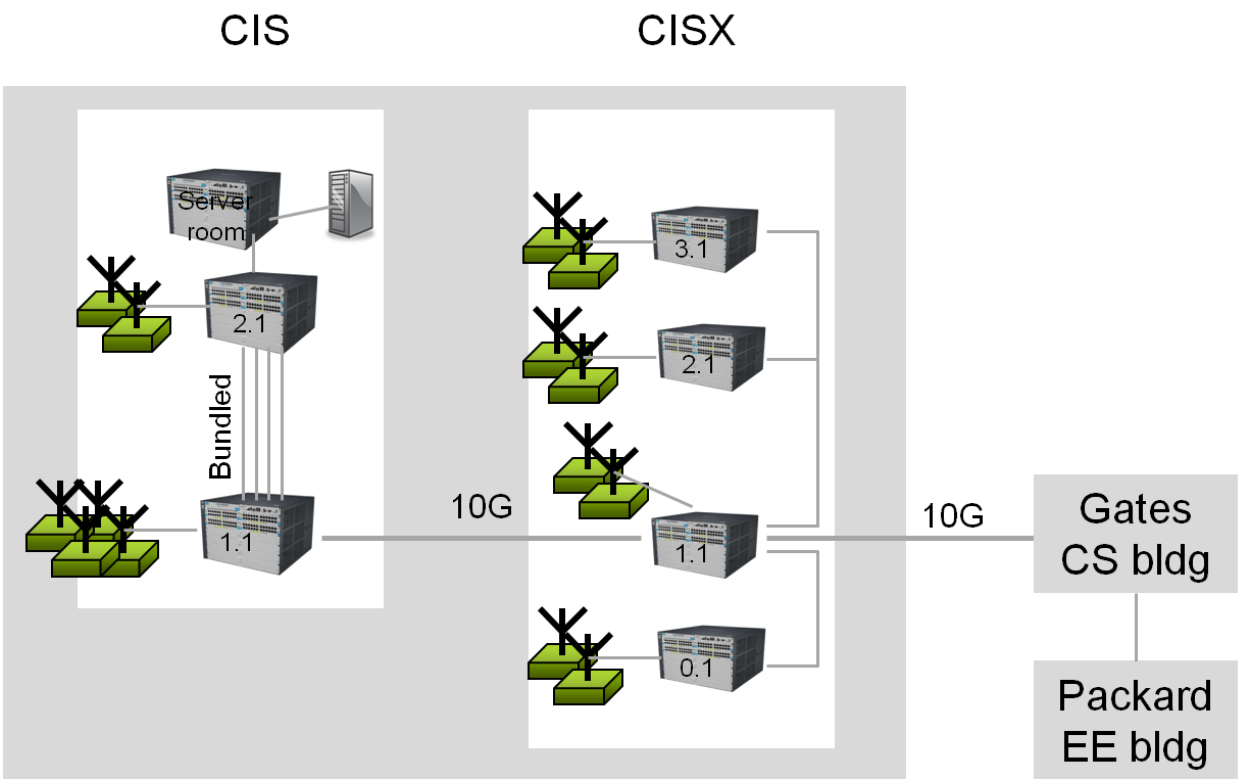

Figure 16 – OpenFlow-enabled CIS/CIX building network

4.2.3 Migration Approach

The approach undertaken was to gradually move individual users and then individual VLANs to OpenFlow-based control. To manage the risk involved in deploying the new technology, Stanford undertook the following 4 main phases:

1. **Add OpenFlow support on hardware**: Most of the hardware (viz., HP Procurve, NEC IP8800 and Pronto 3290) needed a firmware update to provide OpenFlow support. This was a one-time step.
2. **Verify OpenFlow support on switch**: OpenFlow support was verified by adding an experimental VLAN or test hosts to the switch, and allowing those VLANs or hosts to be managed by the external controller. Once the correctness in behavior was verified, Stanford moved onto the next phase.
3. **Migrate users to new network**: To minimize risk, a new non-OpenFlow network was created and users were safely migrated to that new network before using OpenFlow for production traffic. The main steps in this phase are the following:
    a. Add new Production subnet;
    b. Gradually add/move users to new subnet;
    c. Verify reachability within new subnet.
4. **Enable OpenFlow for new subnet**: Once the new subnet was functional, OpenFlow-control was enabled for that subnet by configuring the controller. Again, correctness and

reachability, performance, and stability were verified using the standard monitoring tools described in section 4.1.2, and user experience information collected through surveys.

Note that the wired production VLAN spanned both OpenFlow and non-OpenFlow networks, while the wireless production VLAN was exclusively managed by OpenFlow.

The eventual goal for the Stanford deployment was to expand the OpenFlow support to several other L2 VLANs and then interconnect them at a L3 router.

4.2.4 General Considerations

Typically during migration, most network administrators blame the new technology ("OpenFlow" in our case) when something goes wrong. This happens irrespective of whether the new technology is the true cause or not.

4.2.5 Target Dependencies

The target network deployment included several different controllers, including:

- NOX;
- SNAC;
- NEC Trema;
- BigSwitch controller.

Along with the controller, Stanford also tracked different versions of the switch firmware, the FlowVisor, GENI orchestration tools, and other demo applications. Stanford relied on the release management system of each of these softwares to ensure that they were running a comptabile, and well-functioning system.

4.2.6 Tool Requirements

Section 4.1.2 discussed the monitoring infrastructure used for verifying the health and status of the target network. Besides this, Stanford started building several debugging tools for troubleshooting issues that came up in the network. Most of these tools are online:

- oftrace: OpenFlow control traffic dump analyzer/tracing library [3]
- wireshark dissector for OpenFlow [4]
- mininet: Network emulation package that uses network namespaces [5]
- ofrewind: Replaying network events by replaying control and data plane traffic [6]
- Hassel and NetPlumber: Real Time Network Policy Checking and debugging [7]
- ATPG: Automatic Test Packet Generation for network debugging [8]

4.2.7 GAP Analysis

During upgrade, strict procedures were followed to ensure that only one change was undertaken at any point and also to revert back changes undertaken in the event the network state deteriorated. It would have helped if there had been safeguards in place within the switch firmware or OpenFlow controller to automatically revert configurations in such cases. This was being investigated by the GENI Emergency Reset group as well.

It would have also been helpful if the SDN system, including the OpenFlow Controllers and Switches, allowed for stronger interoperability between the OpenFlow part and non-OpenFlow part. To list a few of the features that were lacking when it comes to deployment in the enterprise campus network:

- The controllers used did not support the Spanning Tree Protocol (STP);
- The SDN system could not discover the non-OpenFlow switches in the topology;
- The switches used did not work well with LACP aggregation;
- The SDN system did not have complete visibility about flows and users that spanned both OpenFlow and non-OpenFlow segments of the network.

4.2.8 Post-Migration Acceptance

Using the monitoring infrastructure, information was collected, as depicted in Figure 17 and Figure 18. That information was used to determine if the network behavior was acceptable.

- **Correctness and Reachability**: Reachability was verified within the new network using user-generated traffic as well as probe-generated traffic. Completion of the requests made were the main factor that confirmed correctness and reachability.
- **Performance**: As described in section 4.1.2, statistics were monitoring in control plane as well as data plane. These statistics were correlated to identify anomalies and incorrect behaviors.

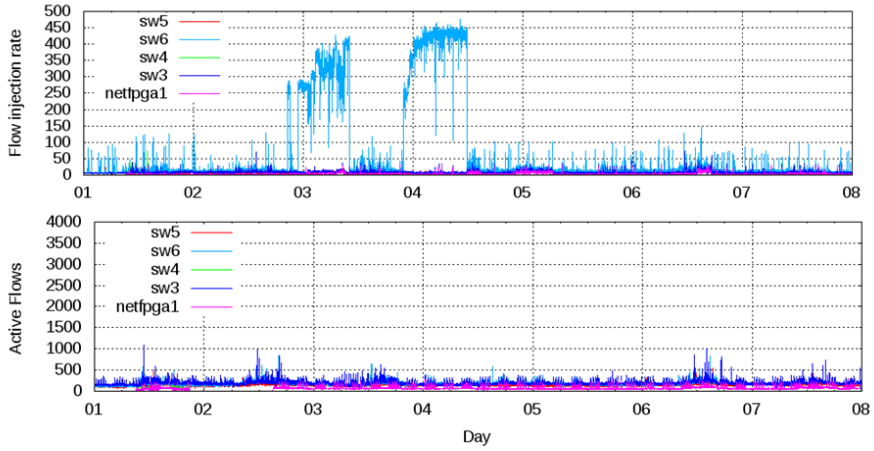

Figure 17 – Illustration of Control Plane Statistics from Dec 1-7, 2010 when SNAC Controller was used[1]

[1] Note that flow_setup_delay, RTT and wget_delay were collected using probes

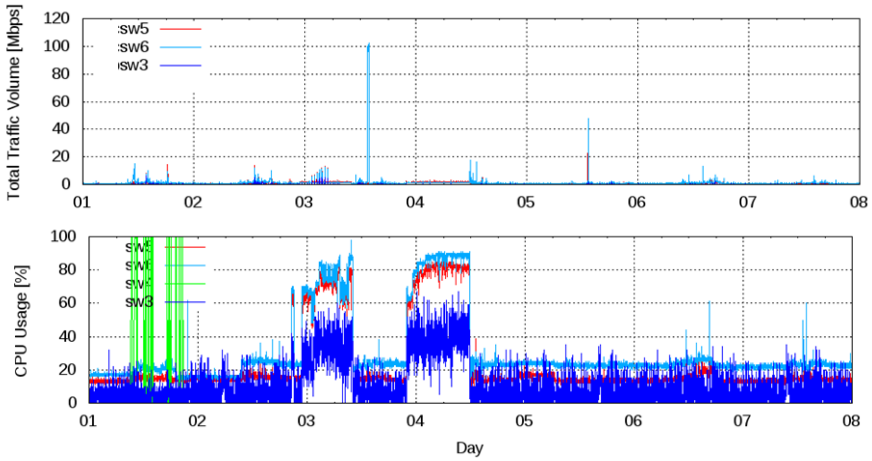

Figure 18- Traffic Volume and Central Processing Unit (CPU) Usage[2]

- **Stability**: For stability, the statistics were monitored over a long period of time. The statitstics were frequently plotted on a chart, as shown in Figure 19, to verify stability and health of the network.

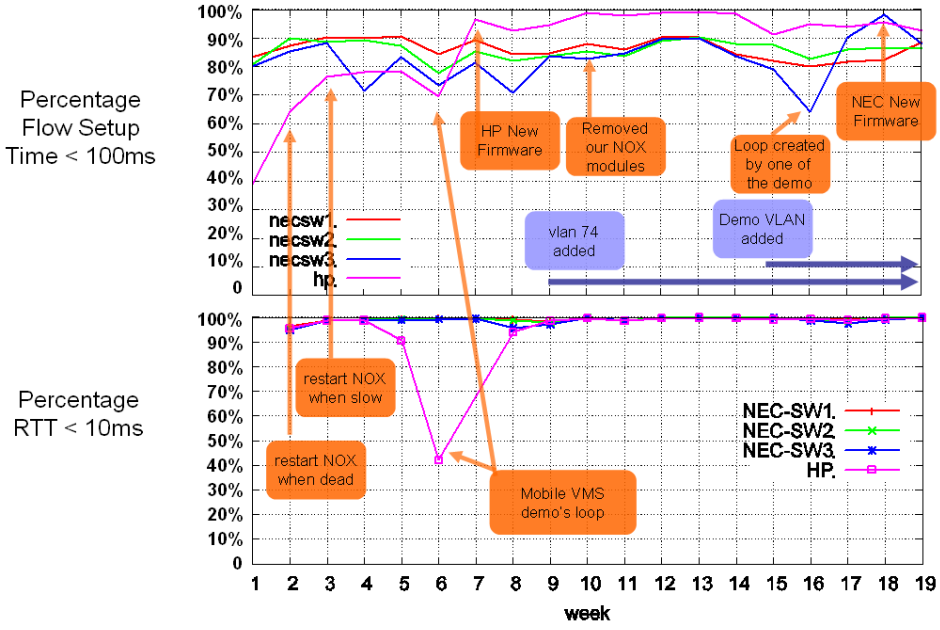

Figure 19 – Progression plot of the data plane statistics to verify stability.

4.2.9 Services Acceptance

The stability of the network gradually improved as the switches, the controllers and understanding of the network matured. The user-level survey undertaken regularly stopped bringing out any relevant data because the users started seeing consistently acceptable service.

[2] Traffic volume and CPU Utilization were collected using Simple Network Management Protocol (SNMP).

4.2.10 Migration Timeline

Stanford campus network OpenFlow migration occured over 3 phases:

1. Proof of concept with WiFi rollout and small production network
2. Slicing for research and production with larger production network
3. Production deployment across 3 buildings

During each phase, there was a 1 week planning, 1 week change, followed by 6 months production use.

4.2.11 Skill Sets Requirements

The expertise and skill sets required included the understanding of VLANs and management of the HP switch. There was also a great need for shell scripting expertise to set up network monitoring to detect any anomalies.

4.3 Additional References

Most of the Stanford OpenFlow deployment details are documented through deployment guides [9] and wiki pages and status reports [10].

5 Scope, Objectives and Terminology

This section provides an overview of the scope and objectives as described in the Migration WG charter, and defines the appropriate terminology and structure of a migration use case. The Migration WG charter lists the following goals, which are in scope and addressed in the subsequent sections of this document:

a) Network scenarios and use cases are to be identified along with deployment guidelines and recommendations.
b) The target network and its core requirements must be fully identified. Not all requirements of the traditional starting network may be met, at least initially, by the target software-defined network.
c) The objective is to simplify the network and lower the cost of operation. A secondary goal is to improve utilization.
d) The migration itself to the target network can be a source of risks. Outages during a migration, impairment of diagnostic and monitoring tools, or simply the scale and performance of the new technology, are all conditions that may be encountered during the intermediate steps. It is a requirement of this document to define guidelines, systems, and tools to facilitate and validate the steps required to migrate to the target network.

Based on the OpenFlow standard, the following objectives and high-level requirements are to be met:

a) The target network software must support programmability, through Application Programming Interfaces (API) capable of extending/combining functionality through the exposure of the underlying device features.
b) The target network must be serviceable, supporting dynamic software updates, with minimal service interruption and with automated updates and rollback.
c) The target network must support heterogeneity, with multiple devices from different vendors supported. Service migration must be considered over this heterogeneous infrastructure. Device-specific methods will need to be defined, but generic workflow orchestration infrastructure and tools are an option.
d) The target network must be maintainable within the necessary assembly of software, tools, and simulators. Either existing tools must be demonstrated to work with the target network, or alternatives must be defined or developed to ensure the operational transparency.
e) The starting network may require preparation and need to transform into a clean intermediate state from which the rest of the migration can proceed safely. Recommendations, guidelines, and tools for this preparation phase need to be specified.
f) Once the migration has completed, the target network must be validated against a documented set of requirements or expectations. Guidelines, systems, and tools must be identified to validate the completed migration.

5.1 Use Case and Migration Terminology

The deployment scenarios considered by the Migration WG are broad and varied. We provide definitions, deployment categories, and terminology here to ensure the document remains consistent with the WG charter.

5.1.1 Migration Approaches

The Charter specifies two migration approaches. The first approach, illustrated in Figure 20, is the direct method of upgrading existing networking equipment with OpenFlow Agents and decommissioning the Control Machine in favor of OpenFlow Controllers and Configurators.

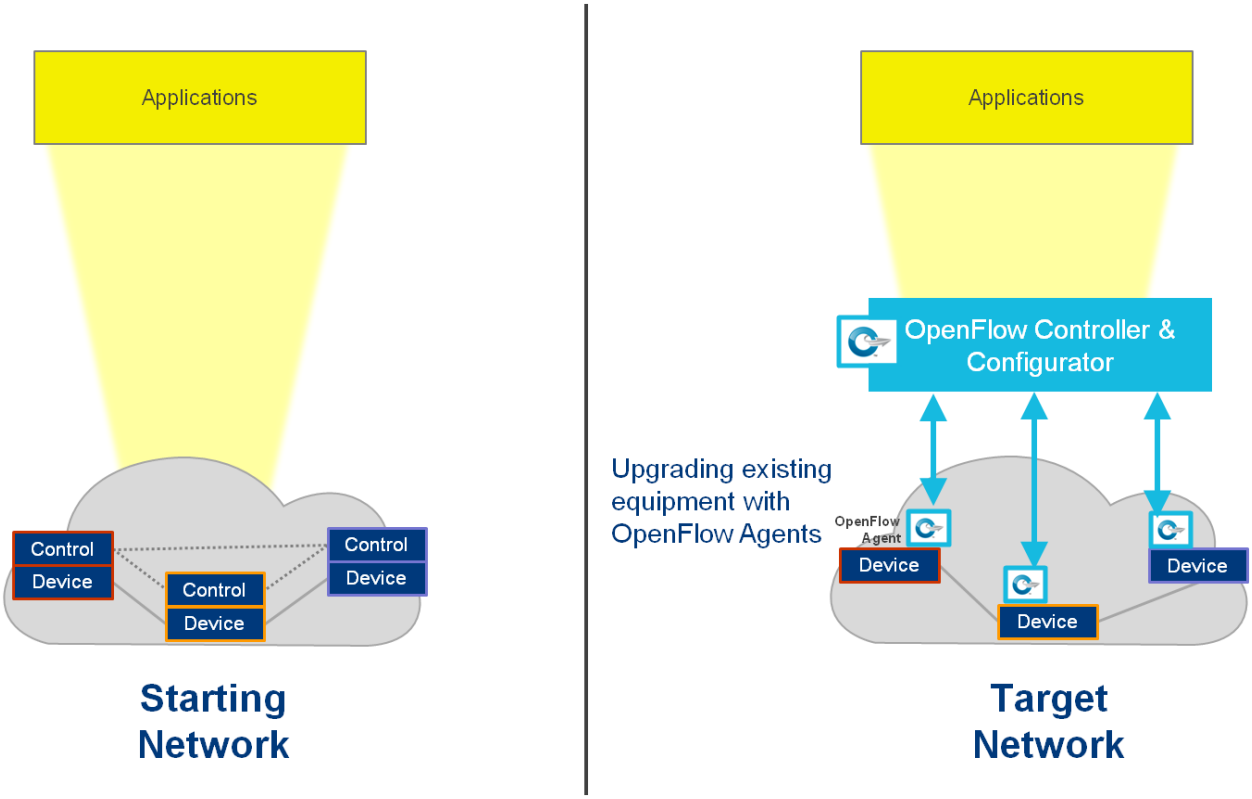

Figure 20 – Direct Upgrade

The second approach includes a phased approach, illustrated in Figure 21, in which OpenFlow devices are deployed in conjunction with existing devices. Network operations are maintained by both the existing Control Machine and by OpenFlow Controllers and Configurators. Once services have been migrated to the OpenFlow target network, the starting network, including the devices and control machine, is decommissioned.

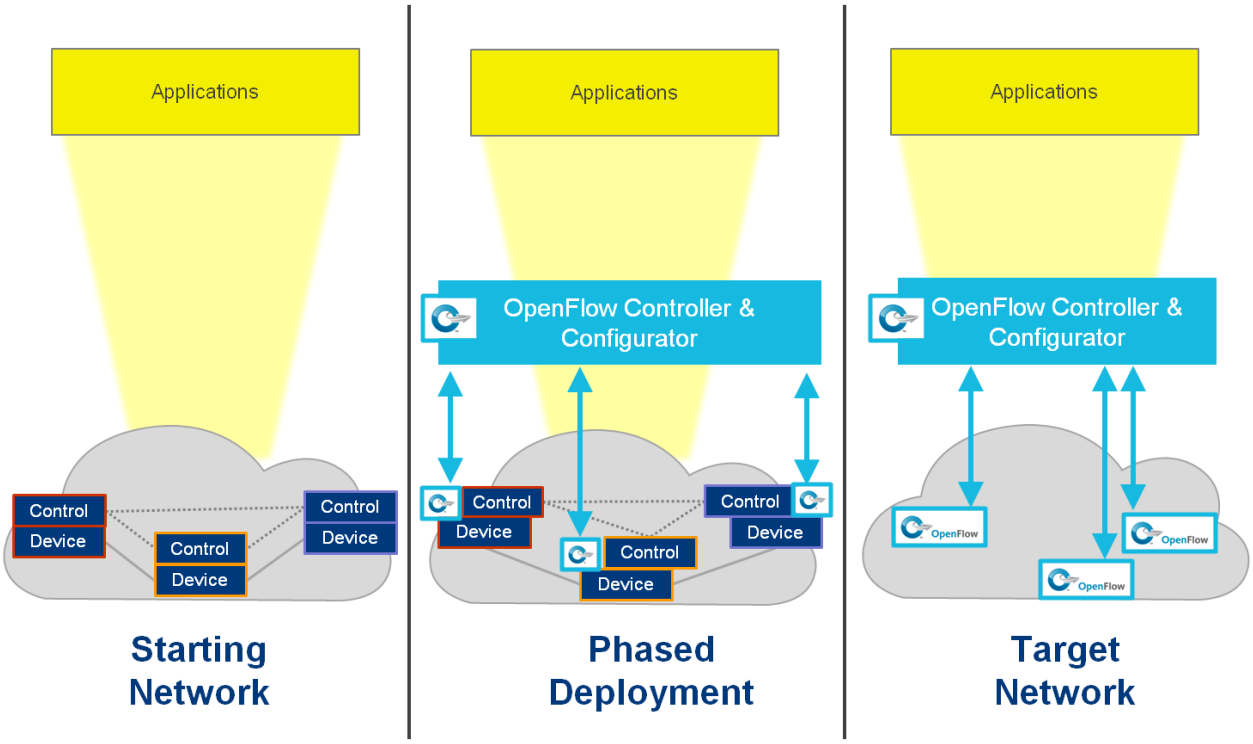

Figure 21 – Phased Migration

Network migrations vary as much as networks do. They can include the two Migration WG charter cases – namely direct upgrades of devices with OpenFlow Agents, or the parallel introduction of OpenFlow devices with the existing equipment, which is later decommissioned. In addition to this, we would find Greenfield deployments; partial migrations, where various flavors of SDN are utilized for a portion of the network; or any non OpenFlow variant of SDN.

Additional types of migration include migration of overlay services (Figure 22) or hierarchical OpenFlow control (Figure 23).

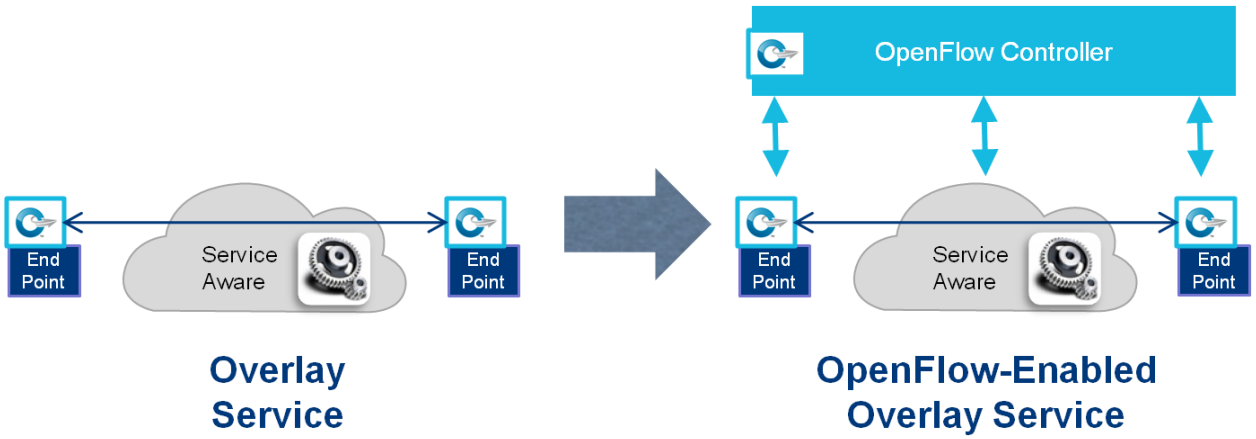

Figure 22 – Overlay Service Migration

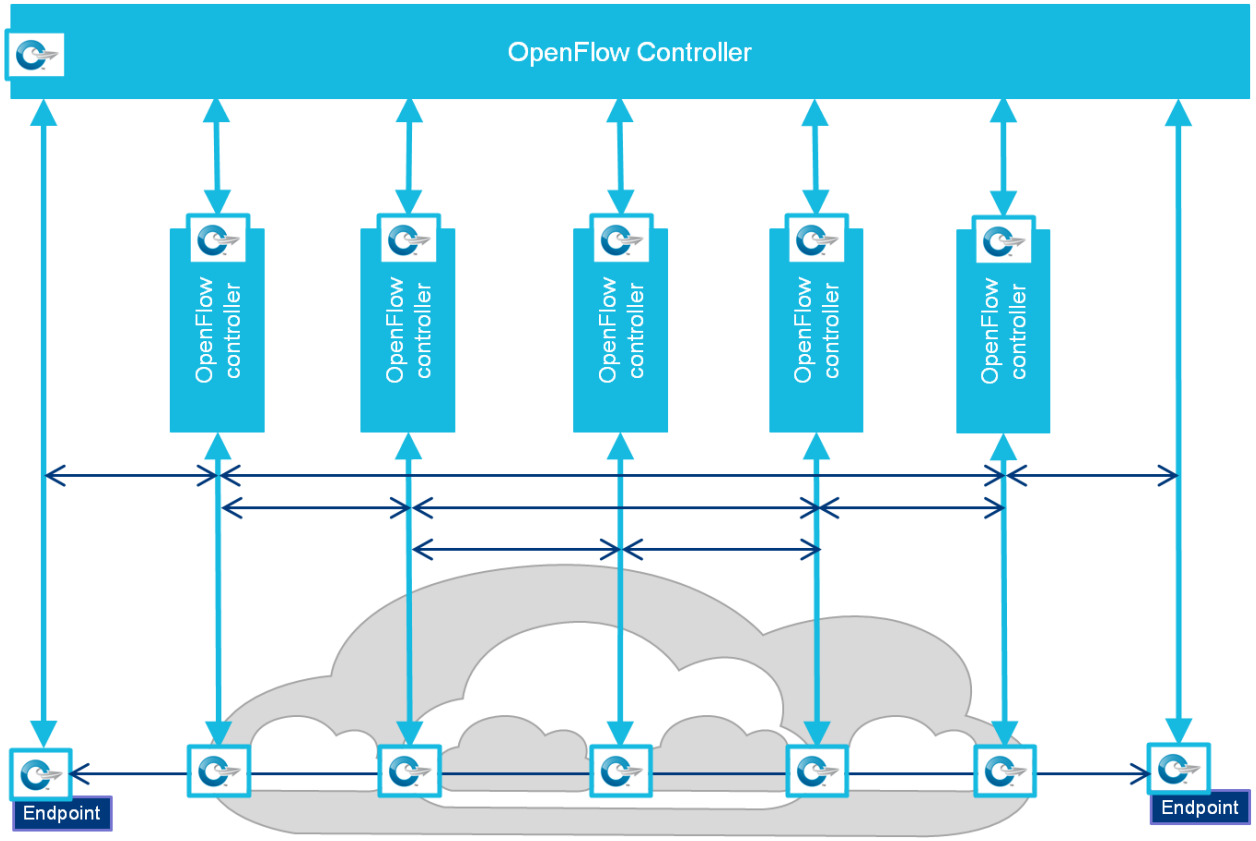

Figure 23 – Multi-Service Hierarchical Network

Migration can also include partial migrations where domain boundaries are OpenFlow enabled (as between Access and Metro in Figure 24) while the domains are not. It can also include the case where some domains are OpenFlow enabled but some adjacent domains are not (as the

Core in Figure 24).

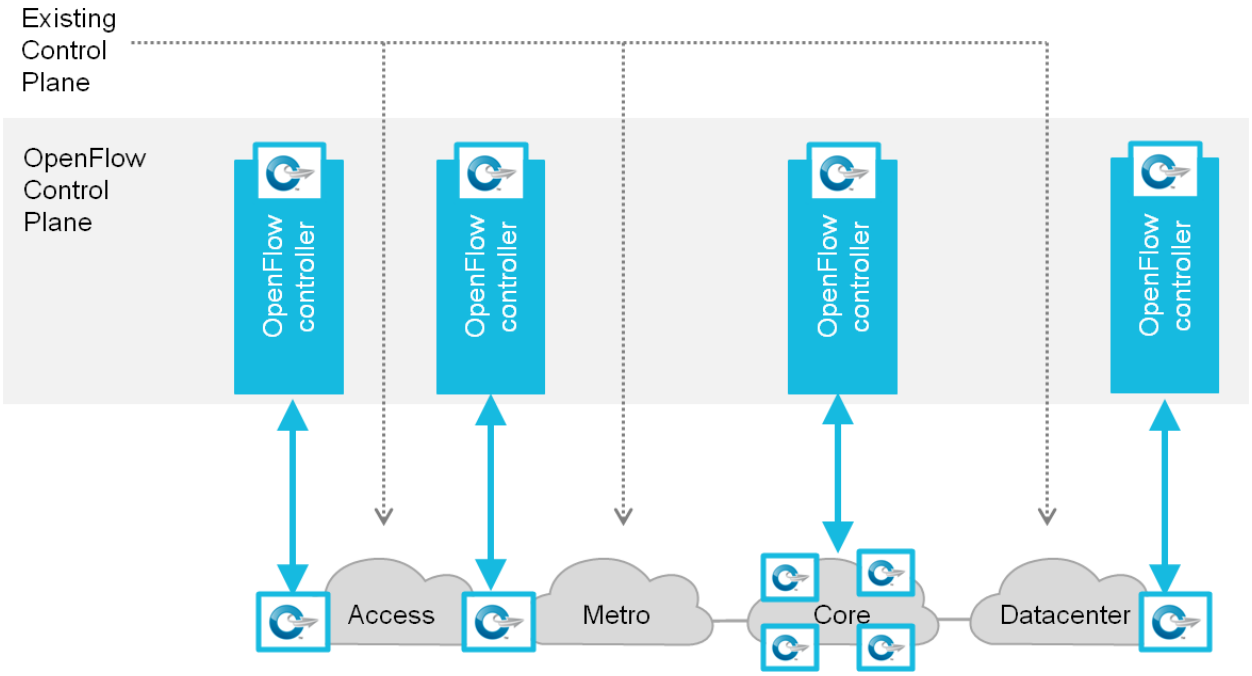

Figure 24 – Diversity in Network Deployments

5.1.2 Migration Approaches – A Closer Look

It is important to understand difference between OpenFlow implementations available on network devices. Devices can be classified as a Legacy, OpenFlow, or Hybrid as depicted in Figure 25. Legacy devices are traditional Switch/Routers with integrated control and forwarding plane. OpenFlow devices are switches with only OpenFlow forwarding planes, with the control plane residing external to the device. Hybrid OpenFlow Switches refers to devices with both legacy control and data plane and OpenFlow capabilities.

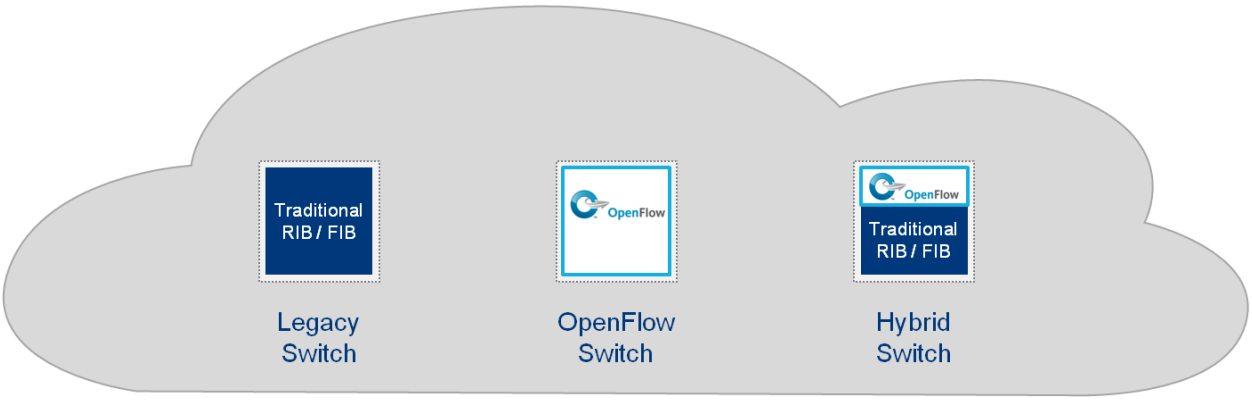

Figure 25 – Types of Devices

Of the two migration approaches defined by the Migration WG Charter, we can restate the migration approaches using the above terminology as follows:

- **Greenfield Deployment**: The Greenfield deployment, is one where there is either no existing deployment or the Legacy Network is upgraded to become OpenFlow enabled and the Control Machine is replaced with an OpenFlow Controller.

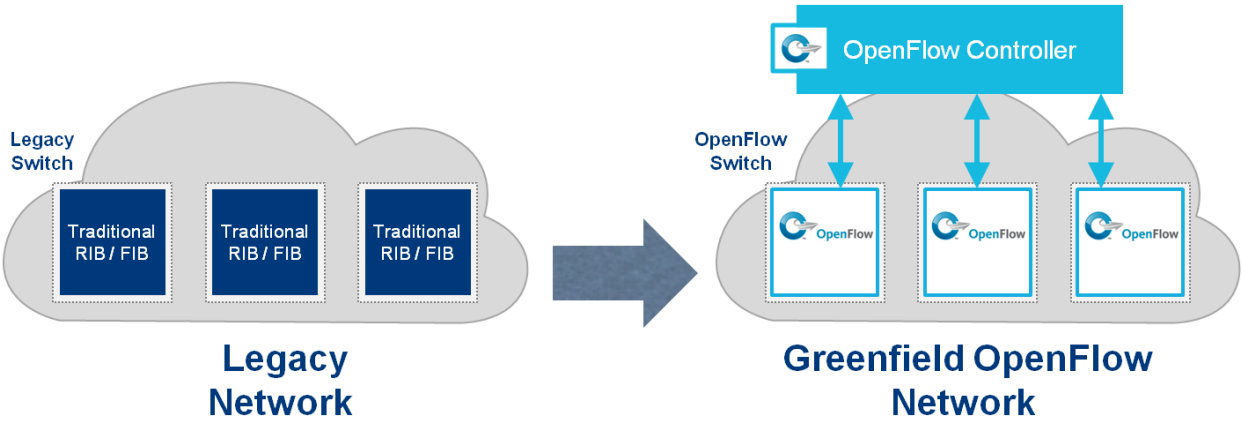

Figure 26 – Greenfield Deployment Model

- **Mixed Deployment**: This migration approach assumes that new OpenFlow devices are deployed and will co-exist with other traditional switches/routers and must communicate with legacy Control Machines. The new OpenFlow Controller and the traditional devices will need to exchange routing information between each other via the legacy Control Machine.

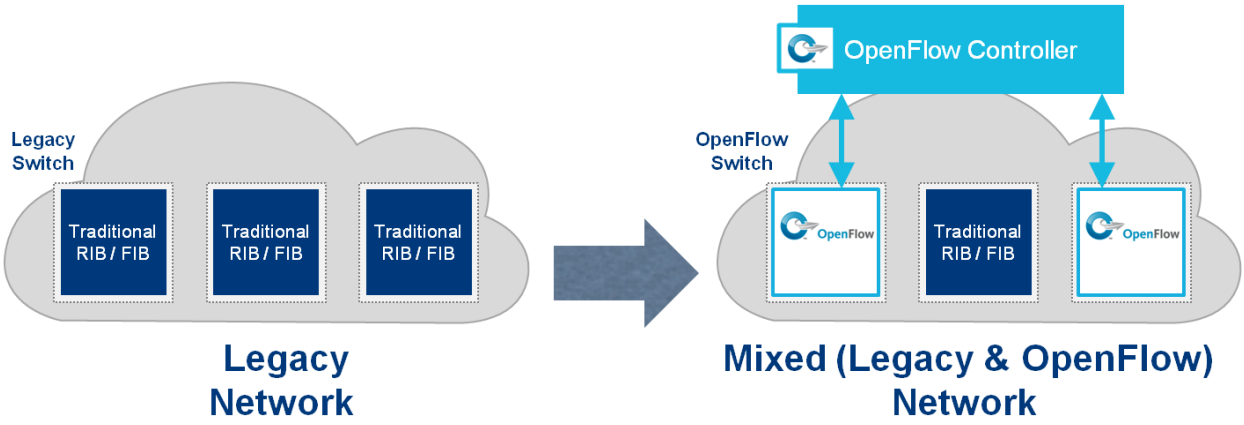

Figure 27 – Mixed (Legacy and OpenFlow) Network Deployment

- **Hybrid Network Deployment**: In this case, both Mixed Network deployments and Hybrid devices with both legacy and OpenFlow functionality can coexists. In this scenario, the Hybrid devices both communicate to the OpenFlow Controller as well as the legacy Control Machine.

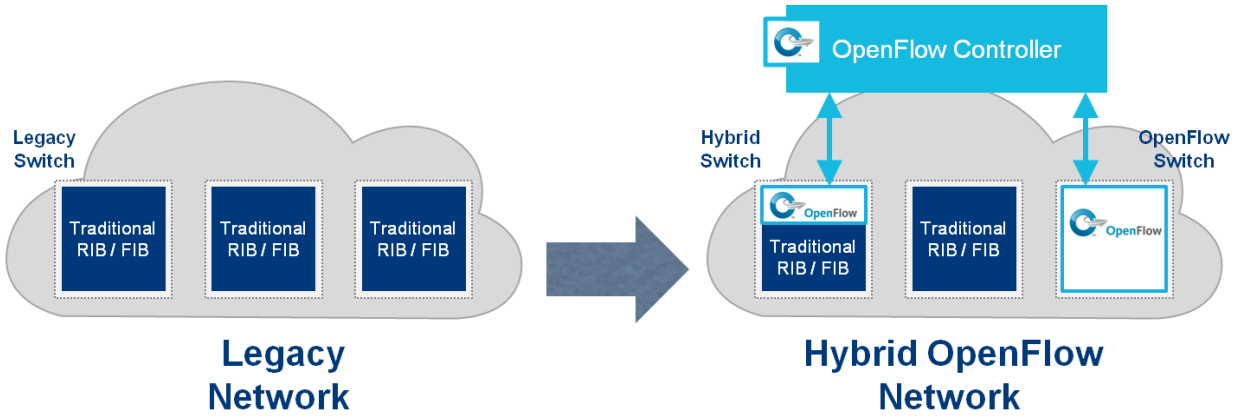

Figure 28 – Hybrid OpenFlow Network Migration

The Mixed and Hybrid migrations are examples the Phased Migration contemplated by the Migration WG Charter. In all cases, these migrations contemplate a single network domain migration. In many cases, the motivation for migrating towards OpenFlow is to enable selected services. In other words, OpenFlow is desired to enable an end-to-end service. Figure 22 simply illustrates the fact that services may be overlayed on top of a conceptual network. Supporting such a service with OpenFlow will require the introduction of an OpenFlow Controller along with underlying OpenFlow devices, either in a Greenfield, Mixed, or Hybrid configuration.

As we peer into the network carrying the overlay service, we begin to see that it is not comprised of a single network domain. In fact, the network may infact be heterogenous layers of networking technology, each capable of being addressed with OpenFlow. One narrative that could be used to define this picture is that the outer devices map Ethernet flows between the end points, onto VLANs. The innermost devices represent two discrete MPLS domains used to carry the VLANs through the network. In this example, the identified devices may be Greenfield devices; however, they are not limited to a single protocol implementation of OpenFlow. In other words, the OpenFlow Controller is capable of understanding and configuring the end-to-end network.

The same end-to-end migration could be traversing a variety of network segements and technologies. For example, and service delivered to an end user's cell phone may traverse a multitude of networks and related technologies. Each technology may be service enabled through OpenFlow, but infact may translate into completely different technologies.

Figure 24 identifies such a distribution where the end user's mobile phone is connected to an Access, Metro, and Core Network; and ultimately the Datacenter where the service originates. This illustration suggests that such a deployement would involve a series of Mixed or Hybrid networks. Some or all could include Greenfield deployements, but less likely from an end-to-end perspective.

5.1.3 Types of Networks

The Migration WG will consider use cases in four major categories: Campus, Enterprise Datacenter, Multi-tenant Datacenter, and Service Provider/Wide Area Network (WAN). Each of

these categories has various subcategories to consider. While not an exhaustive list, we attempt to outline some of the characteristics of each category.

- **Campus Networks** are typically composed of multiple buildings, each building typically having a wiring closet. The buildings are interconnected with a central operations center. Components of the Campus network include a Campus wide backbone with an egress point to the Wide Area Network that is typically associated with a datacenter. In many cases, the network will include logical network/datacenter partitioning – be they for different departments, administration facilities, or campus wide IT resources.
- **Enterprise Datacenters** can range in size, but are typically composed of networking resources used to interconnect various sub-networks of servers (physical or virtual) together with associated storage (e.g. Network Attached Storage (NAS) or Storage Area Network (SAN)), security, and networking functions (e.g. WAN acceleration, Load Balancing, etc.). Requirements for software-defined networking can vary, but application driven services rank high on the list.
- **Multi-Tenant** Datacenters have benefitted greatly from software-defined networking. These datacenters share many aspects of the typical Enterprise Datacenter, however, multiple tenants must typically share the physical resources. Virtualization of computing resources is almost a necessity, with robust features such as Virtual Machine migration facilitating a variety of capabilities, including resource balancing, maintenance, and disaster recovery. Soft Switches within the computing resources themselves are a dominant component of the architecture. The net effect is that portions of the datacenter move and change, demanding that the overlay network must move and change to echo those changes. Increasingly, however, software-defined networking devices help address these requirements.
- **WAN/Service Provider/Carrier Networks** introduce significant diversity. Service provider network architectures and requirements vary. For example, a Mobile Cellular Service Provider will have a radio network; along with a mobile backhaul network which hands off to an access network and ultimately a core network. Different applications of OpenFlow and SDN are being developed and deployed today. Service Providers are using OpenFlow to manage their inter-exchange resources, such as e.g. Google for wide-area network connectivity between datacenters, to ensure appropriate bandwidth is available at appropriate times. Note that this is similar to an overlay-type application. Many other use cases are being developed by the industry, with software-defined solutions addressing Layer 0 through 7 network domains.

5.1.4 Network Layers

In addition to network type diversity, there are many layers at which networks operate. While the initial target architecture and functional capability for OpenFlow was based on Layer 2 flow forwarding control, OpenFlow has evolved to include Layer 0, 1, 2, 2.5, 3, 4, and even layer 7 based network applications. As in current network practice, different data plane layers are associated with different control planes and operational models, and, hence, require appropriate migration recommendations; whether it is the deployment of application driving capacity

scheduling at lower layers, or service chaining based on explicit Deep Packet Inspection criteria at the IP Edge.

If we consider our definition of a Hybrid device, it reflects a network device that can be controlled by an OpenFlow Controller, but also peer with a legacy Control Machine. As we delve deeper into real-world networks, we find that individual devices can operate at multiple layers. As a result, multiple Control Machines may be at play, as well as multiple OpenFlow Controllers. Figure 29 attempts to illustrate such complex networks, identifying where individual network layers may be migrated and where multiple layers may be migrated.

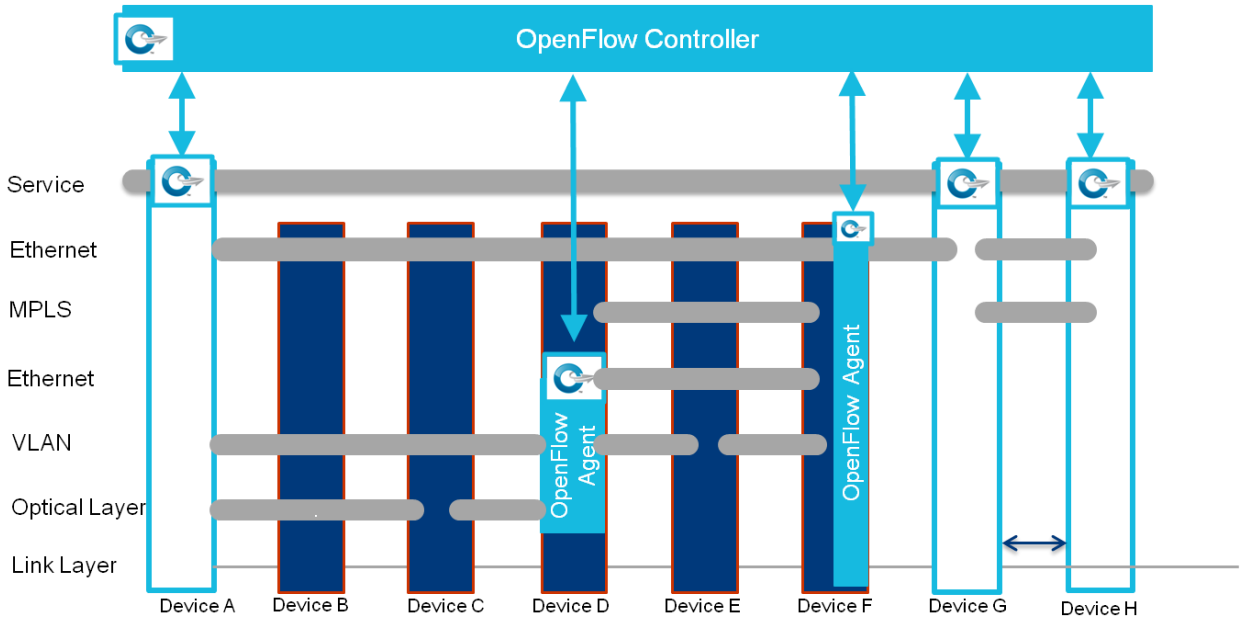

Figure 29 – Example Layering Impacts on Migration

Figure 29 illustrates each physical device involved in the delivery of an end-to-end network. Device A, for example, provides Layer 0 connectivity, but terminates the Optical signal and introduces a VLAN/Ethernet payload onto the Lambda. We suggest that this device may include an OpenFlow forwarding plane to decide which Lambda to utilize.

Device B is simply an Optical switch, where the signal or payload does not terminate (e.g. wavelength is switched by analog technologies). Device C is an Optical-Electical-Optical switch that simply regenerates the VLAN header, possibly “cross connecting” onto another VLAN.

Device D could be seen as a Broadband Network Gateway. These devices can operate at a wide variety of network layers. Often times they act as the first layer 3 point within the network and will peer with the Core Network (often MPLS based). Because of this unique position within the network, services are often introduced at this point in the network. OpenFlow based Service Chaining is widely viewed as an important use case at this point in the network. The illustration does not illustrate that we have terminated layer 3, but does show that the end Ethernet frame is carried over a variety of underlying technologies and Control Machines. This futher illustrates

the fact that OpenFlow may be active at multiple layers within Device D, and as a corollary, multiple OpenFlow Controllers may be responsible for the device.

Device E reflects an intermediate Ethernet switch, or possibly an MPLS device that is not terminating a Label Switched Path (LSP). Devices F & G are MPLS switches, which either forward an LSP or terminate an LSP.

Finally, Device H is an Ethernet device which only see's Ethernet frames coming in on a port, and is ambiviolous as to how the Ethernet frame is delivered.

Migration Implications:

Given the provided details of Figure 29, we try and identify how these "real-world" issues can impact our views and definitions of a Network Migration. Three potential migration scenarios are shown, highlighting devices or layers that may be considered for upgrade.

- **Migration A – End-to-End Migration**: Illustrates an end-to-end Greenfield Migration with all devices operating at the same layer(s). All devices are migrated to be full OpenFlow devices controlled by a single OpenFlow Controller. If we consider the intermediate devices, it could be considered an example of Ships in the Night.

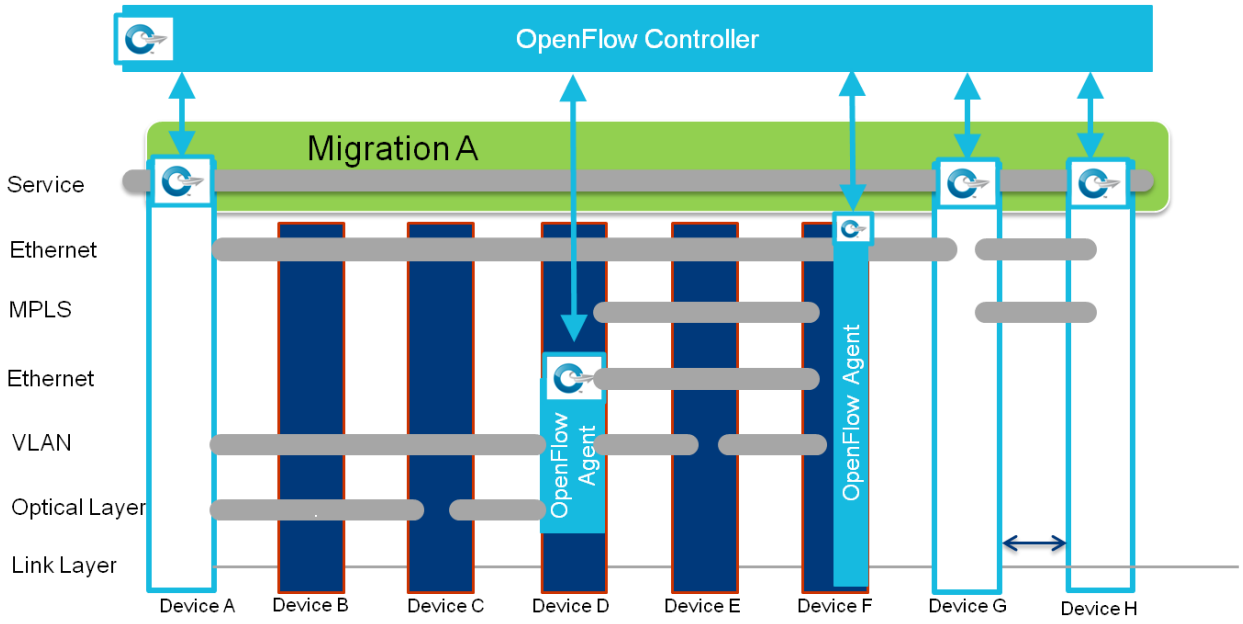

Figure 30 – Example End-to-End Migration

- **Migration B – Migration with Full-Stack Handoff**: Device F in this example is reflective of a Hybrid device (as defined above), while device G is a full OpenFlow device. Device F will still peer with the Legacy Control Machine, but at the same time will apply commands from an OpenFlow Controller. This would be an example of a Hybrid Network Migration.

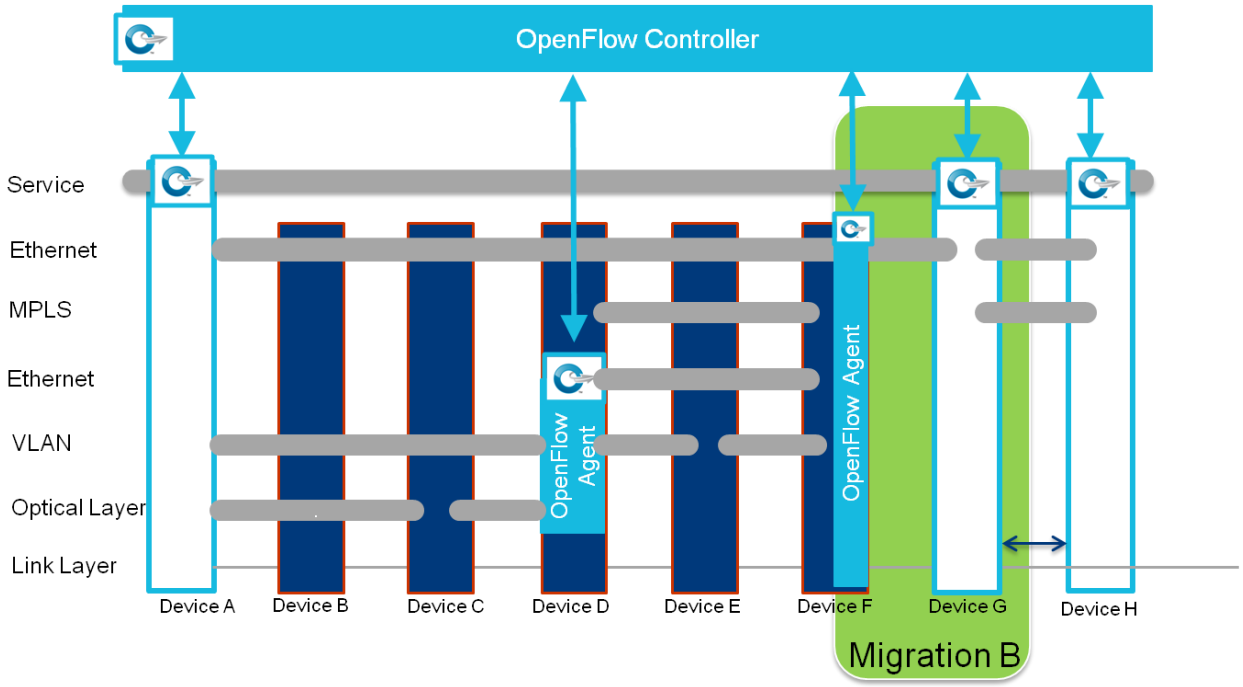

Figure 31 – Example Migration with Full-Stack Handoff

- **Migration C – Partial Stack Migration**: Only a few layers of a multi-stack device are migrated to OpenFlow (Ethernet, VLAN and Optical layers), where layers above and below (MPLS and Ethernet) may continue to use the existing Control Machines and protocols, but select layers are replaced with OpenFlow. For example, OpenFlow and OpenFlow controllers may be used to manage the Optical, MPLS and even Service Chaining features at this point in the network.

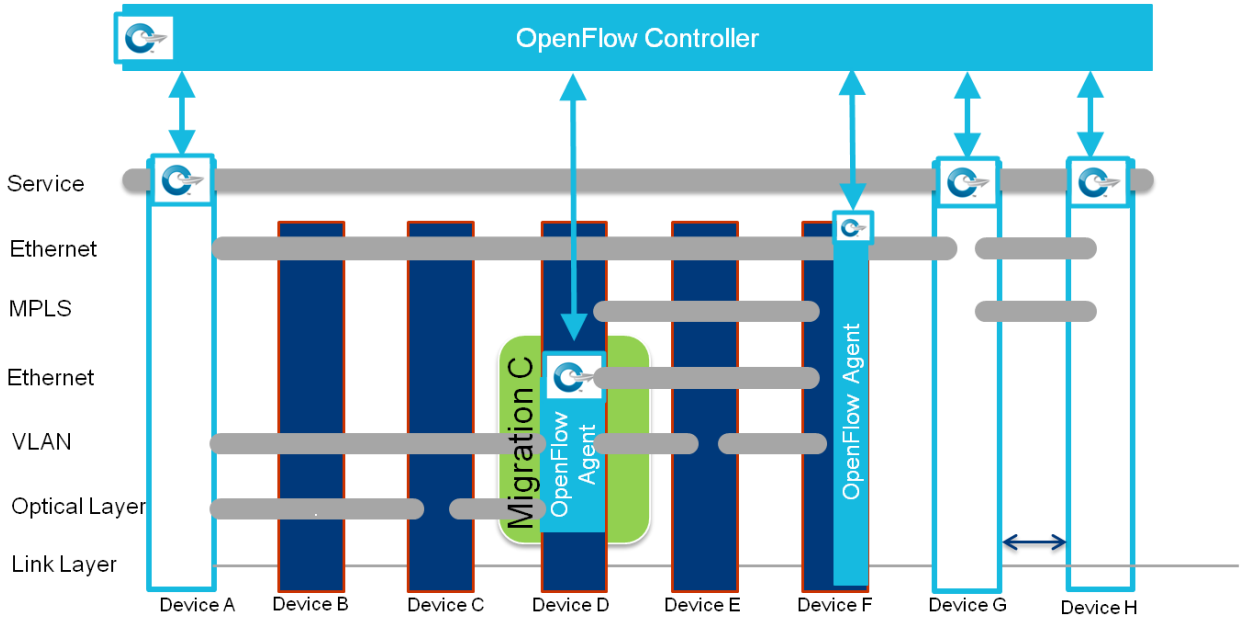

Figure 32 – Example Partial Stack Migration

5.1.5 SDN Deployment Architecture

An OpenFlow-based SDN deployment will include one or more OpenFlow forwarding devices controlled by a logically-centralized OpenFlow Controller and Agent. Physically, the Controller and Agent could be centralized on servers, or embedded within some or all of the OpenFlow forwarding devices. The reasoning for architectural choices made according to each individual use case should be fully documented.

Furthermore, the nature of the OpenFlow traffic between the Controller and Agent can impose various performance requirements on the deployment. For example, the Controller and Agent may simply be used for provisioning requests. These may be rare (e.g. scheduling of optical circuits) or more frequent (e.g. supporting VM migration within a datacenter). The traffic between the Controller and Agent may include signaling protocols that have not been migrated to OpenFlow, such as Dynamic Host Configuration Protocol (DHCP), Password Authentication Protocol (PAP), Challenge Handshake Authentication Protocol (CHAP), or more elaborate Internet Protocol Television (IPTV) or 3GPP signaling. The anticipated capacity required to support sustained and bursts of this traffic should be considered. Finally, in cases where exceptioned flows are diverted up to the Controller, the anticipated rate of exceptions should be fully specified. In cases where 100% of traffic could be diverted to the Controller, then very specific requirements should be considered when designing the forwarding plane and controller connectivity (e.g. if a forwarding device can pass 1Tbps of traffic and 100% of traffic could be exceptioned to the controller, then clearly 10 Gbps of connectivity to the controller is not going to be sufficient).

Finally, the flow types being delivered by the target software-defined network need to be considered. For example, if the traffic type is exclusively real-time voice traffic, requirements will be different than a network carrying multi-cast video traffic, or simply general best effort data traffic. Overall, a clear understanding of the traffic pattern is crucial for proper migration planning.

5.1.6 Security Considerations

Secure networks are critical to all businesses especially with their increased migration to the cloud and towards software-defined networks. SDN provides a centralized intelligence and control model, which provides much needed flexibility to combat threats against networks. Less proprietary and more granular security solutions are needed within modern networks. SDN can be utilized to manipulate the network flow paths based on traffic analysis and statistics provided by the controller. OpenFlow based SDN will provide granular policy management for multiple simultaneous policies. OpenFlow will provide an increased ability for flexible and intelligent responses to threats including rapid containment and isolation.

Similar to the requirements needed to confidently migrate data and infrastructure to the cloud, secure SDN will also be critical in providing network administrators the confidence to successfully migrate a traditional network towards software-defined networking. SDN networks are required to be just as secure as traditional networking. SDN has the potential of being even more secure than legacy methods through faster threat detection and more granular response

mechanisms. During the migration process, including while legacy and SDN networks coexist, maintaining secure and isolated networks is required. Existing security services will need to be migrated to an OpenFlow enabled SDN network. Ownership of security policies and resources, between starting and target networks, must be clearly and accurately maintained.

One way to migrate security mechanisms to an SDN network is to:

1. Add an experimental network slice (e.g. VLAN)
2. Harden and test that experimental slice with open SDN security solutions (e.g. Security Enhanced Floodlight).
3. Enable OpenFlow and secure SDN solutions on a new network slice (e.g. VLAN)
4. Gradually move users to this new secured network slide.

6 Common Best Practices

The use cases covered in this document exemplify migration strategies for different network types including enterprise/campus, WAN, and carrier/service provider networks. This section summarizes some of the lessons learned and deployment practices based on the migration approaches discussed in this document. It is critical to note that, during the migration of any production network, the main objective is to provide service continuity with minimum or no disruption.

6.1 Pre-Migration Planning

A successful migration includes a pre-migration planning phase where the following practices are recommended:

- **Gap Analysis**: A detailed gap analysis should be performed to understand the impact on existing services. For any identified gaps, ensure that alternate options are available to mitigate possible challenges that may be encountered during migration.
- **Check Lists**: Pre- and post- migration check lists should be created with specific samples of applications and/or source destination prefixes which will be used for connectivity and service continuity checks. Carrying out pre- and post-migration assessments in due course helps to ensure that non-migration related issues and failures can be excluded.
- **Back-out Procedures**: Mitigation aspects should be considered. A detailed method of procedure should be created to follow step-by-step migration process with back-out procedures clearly documented in case of unexpected results. It is worth investigating if back-out procedures that revert the configuration to the starting network can be automated to minimize disruption in case of deteriorated performance.
- **Feature-Set Analysis**: A detailed analysis of the OpenFlow features and desired capabilities should be performed on the controller and OpenFlow switch to ensure the feature set is consistent with the requirements.

6.2 Migration Process

The following practices are important for a smooth migration:

- **Provisioning Tools**: All the necessary network management tools should be provisioned for the migrated network for proper management and monitoring of traffic and devices during and post-migration.
- **OpenFlow Version Control**: As there are several versions of OpenFlow, it is important to verify compatibility between versions of OpenFlow protocol implemented in the controller and the switch.
- **Upgrades**: The OpenFlow devices must be upgraded to run the appropriate code and hardware firmware before the migration can be initiated.
- **Connectivity**: The connectivity between the controller and the OpenFlow devices should be confirmed.

- **Service Availability**: In a mixed environment, a dummy service such as customer VPN can be created to verify the service availability.
- **Troubleshooting**: Appropriate troubleshooting steps such as ping, trace or accessing an application can be employed to check the connectivity.

7 Conclusion

This document is the first deliverable of the ONF Migration Working Group whose charter is to produce methods for migrating network services from a traditional network to an OpenFlow-based software-defined network, and whose broader objective is to help the network industry accelerate the adoption of open SDN. Traditional networks in scope include WAN/service provider/carrier networks, datacenter networks, enterprise networks and campus networks. The initial focus of this work has been to examine real-world SDN migration use cases, gather best practices and make recommendations on migration methods, tools and systems.

This document provides a framework for migration methods exemplified by a set of target networks. This framework includes the description of target network core requirements, starting network migration requirements, phased migration requirements, and finally validation requirements to ensure a complete and successful network migration.

Enabling new services is often an important motivation for SDN migration. These services can be end-to-end, overlay on virtual networks, spanning several network segments and/or across several layers of networking technologies, some or all of which could possibly be addressed by OpenFlow. OpenFlow is still evolving as new use cases and deployment models are defined; OpenFlow could possibly address Layer 0, 1, 2, 2.5, 3, 4-7 applications. There is value in examining use cases from different network types to better understand unique migration strategies, tools and methods which could be specific to each service or network type given the large diversity of emerging SDN use cases. This document examines three use cases: Google Inter-Datacenter WAN, NTT Provider Edge, and Stanford Campus Network use cases.

Given that the Google WAN and Stanford campus software-defined networks use cases, both covered in this document, have been in production for a number of years, it can be concluded that traditional networks can be successfully migrated to OpenFlow-based software-defined networks leveraging the framework and approaches described in this document. Clearly, more work is still needed to cover the diverse range of possible SDN deployments. The ONF Migration Working Group will continue to revise and refine its recommendations for SDN migration methods, tools and systems, including additional SDN migration use cases as they are defined and deployed.

This is a living document that will continue to adopt new use cases and document the migration requirements defined within the ONF Migration Working Group’s charter. Each use case shall document a list of characteristics described in Appendix A.

The ONF Migration Working Group's future work, as currently planned, includes: publishing a document on SDN migration tools and metrics – along with working code for validating such migration metrics, and demonstrating a prototype migration tool chain, including simulators and software maintenance capabilities.

Appendix A: Use Case Requirements

This document is a living document that will continue to adopt new use cases and document the migration requirements defined within the ONF Migration Working Group's charter. Each use case shall document the following characteristics. Note that the detailed use case subsections may not correspond exactly to this structure as each use case may have different requirements.

A. 1 Starting Network

The starting network must be very well understood and documented. The type of network, the nature of the existing equipment (e.g. hardware and/or software switches/equipment), and the type of control machine (e.g. routing protocol, switching protocols) currently deployed must all be specified. In addition, the application of the starting network must be outlined, along with existing Operation Support Systems (OSS), tools, and diagnostic processes. A discussion of network resilience in the presence of migration outages (e.g. no outages, brief outages, or sustained outages tolerated) must further be documented. Finally, in the case where the starting network must be enhanced before it becomes suitable for migration, a description on the current status and the necessary preparation are required.

General Description

General description of the purpose of the network, main services delivered, operational mode and business/revenue objectives.

Operational Mode

A list of network layers and protocols that are used in the network, including a description of which ones are included or excluded from the migration; for example, inclusion of layer 3 edge router running Boarder Gateway Protocol (BGP) or exclusion of the optical layer network using Generalized Multi-Protocol Label Switching (GMPLS).

Deployed Equipment

An inventory of the deployed equipment within the network and a description of interfaces, traffic flows, etc. It may be appropriate to identify which equipment can be upgraded to support the target software-defined network.

Redundancy Model

A description of the redundancy model that is in place, if any. In the case where a redundancy model is in place, a specification of its functional requirements to be addressed by the target software-defined network.

Management Tools

A description of the management systems that are used to monitor, provision, commission, upgrade and troubleshoot the network including the protocols used to accomplish this, and if it is expected that the use of existing tools is continued in the target network.

Network Capacity

A description of the speeds and feeds, number of end users, anticipated volume of traffic per node, and/or the overall size of the network. Any capacity requirements that may impact the target software-defined network should be identified here. Specific consideration should be made with respect to limitations with OpenFlow (e.g. Access Control List (ACL) sizes).

Upgrade Requirements

Outage requirements should be specified so that target software-defined network can state whether these conditions can or should be met during a migration. What are the tolerated outages duration? Can the network be out for the weekend or a late night outage? Can the outage be service affecting, but sub minute or second? Must the outage be non-service affecting?

Availability

The current network requirements related to availability (e.g. five nines, four nines, three nines).

Problems and Challenges

Are there any particular problems with the starting network or challenges that motivate the migration towards the target software-defined network? For example, a campus network where tenants do not have any flexibility to use the network as a test bed. Another example could be the mobile backhaul, where configuration issues drive up the cost of deployment. This section focuses on the limitations of the starting network while the expected improvements and merits are described in the target network section 0.

Pre-Migration Assessment

The network should have a core set of requirements defined, with appropriate tools to qualify the existing performance levels. Included within this section would be a procedural test assessment that would be executed prior to the migration, providing a check point or benchmark for the target software-defined network to be compared against.

Services Assessment

In many cases, there are service dependencies associated with the starting network. For example, SAN or NAS in datacenters or Evolved Packet Core (EPC) functions in mobile networks. While not immediately tied to the network infrastructure, they are directly impacted by the network performance. These services should be identified along with test tools. A test assessment should be executed to provide a check point or benchmark for the target software-defined network to be compared against.

A. 2 Target Software-Defined Network

The target OpenFlow-based SDN network must be fully documented. Characteristics of the current or starting network scenario that are insufficient should be identified. Whether it is programmability and application interaction, serviceability or the ability to deploy more effectively, the need for heterogeneous equipment options, or simply provide a more maintainable network, the motivation for migration should be well understood.

The following categories should be outlined:

Objectives

A problem statement of the overall issues to be solved by the target software-defined network should be made (e.g.Opex or Capex savings, new service creation, scalability, etc.).

The desired outcome should document how the migration will address these problems. What characteristics of a software-defined network will address key issues identified in the problem statement?

SDN Architecture

The architecture of the target network deployment should be identified, including details of whether the target software-defined network will be based on OpenFlow, and if so which version, what interfaces and devices will be impacted, which physical architecture changes are required to manage the new traffic patterns, where the controllers will be deployed, etc. Finally, the nature of controller traffic must be clarified; the anticipated mix and relative volumes between provisioning, signaling and bearer traffic should dentify the specific processing and input/output requirements.

Migration Approach

The specific approach planned for the migration should be documented. Initially, the two approaches from the Charter should be considered. 1) upgrade existing equipment and swap out control machines for OpenFlow based SDN, or 2) deploy OpenFlow based SDN devices and control plane in parallel with existing equipment, then decommission existing equipment.

Dependencies

For each of the relevant requirements within the starting network definition, a traceability requirement should be documented, even if the purpose is to identify that the requirement cannot be met.

- **Redundancy Model**: There are several redundancy models that should be considered:
  - The target network must emulate a redundancy model.
  - The target network must participate in a redundancy model.
  - The target network must support an external redundancy model.
- **Management Tools**: Given the management tools and protocols used by the starting network, a specification of their viability in the migrated network should be documented and verified. If a continued use in the target network is expected, requirements that these tools place on the target software-defined network should be documented.
- **Network Capacity**: Care should be taken to ensure that the target software-defined network can support the intended traffic loads. Of some consideration, it should be verified that speeds and feeds can be met.
- **Upgrade Requirements**: The migration plan should spell out how the migration will impact live traffic, and whether that impact will comply with upgrade requirements.
- **Availability**: The migration plan should specify whether or not the target network will satisfy the existing availability requirements.

General Considerations

Migrations can be impacted by numerous external criteria. External influences may depend on the use case. This section should be used to identify these external considerations and address how to mitigate any associated risk to service continuity.

- Regulatory, high availability, security, traffic types (e.g. voice of IP/video)
- Things to think about/check list
- Controller impact on traffic (provisioning, signaling, application, firehose)
- Risk factors, Murphy's law, remedies, recommendations

Target Dependencies

This section should specify any dependencies from the equipment or software used by the target software-defined network. This may include specific requirements from individual vendors or simply provide general technology requirements.

Tool Requirements

Any additional tools required to sustain equivalent support within the target network should be specified. Issues unique to OpenFlow or the selected equipment should be called out. For example, visibility within the active ACL lists, where ageing of flows might start degrading performance as new and old flows compete for limited resources.

GAP Analysis

What is missing to get the migration completed? Are existing tools sufficient, or is there an opportunity for new tool development? Where gaps exist, contingencies should be specified.

Migration Procedures

This section should provide a step-by-step documentation for the migration. There may be iterative cases where the migration must be broken down into step functions. Issues such as acceptance criteria should also be iteratively documented in these cases.

Overall, the plan from starting network through target software-defined network must be captured within this section. In cases where the starting network isn't capable of migration, this section should further document the necessary steps to prepare the starting network for migration.

Along with the step-by-step procedures, there should be a set of checkpoints, as necessary.

Post-Migration Acceptance

As a complement to the Pre-Migration Assessment, equivalent, if not identical Acceptance procedures should be executed to verify whether or not the migration has been successful. Furthermore, as a troubleshooting mechanism, these Acceptance procedures should be referenced whenever problems occur within the network post migration.

Services Acceptance

As a complement to the Pre-Migration Services Assessment, equivalent, if not identical Acceptance procedures should be executed to verify whether or not the migration can successfully support the necessary services that are run over the network.

Migration Timeline

SDN migration timelines can vary considerably depending on a number of factors, including the the size, type and complexity of the starting network infrastructure, as well as the availability of tools and skill sets required. Thorough time planning should be undertaken and documented to ensure smooth implementation of the SDN migration taking into account possible roll-backs.

Skill Sets Requirements

Depending on the type of an OpenFlow- based SDN deployment (e.g. size, type and complexity of starting and target networks), the migration process is likely to require new skill sets which should be thoroughly investigated and documented.

A. 3 Options

Alternative solutions should be identified including OpenFlow-based SDN, alternative SDN and non SDN solutions. Specify which of the drivers is best met with each option. Reference larger strategies where applicable. For example, an OpenFlow SDN solution may not be significantly differentiated from a non SDN solution; however, an overall Network Function Virtualization (NFV) strategy may increase the utility of an SDN based solution.

A. 4 References

Comparable network deployments should be referenced, with specific attention given to areas related to the drivers. This section should further be used to capture any and all associated collateral with the migration.

Appendix B: Bibliography

[1] http://yuba.stanford.edu/foswiki/bin/view/OpenFlow/Deployment/Monitoring

[2] http://openflow.org/videos

[3] http://www.openflow.org/wk/index.php/Liboftrace

[4] http://www.openflow.org/wk/index.php/OpenFlow_Wireshark_Dissector

[5] http://mininet.org/

[6] http://www.openflow.org/wk/index.php/OFRewind

[7] https://bitbucket.org/peymank/hassel-public/

[8] http://eastzone.github.io/atpg/

[9] http://www.openflow.org/wp/deploy-openflow

[10] http://www.openflow.org/wk/index.php/Gates_deployment

[11] http://cseweb.ucsd.edu/~vahdat/papers/b4-sigcomm13.pdf

[12] Sushant Jain, Alok Kumar, SubhasreeMandal, JoonOng, Leon Poutievski, Arjun Singh, SubbaiahVenkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jonathan Zolla, UrsHölzle, Stephen Stuart and Amin Vahdat, “B4: Experience with a Globally-Deployed Software Defined WAN”, To appear in the proceedings of SIGCOMM’13, August 12–16, 2013, Hong Kong, China.

[13] https://www.opennetworking.org/images/stories/downloads/sdn-resources/customer-case-studies/cs-googlesdn.pdf

[14] R. Raszuk, J. Heitz, A. Lo, L. Zhang, X. Xu, “IETF RFC6769: Simple Virtual Aggregation (S-VA)”

[15] R. Raszuk, “onf2013.037.00: BGP Free Edge Migration WG Presentation”

Contributors

(In alphabetical order)

Salman Asadullah

Hakki C. Cankaya

Justin Dustzadeh

Ted Hardie

Bhumip Khasnabish

Mario Kind

Siegfried Luft

Mike McBride

Manuel Paul

Robert Raszuk

Evelyne Roch (Editor)

Srini Seetharaman

Mukhtiar Shaikh