

CORD Fabric, Overlay Virtualization, and Service Composition

Saurav Das

Open Networking Foundation

Ali Al-Shabibi, Jonathan Hart, Charles Chan and Flavio Castro

Open Networking Lab

Hyunsun Moon

SK Telecom

March 1, 2016

Introduction

This design note describes the implementation and design choices made for the CORD fabric, as well as the design for virtual networking and service composition. We also cover the interaction of the fabric with the vRouter, vOLT and multicast applications in CORD. Briefly, the CORD network architecture involves the following:

1. SDN based Leaf-Spine fabric built with bare-metal (OCP certified) hardware and open-source switch software. This is a pure OF/SDN solution—while we do use ARP, MAC, VLAN, IPv4, MPLS, VXLAN and other data plane headers, there is no usage of any of the distributed protocols found in traditional networking within the fabric. A non-exhaustive list of all the control-plane protocols we do not use ‘inside the fabric’ includes: STP, MSTP, PVSTP, RSTP, LACP, MLAG, OSPF, IS-IS, Trill, RSVP, LDP and BGP.
2. The fabric has the following characteristics:
 - a. L2 switching within a rack handled at leaf-switches (ToRs).

- b. L3 forwarding across racks using ECMP hashing and MPLS segment routing.
 - c. VLAN cross-connect feature to switch QinQ packets between OLT I/O blades and vSG containers.
 - d. IPv4 multicast forwarding and pruning for IPTV streams from upstream router to residential subscribers.
 - e. vRouter support for interfacing with upstream metro-router, providing reachability to publicly routable IPs.
 - f. Dual-homing of servers to Leaf-switches (ToRs).
 - g. Ability to use the fabric in single-rack or multi-rack configurations.
3. The fabric forms the underlay network in an overlay/underlay architecture. The overlay (sometimes referred to as the outer fabric) is also SDN based, with the following characteristics:
- a. Use of software-switches (eg. OvS with DPDK) with a custom-designed pipeline for service-chaining.
 - b. Distributed load-balancing per service in each OvS.
 - c. VxLAN tunneling in OvS for overlay-based virtual networks.

The biggest advantage of common SDN control over both the overlay infrastructure as well as the underlay fabric is that they can be orchestrated together to deliver the features and services that Central Offices require, with the agility and economies of datacenter operations.

ONOS Control Applications

Figure 1 shows the CORD architecture. The fabric at the heart of the implementation is responsible for interconnecting all parts of CORD, including access I/O blades, compute nodes (servers), and upstream routers. In addition, the fabric control application that runs on ONOS interacts with a number of other applications to provide CORD services.

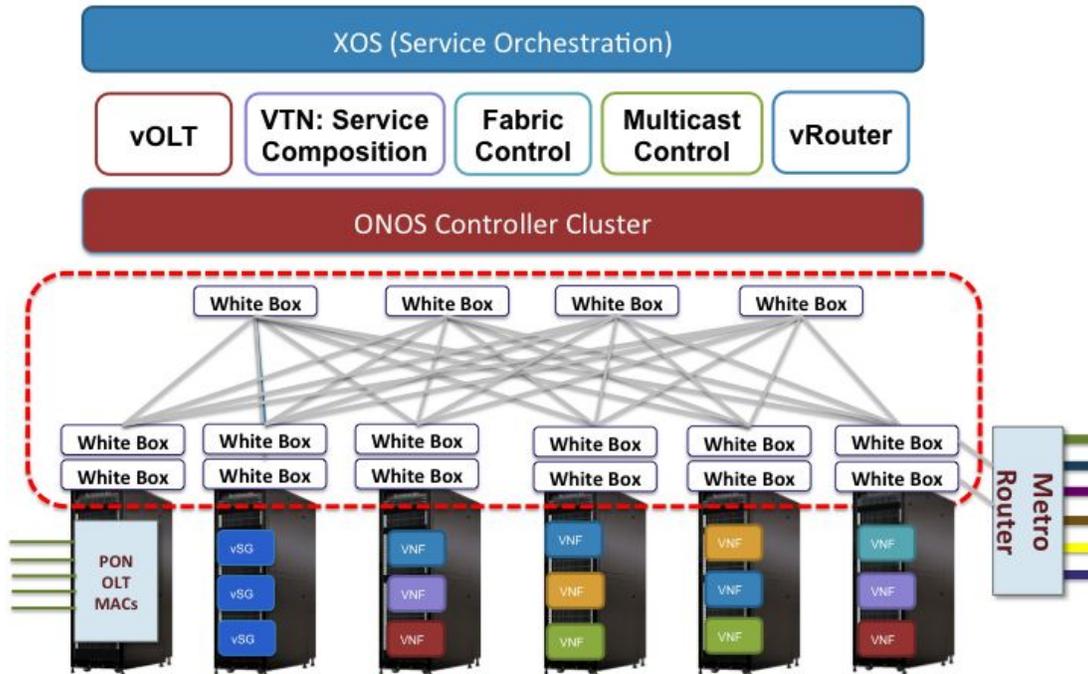


Figure 1. CORD Architecture.

In the current implementation, there are actually two sets of ONOS controllers with different responsibilities. The first ONOS cluster (*onos-cord*) is responsible for the overlay infrastructure (virtual networking and service composition) and the access infrastructure. This cluster hosts the VTN and vOLT applications, respectively.

The second ONOS cluster (*onos-fabric*) is responsible for controlling the fabric and interfacing with conventional upstream routers. This cluster hosts the Fabric Control and vRouter applications, respectively.

Multicast control is via two additional applications, IGMP snooping and PIM-SSM, where the former runs on *onos-cord* and the latter runs on *onos-fabric*. (For simplicity, we show only a single Multicast Control application in Figure 1.)

In principle, all the applications could run on a single ONOS cluster. We chose to split responsibilities to have better isolation and separation of concerns, which was especially helpful during development. However, to simplify the exposition and diagrams, we show all the applications running on a single ONOS cluster in Figure 1.

Fabric Hardware & Software

The switches in the fabric leverage software and hardware from a number of other open source projects, as shown in Figure 2.

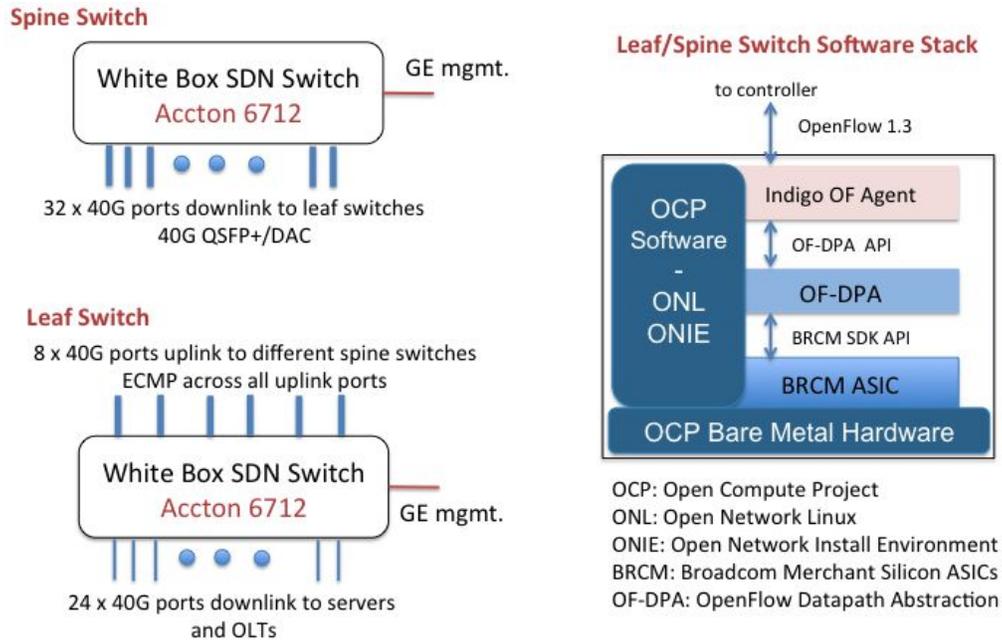


Figure 2. Fabric Software and Hardware.

The leaf and spine switches are identical Accton 6712 (OCP-certified) switches with 32 40G ports. The only difference is in their usage. The leaf switches use 24 of the 32 ports to connect servers, access blades and metro equipment, reserving 8 of the ports to connect to spine switches. The spine switches use all of their ports to connect to the leaves. There is nothing sacred about this arrangement. In smaller deployments, more leaf ports can be used to connect servers, and in the extreme case, the spines can be eliminated altogether. Similarly, other deployments can use different switch models with 10G or 1G ports (eg. Accton 5712 or 5710 switches). The only requirement is the support of the switch software stack, which we describe next.

A key aspect of this work is that both leaf and spine switches run exactly the same open source switch software stack. This stack was first proposed and integrated in the ONF Atrium project [1], and includes ONL and ONIE as the switch operating system and boot loader. It also includes Broadcom's OF-DPA software [2], which opens up a number of key APIs from their proprietary SDK API. OF-DPA presents an abstraction of the switch ASIC pipeline (forwarding tables and port-groups) in OpenFlow terms, so that an OF 1.3 agent like Indigo can be layered on top of the API. With this layering, an external controller (like ONOS) can then program all the forwarding tables in the switch ASIC thereby leveraging the full capabilities of today's modern ASICs. Figure 3 shows a simplified view of the OF-DPA pipeline.

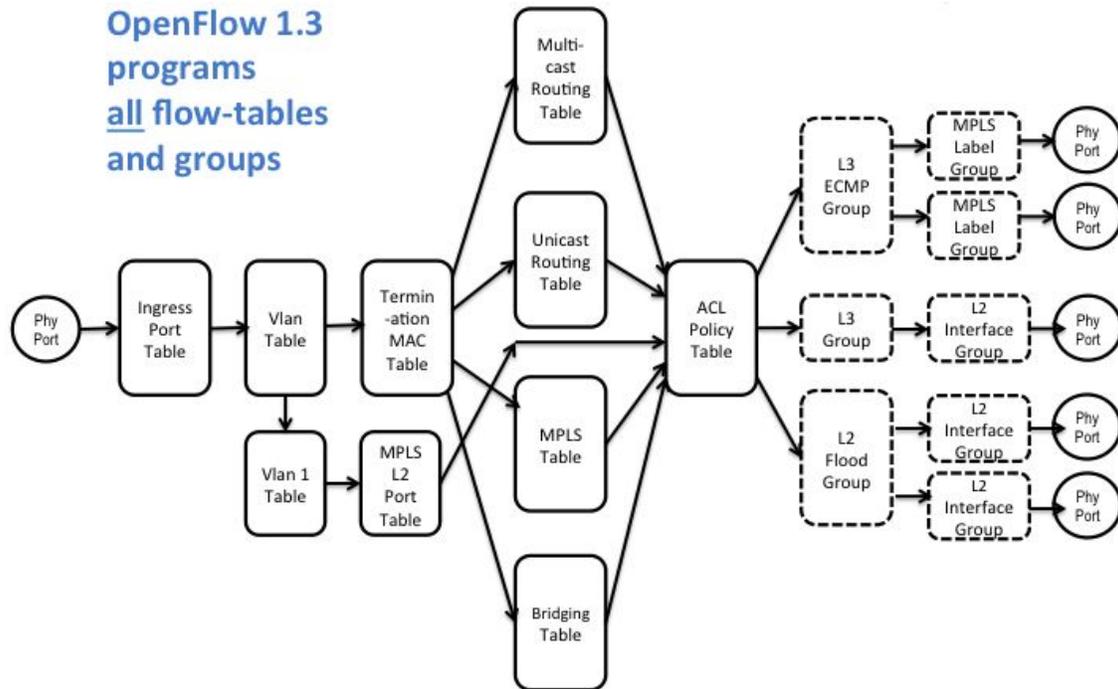


Figure 3. Fabric Chip Pipeline (Broadcom's OF-DPA).

Fabric Support for OLT—vSG Communication

Residential subscribers connect to CORD via a GPON network. The head-end node of the GPON network is called an OLT, which traditionally is chassis based monolithic hardware available from several vendors. In CORD, we disaggregate the OLT, by keeping only the OLT physical layer in dedicated hardware (called OLT-MACs), and moving all other functions into software distributed over the CORD cloud (see Figure 1). One such function is a Subscriber Gateway, which is implemented by a container [3].

Traffic between the OLT blades and the vSG containers is double-tagged, with the outer-VLAN-tag identifying the PON network a subscriber belongs to, and the inner-VLAN tag identifying the subscriber itself.

This residential access infrastructure (Figure 4) is handled by the vOLT control application, in concert with XOS; details can be found in other design notes [4]. Once the customer has been identified and authenticated, the double-VLAN tags have been assigned for the customer, and a vSG container has been instantiated in a compute-node, the fabric-control application is informed of the location of the OLT blade the subscriber is connected to, and the compute node where the vSG container has been instantiated. The app then programs forwarding rules in the fabric to enable this communication through the fabric.

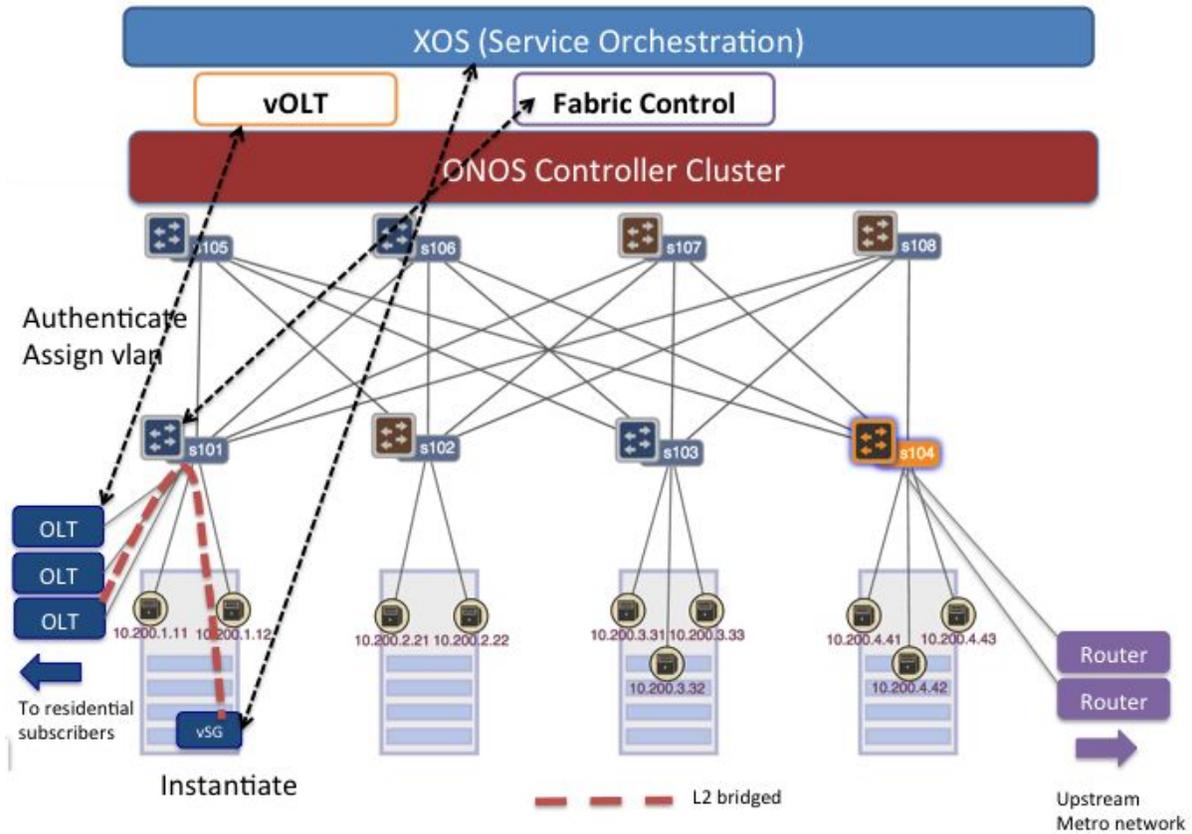


Figure 4. OLT to vSG Communication.

Typically, the OLT blade and vSG containers are physically located in the same rack and therefore connected to the same Top-of-Rack leaf switch. In this case, the communication is enabled in the fabric via a VLAN cross-connect. The cross-connect is a special case of L2 bridging where the forwarding is based on just the VLAN headers, while ignoring the destination MAC addresses, of which there could be many from the same subscriber (different home devices). The fabric does not need to learn or have any a priori knowledge of these MAC addresses. It simply needs to connect the VLANs identifying the subscriber to the physical port on which the compute-node hosting the vSG for that subscriber is located. As a result, a particular vln-crossconnect is always implemented between exactly two physical ports on the same leaf-switch. In future releases, we will add the ability to physically locate the vSG containers anywhere in the CORD cloud (instead of the same rack as the OLT blades).

Service Composition

Services in CORD are composed using the best practices of Cloud operations. In CORD, services are instantiated by the network operator using XOS (Figure 5). In turn XOS presents ONOS with a service graph for subscriber traffic. This graph is then decomposed into flow-rules that are programmed in the CORD networking infrastructure

by the VTN application in ONOS. Details of XOS and service graphs are discussed in other design notes. This section gives a brief overview of the implementation choices made for realizing service composition in the network infrastructure.

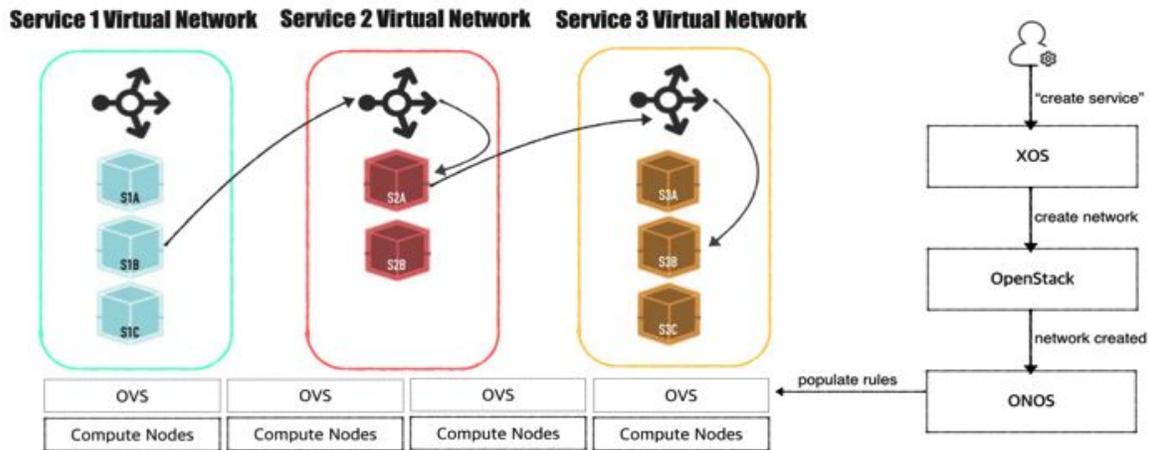


Figure 5. Service Composition.

In CORD, service composition is implemented using overlay-networks and network virtualization. At a high level,

- Services have their own virtual networks (VNs)—the VMs or containers that instantiate the service are part of the same virtual network, and these instances can be created in any part of the CORD cloud (ie. on any compute-node in any rack).
- It is possible to dynamically grow and shrink the number of VMs/containers (and hence the virtual network) that instantiate a service. This feature is essential for achieving scale in the cloud.
- Each compute node hosts VMs or containers (belonging to multiple service VNs) that are connected to OVS acting as a highly programmable OpenFlow controlled hypervisor switch.
- Each Virtual Network (or service) has its own Load-Balancer distributed across every OVS in the network. The load-balancers job is to select a VM instance instantiating the service, amongst all the VM's within the service's virtual network.
- Service composition walkthrough: Let's say a VM S1B, which is an instance in Service 1 VN, figures out that the next service some traffic needs to go to is Service 2. Instead of sending it directly to a VM in service 2, it sends the traffic to the load-balancer for Service 2, which then selects the VM (say S2C). It is also

possible that the operator wishes that traffic goes directly from S2C specifically to VM S3A, instead of the load-balancer for Service 3.

- XOS and ONOS's VTN app coordinate to keep the virtual infrastructure state updated. As VMs are added/deleted for a service, and service chains are created, VTN updates tables in a special purpose OVS pipeline to reflect the desired subscriber service flow. A functional block-diagram of the OVS forwarding pipeline is shown in Figure 6.
- Subscriber traffic is NATted by the vSG. After the vSG, subscriber traffic can directly head out to the Internet, or proceed through a series of services and then head out to the Internet. To transport subscriber traffic between services, we use VxLAN encapsulation. VxLAN tunnels are used to interconnect VMs and services. The encapsulated traffic is routed or bridged by the underlay fabric (which we describe next) using infrastructure IP/MAC address belonging to the fabric and the compute nodes. Finally, when routing out to the Internet and back, the fabric routes non-encapsulated NATted IP packets, with help from BGP routes learned by the vRouter application.

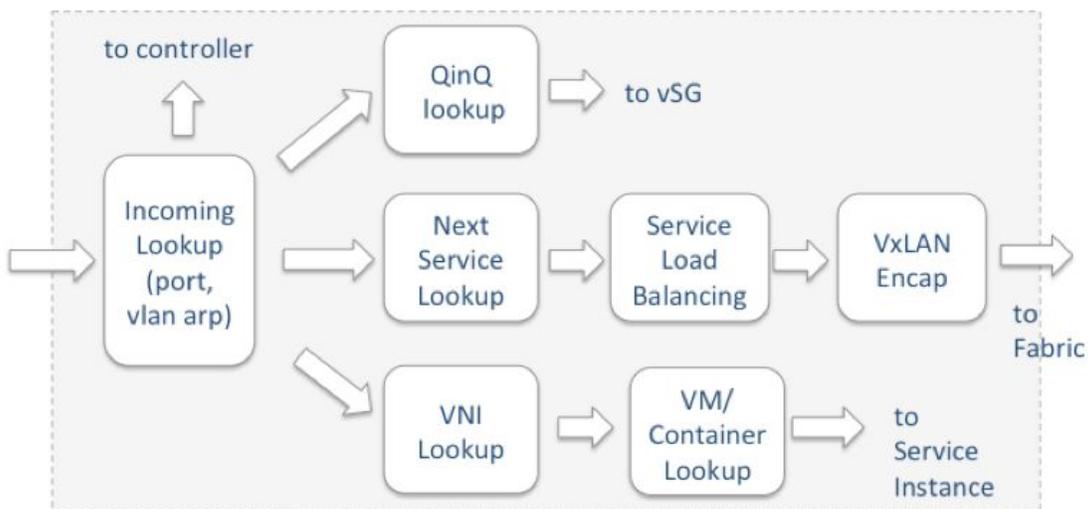


Figure 6. CORD OvS Pipeline.

Fabric Operation as an Underlay

For transporting VxLAN encapsulated traffic between services in the overlay network, the fabric acts as an underlay. The fabric control application's job is to effectively utilize the cross-sectional bandwidth provided by the leaf-spine architecture. For intra-rack traffic between servers in the same rack, the fabric does regular L2 bridging. For inter-rack traffic between servers in different racks, the fabric does L3 routing (with MPLS). In other words, the underlay fabric behaves like an IP/MPLS network that routes

traffic between attached subnets, where each rack has its own subnet. Traffic within a subnet (and therefore within a rack) is L2 bridged.

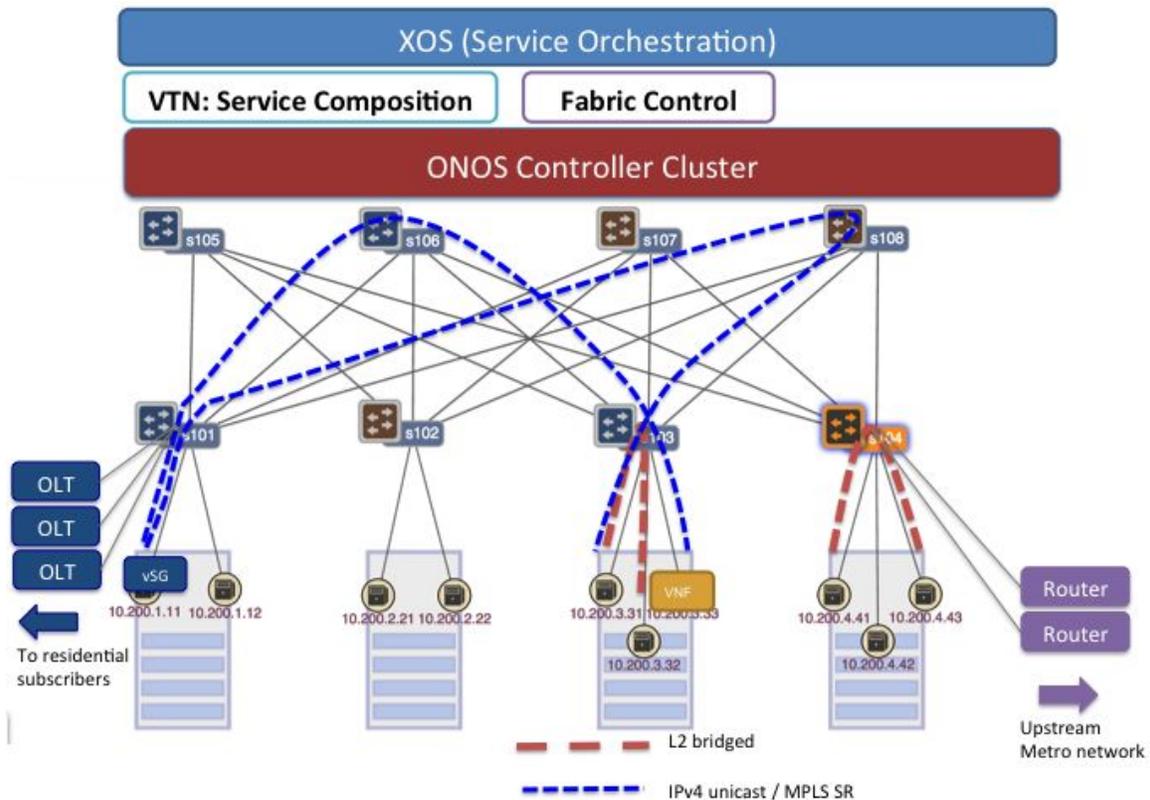


Figure 7. Fabric as an underlay.

With reference to Figure 7, each rack has its own subnet 10.200.x.0/24. Servers in the same rack are assigned (infrastructure) IP addresses in the same subnet (for example 10.200.1.11 and 1.12 in rack 1). Traffic between these servers is bridged by the ToR (leaf switch) they are connected to (s101 in Figure 7). Traffic destined to another rack gets routed by the same ToR. Here we use concepts from MPLS Segment Routing, where we attach an MPLS label to designate the ToR switch in the destination rack, and then hash the flows up to multiple spines. The spines do only MPLS label lookups to route the traffic to the destination ToR. A packet walk-through is shown in Figure 8.

The fabric design choices come out of a desire to have forwarding separation between edge and core (in this case the leaves and spines). Ideally, for a multi-purpose fabric, it is best if the core remains simple and ‘same’ for all use-cases, and only the input to the core changes on a per-use case basis. In our case, this means that the core forwards traffic only on the basis of MPLS labels, whereas the input at the leaf switch can comprise of IPv4, or IPv6, or multicast, or MAC, or VLAN, or VLAN+MAC or QinQ or even other MPLS labels.

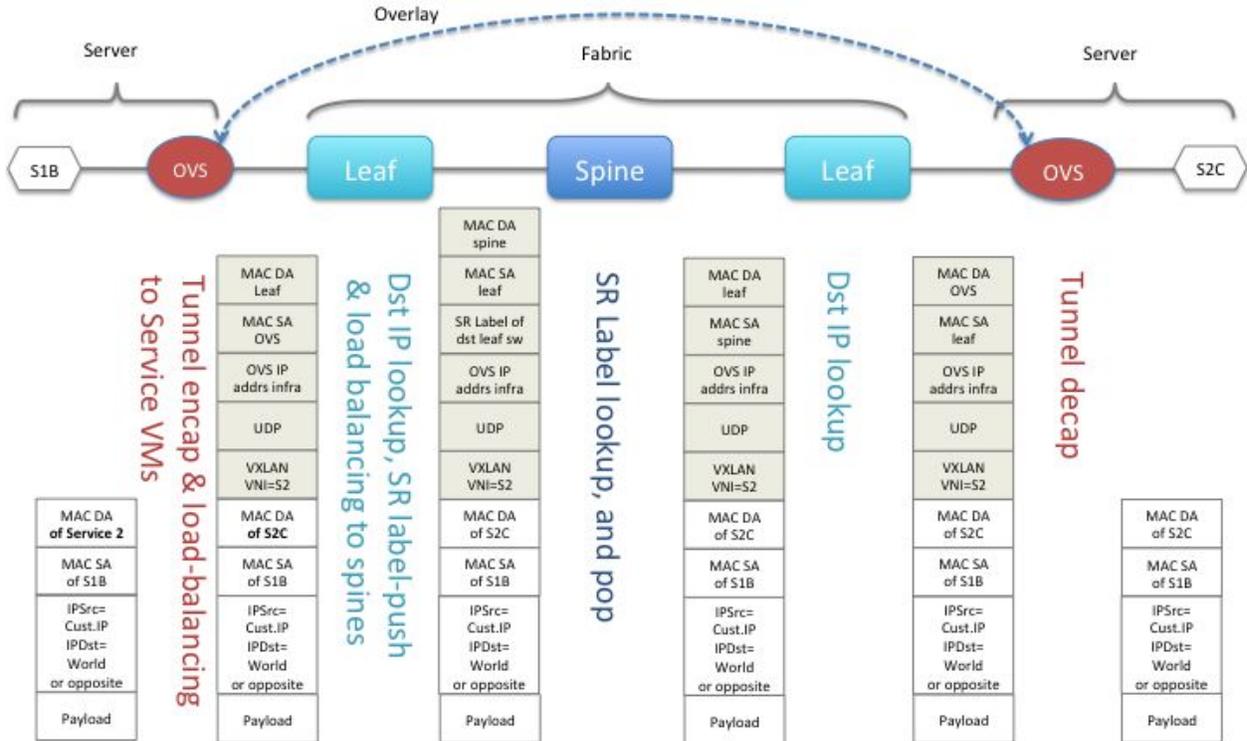


Figure 8. Overlay and Underlay packet walk-through.

The fact that we use MPLS labels to achieve this edge-core (leaf-spine) separation is immaterial. We could have just as well used VLAN tags. However in our experience with current ASICs, it is easier to use labels instead of tags, as they are treated differently in current ASIC pipelines and there are different rules to follow and associated limitations in usage.

Furthermore, Segment Routing is just a convenient way to use MPLS labels. SR gives us the concept of globally significant labels which we assign to each leaf and spine switch. This leads to less label state in the network, compared to traditional MPLS networks where locally-significant labels have to be swapped at each node. In addition, by default, SR requires the use of ECMP shortest paths, which fits well with the leaf-spine fabric, where leaf switches ECMP hash traffic to all the spine switches. Finally, SR is source-routing, where we can change the path traffic takes through the network, by simply changing the label assignment at the source (leaf) switch - there is only one switch to 'touch' instead of the entire 'path' of switches. This is how we have performed elephant-flow traffic-engineering in the past using external analytics [REF].

In addition, an overlay/underlay architecture simplifies network design, as it introduces a separation of concerns - the underlay hardware fabric can be simple and multi-purpose without knowing anything about virtual networks, while the overlay network can be

complex but implemented in software switches where more sophisticated functionality like service-chaining and distributed load-balancing can be achieved. Importantly, this architecture does not mean that we cannot have the overlay and underlay network interact with each other and gain visibility into each other. This is especially true in an SDN network where we have the same controller-based control plane for both overlays and underlays.

Fabric and Multicast

Coming soon...

Fabric and vRouter

Coming soon...

References

[1] Atrium.

https://www.opennetworking.org/images/stories/news-and-events/sdn-solutions-showcase/Atrium_Overview.pdf

[2] OpenFlow Data Plane Abstraction (OF-DPA).

<https://www.broadcom.com/collateral/pb/OF-DPA-PB100-R.pdf>

[3] Virtual Subscriber Gateway (vSG). *CORD Design Notes* (March 2016).

[4] Virtual OLT (vOLT). *CORD Design Notes* (March 2016).