Received: 2 November 2018 | Revised: 3 May 2019 | Accepted: 13 July 2019

DOI: 10.1002/dac.4128

RESEARCH ARTICLE

WILEY

Circuit-based logical layer 2 bridging in software-defined data center networking

Yao-Chun Wang | **Ying-Dar Lin**

Computer Science, National Chiao Tung University, Hsinchu, Taiwan

Correspondence
Yao-Chun Wang, Computer Science, National Chiao Tung University, Hsinchu, Taiwan.
Email: scott0612@cht.com.tw

Summary

With the expansion of the size of data centers, software-defined networking (SDN) is becoming a trend for simplifying the data center network management with central and flexible flow control. To achieve L2 abstractions in a multitenant cloud, Open vSwitch (OVS) is commonly used to build overlay tunnels (eg, Virtual eXtensible Local Area Network [VXLAN]) on top of existing underlying networks. However, the poor VXLAN performance of OVS is of huge concern. Instead of solving the performance issues of OVS, in this paper, we proposed a circuit-based logical layer 2 bridging mechanism (CBL2), which builds label-switched circuits and performs data-plane multicasting in a software-defined leaf-spine fabric to achieve scalable L2 without overlay tunneling. Our evaluations indicate that direct transmission in OVS improves throughput performance by 58% compared with VXLAN tunneling, and data-plane multicasting for ARP reduces address resolution latency from 149 to 0.5 ms, compared with control-plane broadcast forwarding. The evaluation results also show that CBL2 provides 0.6, 0.4, and 11-ms protection switching time, respectively, in the presence of switch failure, link failure, and port shutdown in practical deployment.

KEYWORDS

cloud, datacenter, layer 2, multitenancy, network virtualization, OpenFlow, SDN

1 | INTRODUCTION

Infrastructure-as-a-Service (IaaS)[1] providers enable enterprise customers (who are also called tenants) to obtain flexible and on-demand virtualized infrastructures, by using virtualization technologies that share the computing resources (eg, servers, storages, and networks) in a data center. In a multitenant cloud, customers build their own virtual data center (VDC) with multiple virtual private clouds (VPCs),[2,3] to provide a variety of services, which can be accessed via internet or VPN. The data of each customer are securely isolated from other customers. Figure 1 shows the layout of multitenant cloud. For building a web service, a VPC network may be configured to a typical three-tier client-server architecture with a virtual router and three logical L2 subnets for virtual machines (VMs) in the three tiers: Web, App, and Database. A logical L2 network defines the broadcast domain boundary and provides interconnectivity between the access points (ie, switch ports with designated VLAN IDs) under this broadcast domain.

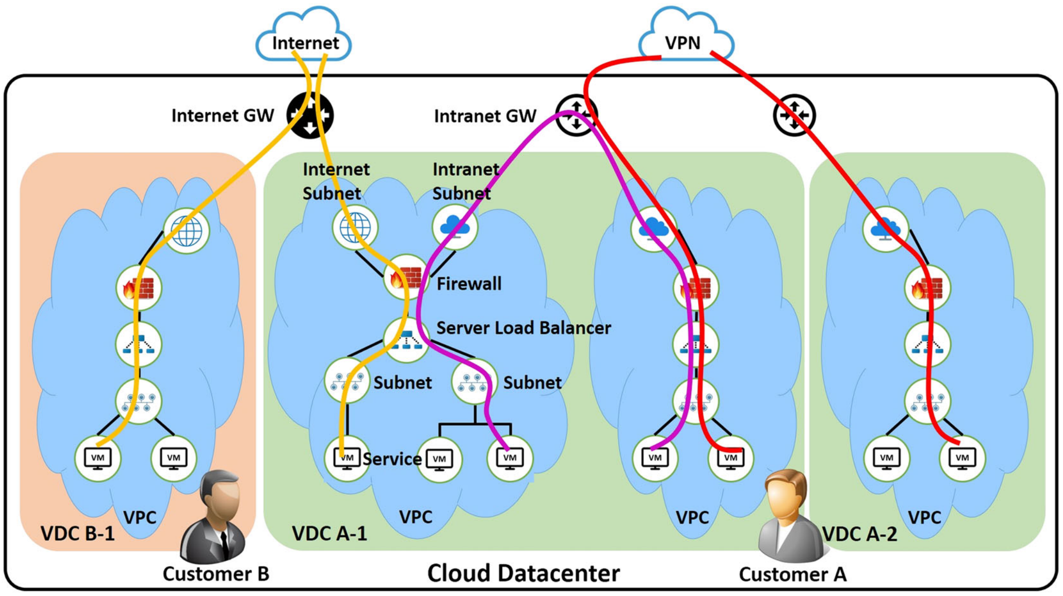

FIGURE 1 The layout of multitenant cloud

1.1 | Software-defined data center networking

With the large and rapid-growing traffic transfers in the data center network, the traditional three-tier network design (core, aggregation, and access layers) is being replaced with the leaf-spine design[4,5] in modern data centers. The advantages of leaf-spine architecture include the ability to provide predictable latency for east-west traffic flow, to improve the utilization of link resources, and to be easily scaled out for the demands of bandwidth and access port number. And with the expansion in the size of a data center, software-defined networking (SDN)[6] is added to simplify network management with central control ability. SDN also creates opportunities for traffic optimization in a data center by enabling flexible and dynamic flow control.[7-10]

1.2 | Virtual L2 overlays

To achieve L2 abstractions over a leaf-spine fabric for the access points located on different leaf switches, VLAN and Virtual eXtensible Local Area Network (VXLAN) protocols[11] are widely used in a multitenant data center. VXLAN is an encapsulation protocol advocated by Internet engineering task force (IETF) for building overlay networks on top of existing underlying L2/L3 network. Compared with VLAN, VXLAN resolves the issue of scalability with 24-bits VNI (VXLAN Network Identifier) in VXLAN header, supporting more than 16 million L2 divisions.

In most deployment cases, Open vSwitch (OVS)[12] is commonly used as a VXLAN gateway, which is also called VXLAN Tunnel End Point (VTEP), for all VMs in a physical server, which connect to a leaf switch. The uplink packets coming from VMs will be encapsulated by OVS and then be sent to a leaf switch. Similarly, the downlink packets coming from a leaf switch will be decapsulated and are then sent to VMs. Using OVS to build overlay tunnels is a quick and convenient way to achieve logical L2 networks for tenants, by keeping the existing leaf-spine unchanged. However, the poor VXLAN performance of OVS caused by software implementation[13] is of huge concern. Our evaluations indicate that direct transmission in OVS improves throughput performance by 58% compared with VXLAN tunneling. Although there are some researches or products that focus on hardware acceleration methods,[13,14] the deployment cost of a data center should also be taken into consideration.

1.3 | Overlay tunneling versus underlay label switching

In the light of this, instead of solving the performance issues of OVS, we propose a circuit-based logical layer 2 bridging mechanism (CBL2) as an alternative to enhance the transmission performance in logical L2 networks. The basic idea behind CBL2 is to replace the overlay tunneling enabled by OVSs in servers with underlay label switching performed by a software-defined leaf-spine fabric. With this approach, we designed forwarding entries for OpenFlow 1.3-enabled[15] spine and leaf switches to provide virtual L2 forwarding. Consequently, there is no need for OVS to carry out VXLAN encapsulation. Table 1 gives a comparison between overlay tunneling and underlay label switching.

CBL2 specifically provides the following features:

1.3.1 | Scalable logical L2 with reusable VLAN

To recognize the members (ie, access points) of a logical L2 network across different leaf switches while providing scalability, we consider the combination of a VLAN ID and a leaf switch port (or two leaf switch ports for high availability [HA]) as an identifier for an access point, by matching packets come from servers against both ingress port and VLAN ID at leaf switches, with OpenFlow 1.3. In other words, 4096 VLAN IDs are supported by each leaf port instead of the whole infrastructure (ie, per-port VLAN), which means the access points with the same VLAN ID but on different leaf ports can belong to different logical L2 networks.

In order to connect the access points located on different leaf switches for each logical L2 network, CBL2 establishes a fully meshed logical network that consists of the leaf switches, where the access points located on, and the circuits between every pair of these leaf switches. To carry a great number of circuits on each directed link between spine and leaf, we build circuits with label (eg, VLAN tags and MPLS labels) switching.

1.3.2 | Low-controller-overhead logical L2 broadcasting

As noted above, the data of each customer must be securely isolated from other customers. To provide isolated logical L2 networks, the broadcast traffic in an L2 network, such as Dynamic Host Configuration Protocol (DHCP) discover and Address Resolution Protocol (ARP) request, should be limited to specified access points.

An intuitive way[16,17] to handle ARP in an SDN environment with separate control and data planes is to intercept the ARP requests at switches and forward them to the controller via the preinstalled flow entries. The controller then redirects the request packet to the requested VM or replies the requesting VM with the requested destination MAC address directly, working as an ARP proxy. However, with control-plane forwarding, both controller overheads and response delays increase as the VM number increases.

For the purpose of dealing with broadcast in data plane, CBL2 handles the broadcast traffic by adopting data-plane multicasting, so as to multicast the broadcast packet sent from an access point to all the other access points within the same broadcast domain, via the leaf switch and the established circuits. As for unicast traffic, CBL2 forwards packet based on its ingress port, label, and destination MAC address, instead of the destination MAC only, to ensure that each L2 access is limited to a logical L2 network.

1.3.3 | Dynamic logical L2 expansion

We rely on OpenFlow 1.3-enabled leaf-spine fabric to provide virtual L2 forwarding. All the spine and leaf switches are controlled by our CBL2 application on an OpenFlow controller. OpenFlow is a popular SDN standard, an OpenFlow 1.3-enabled switch consists of one or more flow tables and a group table, which contains flow entries and group entries to perform packet lookups and forwarding. The flow tables are sequentially numbered by table IDs, starting at 0. Pipeline processing always starts at the first flow table, and other flow tables may be used depending on the outcome of the match in the first table. Since flow tables can be linked in sequence to provide pipeline processing, we designed table pipelines, which have a clear division of labor in tables and highly shared forwarding configurations (ie, group entries),

TABLE 1 Overlay tunneling and label switching

	Device Support	Key Component	Performance
Overlay tunneling	Software OVS	VXLAN	Low
Label switching	Hardware leaf-spine fabric	SDN	High

Abbreviation: OVS, Open vSwitch.

to support dynamic expansion of logical L2 networks with low-network update overhead. The formulas for estimating the consumption of flow entries will also be covered in this paper.

1.3.4 | High availability

To be able to ensure the resilience of a data center is a critical issue.[18,19] CBL2 provides HA in the presence of switch and link failures, by deploying redundant spine and leaf in the fabric. Our table pipeline design makes use of the fast failover feature of OpenFlow group entries to enable path protection in spine and leaf switches.

The contribution of this paper is summarized below. First, we proposed a scalable circuit-based L2 bridging mechanism that achieves high-performance L2 abstractions in a multitenant cloud over a software-defined leaf-spine fabric without overlay tunneling. Second, we proposed an agile OpenFlow table pipeline design that offers dynamic expansion of logical L2 broadcast domain and supports HA deployment.

In our evaluation, we compared the throughput performance with OVS between cases with and without VXLAN tunneling and compared the average address resolution time for ARP requests in a leaf-spine fabric, between broadcast handling by control plane and data plane. For HA of CBL2 deployment, we measured the protection switching time in the presence of switch and link failures.

The remainder of this paper is organized as follows: Section 2 discusses related works; Section 3 describes the problem statement; Section 4 illustrates solution details; Section 5 gives an analysis of flow entries consumption; Section 6 shows the experimental results and related observation; and Section 7 contains the conclusions.

2 | RELATED WORK

In this paper, we focus on two issues relevant to virtual L2 forwarding: how to provide end-to-end transparent connectivity and how to handle broadcast traffic. There are some pure SDN/OpenFlow solutions[16,17] relying on VXLAN tunneling in OVS and handling broadcast with controller. Wide-area Layer-2 (WL2)[16] is an SDN solution to an internet-scale L2 network across multiple data centers. WL2 creates full-mesh virtual overlay networks among local servers in a data center by establishing a logical Ethernet link between every pair of virtual switches (in servers) through L2 tunneling (eg, VXLAN and GRE). L2 broadcast-based control traffic is intercepted by the virtual switches and forwarded to the controller, to avoid broadcast. Central Office Re-architected as a Data center (CORD)[17] network infrastructure adopts a combination of an underlay leaf-spine fabric, overlay virtual networking, and unified SDN control over both underlay and overlay. In underlay fabric, L2 switching within a rack is handled at leaf-switches (ToRs), and L3 forwarding across racks relies on ECMP hashing and MPLS segment routing. All ARP packets are sent to the controller and to be handled with proxy ARP. For overlay, CORD builds virtual networks for services with VXLAN tunneling in software switches (eg, OVS with DPDK[14]). As mentioned above, VXLAN tunneling in OVS usually needs hardware acceleration to remedy performance problems, and control-plane broadcast handling may increase controller overheads and response delays.

There are also some commercial products[20,21] with confidential implementation details that provide virtual L2 connectivity through physical leaf-spine fabric to support a variety of software switches in servers and handle broadcast in data plane without controller. The Cisco Application Centric Infrastructure (ACI)[20] managed by Cisco Application Policy Infrastructure Controller (Cisco APIC)[22] abstracts policy and connectivity from the underlying fundamental network by using an extended VXLAN encapsulation frame format in leaf-spine fabric. In ACI, tenants can be divided into Layer 3 constructs, known as Virtual Routing and Forwarding (VRF) instances. Within each VRF instance, a bridge domain is created to provide L2 connectivity between endpoints (eg, physical or virtual network interface cards and their associated VLANs). The leaf nodes act as the VTEPs, and all tenant traffic in the fabric is tagged at the first-hop leaf ingress port with an extended VXLAN header, which carries the VNI to identify bridge domains for L2 traffic. However, in current implementation of ACI, the VLAN namespaces cannot overlap for a given leaf node because of the local significance of VLAN ID for the leaf node.

Big Cloud Fabric (BCF)[21] is a scale-out SDN fabric with the ability to support multitenancy. BCF architecture consists of a physical leaf-spine switching fabric, and the fabric can be extended to virtual switches residing in hypervisor optionally. A tenant in BCF is a logical grouping of L2 and/or L3 networks and services. A logical segment of a BCF tenant represents an L2 network consisting of endpoints, and the endpoints on the same VLAN can communicate with each other. In this system, the L2 forwarding in physical leaf-spine fabric is implemented without fabric encapsulation.

Moreover, the broadcast traffic is handled by data plane, to support headless mode (ie, fabric forwards traffic in the absence of controller) for high availability. Nevertheless, according to the BCF datasheet, BCF supports 4 K VLANs at most (ie, 4 K logical segments), which means that BCF is not scalable for VLAN tenants in a large-scale cloud environment.

Table 2 gives a summary of related works. Compared with the southbound protocol used by Cisco APIC and ACI, pure OpenFlow design can avoid vendor lock-in for network expansion. Considering to the pros and cons of each method summarized above, we designed CBL2, which provides scalable L2 with per-port VLAN through physical OpenFlow leaf-spine fabric and handles broadcast without the controller.

3 | PROBLEM STATEMENT

The objective of CBL2 is to provide virtual L2 forwarding over a leaf-spine fabric. More precisely, given an OpenFlow 1.3-enabled leaf-spine fabric with redundant leaf and spine, a group of access points (an access point includes a VLAN ID and a redundant pair of physical ports on redundant leaves) and the MAC addresses attached to each access point, we designed table pipelines in leaf and spine switches to bridge these access points and MAC address, with the following aims: (a) to provide scalable L2 with per-port VLAN; (b) to support dynamic expansion of both access points and MACs without service interruption; (c) to provide HA in the presence of switch and link failures to avoid Single Point of Failure (SPOF); and (d) to keep data plane traffic unaffected in the absence of controller.

4 | SYSTEM DESIGN

CBL2 provides on-demand virtual L2 provisioning services through northbound APIs. CBL2 is implemented as a controller application, controlling an OpenFlow 1.3-enabled leaf-spine fabric. To achieve scalable virtual L2 without VXLAN tunneling in OVS, CBL2 provides per-port VLAN, which allows VLAN reuse for multiple different logical L2 networks and builds circuits from leaf to leaf with label switching to bridge the access points of a logical L2 network. CBL2 also uses data-plane multicasting to achieve broadcasting in a logical L2 network. Since there is no need to send any L2 packets to a controller, the fabric can forward traffic in the absence of controller.

In this section, we first introduce the HA network architecture and then elaborate the OpenFlow table pipeline design for circuit creation, broadcast forwarding, and unicast forwarding. Finally, we provide the steps required to update a logical L2 network.

4.1 | Network architecture

Figure 2 shows the leaf-spine fabric controlled by CBL2 to provide HA logical L2 networks. Each server connects to a leaf group, which consists of a redundant pair of leaf switches, with port bonding configuration (eg, bonding two network interfaces into a logical interface) on virtual switch in the server. In each leaf group, there is an interleaf link

TABLE 2 Related works

	Southbound Protocol	Virtual L2 Connectivity	Deployment	Broadcast Handling	Proprietary
Cisco ACI Fabric[20]	OpFlex	Overlay (extended VXLAN)	Spine-leaf	Data plane	Yes
Big Cloud Fabric (BCF)[21]	Vendor-extended OpenFlow	Underlay	Spine-leaf (+ server OVS)	Data plane	Yes
CORD[17]	OpenFlow	Underlay (segment routing), Overlay (VXLAN in OVS)	Spine-leaf + server OVS	Control plane, higher delays	No
WL2[16]	OpenFlow	Overlay (VXLAN in OVS)	Server OVS	Control plane, higher delays	No
CBL2	OpenFlow	Underlay (label switching)	Spine-leaf	Data plane	No

ABBREVIATIONS: ACI, Application Centric Infrastructure; CBL2, circuit-based logical layer 2; CORD, Central Office Re-architected as a Data center; OVS, Open vSwitch; VXLAN, Virtual eXtensible Local Area Network.

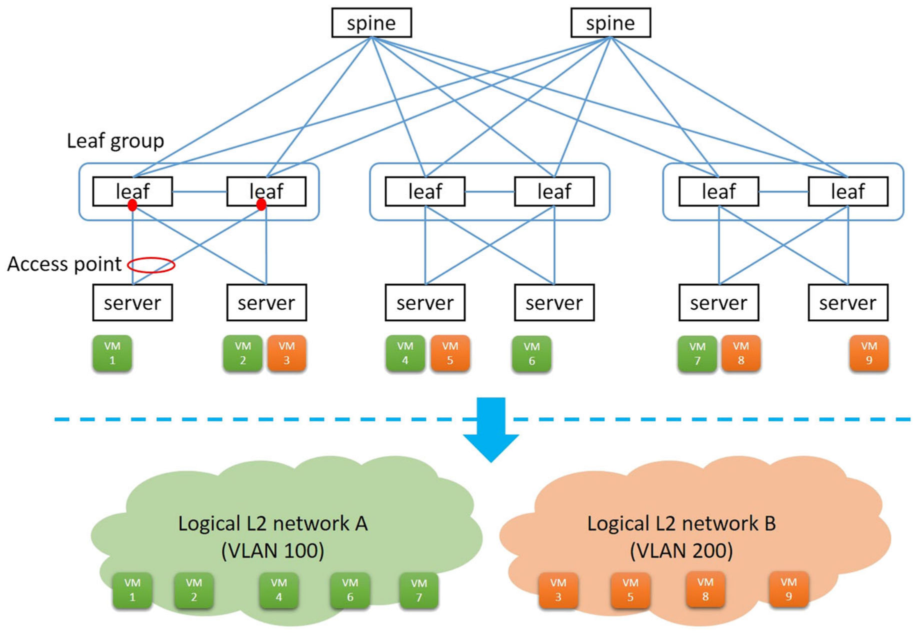

FIGURE 2 The leaf-spine fabric with redundant leaf and spine

between two leaves used for traffic migration between leaves in the presence of link failure between leaf and spine or between leaf and server.

Since there are many VMs in a server and these VMs may belong to different L2 networks with different VLAN IDs, each VLAN coming out of a bonded interface of a server represents an access point attached to a logical L2 network.

4.2 | Circuit creation

To bridge the given access points belonging to the same logical L2 network, CBL2 first establishes bidirectional circuits between every pair of leaf groups, which contain the access points, as shown in Figure 3. Each circuit from one source leaf group to another target leaf group is responsible for delivering the VLAN traffic received by the access points attached to the source leaf group, to one of switches in the target leaf group, via a spine.

To carry a very large number of circuits on each directed link between spine and leaf, we built circuits with MPLS label switching, which means that the packet will be forwarded along the circuit based on its assigned label, and the label may be swapped on the way so as to avoid label conflict between circuits.

The ability for a flow entry to point to a group entry enables OpenFlow to represent additional forwarding methods, which are determined by the group type (eg, all, select, indirect, and fast failover) of the group entry.

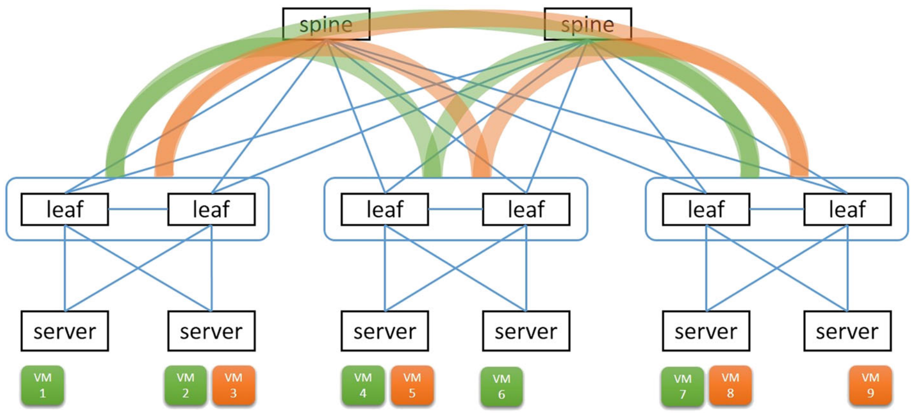

FIGURE 3 Bidirectional circuits between every pair of leaf groups

To achieve path protection for each circuit, at each leaf of the source leaf group, CBL2 installs a select-type group entry "*Path_Group*" for pushing MPLS tag and sending packets to one of the spines (based on a switch-computed selection algorithm). At each spine, CBL2 installs flow entries to the flow table "*Path_Table*" for matching packets from leaves, based on packet's switch input port (*in_port*) and MPLS tag. It then installs a select-type group entry "*DST_Leaf_Group*," which is pointed by flow entries in the "*Path_Table*" to swapping MPLS tag as needed and sending packets to one of the leaves in the target leaf group. Figure 4 shows the table pipeline design for circuits at spines.

In the end, CBL2 establishes a fully meshed logical network that consists of the leaf switches, where the access points located on and the circuits between every pair of these leaf switches.

4.3 | Broadcast forwarding

Instead of using control-plane broadcast forwarding (eg, the controller redirects the request or replies to the request directly), CBL2 uses data-plane multicasting to achieve broadcasting in a logical L2 network. This enables the fabric to forward traffic in the absence of controller and reduces both controllers overhead and response delay.

Figure 5 shows the table pipeline design for broadcast forwarding at leaves. When a broadcast packet comes in on any port of a leaf, it will be identified by the flow table "*DST_Mac_Table*" and will be directed to the "*Multicast_Table*" to determine the direction of packet and taking appropriate actions based on the packet's *in_port* and VLAN/MPLS tag. Figure 6A shows the behavior of broadcast forwarding in CBL2. The forwarding options for packets from different directions are detailed below.

4.3.1 | Uplink

For a broadcast packet coming from an access point with a specified *in_port* and VLAN tag, an all-type group entry "*Multicast_Group*" will be applied, which uses "group chaining" to execute all the children group entries, for a "*Local_Multicast_Group*," an "*Inter_Leaf_Group*," and multiple *Path_Group*, in order to multicast the packet to all the other access points within the same broadcast domain, via the leaf switch and the established circuits.

The all-type group entry *Local_Multicast_Group* sends the packet to all the other access points located on the same leaf, which receives the packet.

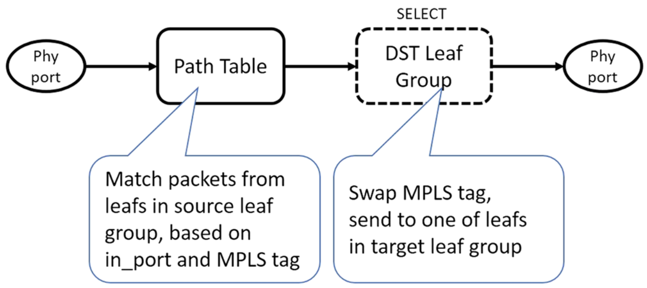
SELECT
Phy port
Path Table
DST Leaf Group
Phy port
Match packets from leafs in source leaf group, based on in_port and MPLS tag
Swap MPLS tag, send to one of leafs in target leaf group

FIGURE 4 The table pipeline design for circuits at spines

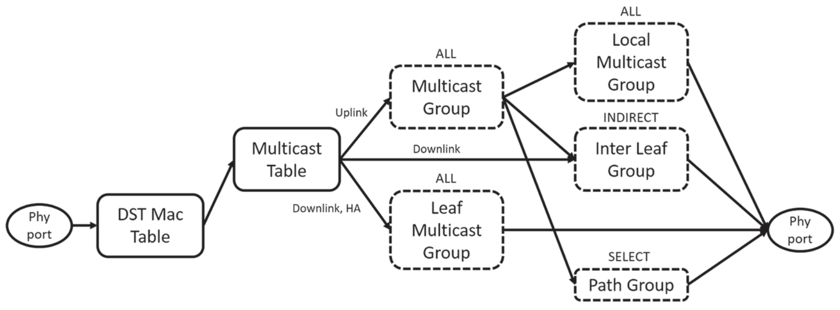

FIGURE 5 The table pipeline design for broadcast forwarding at leaves

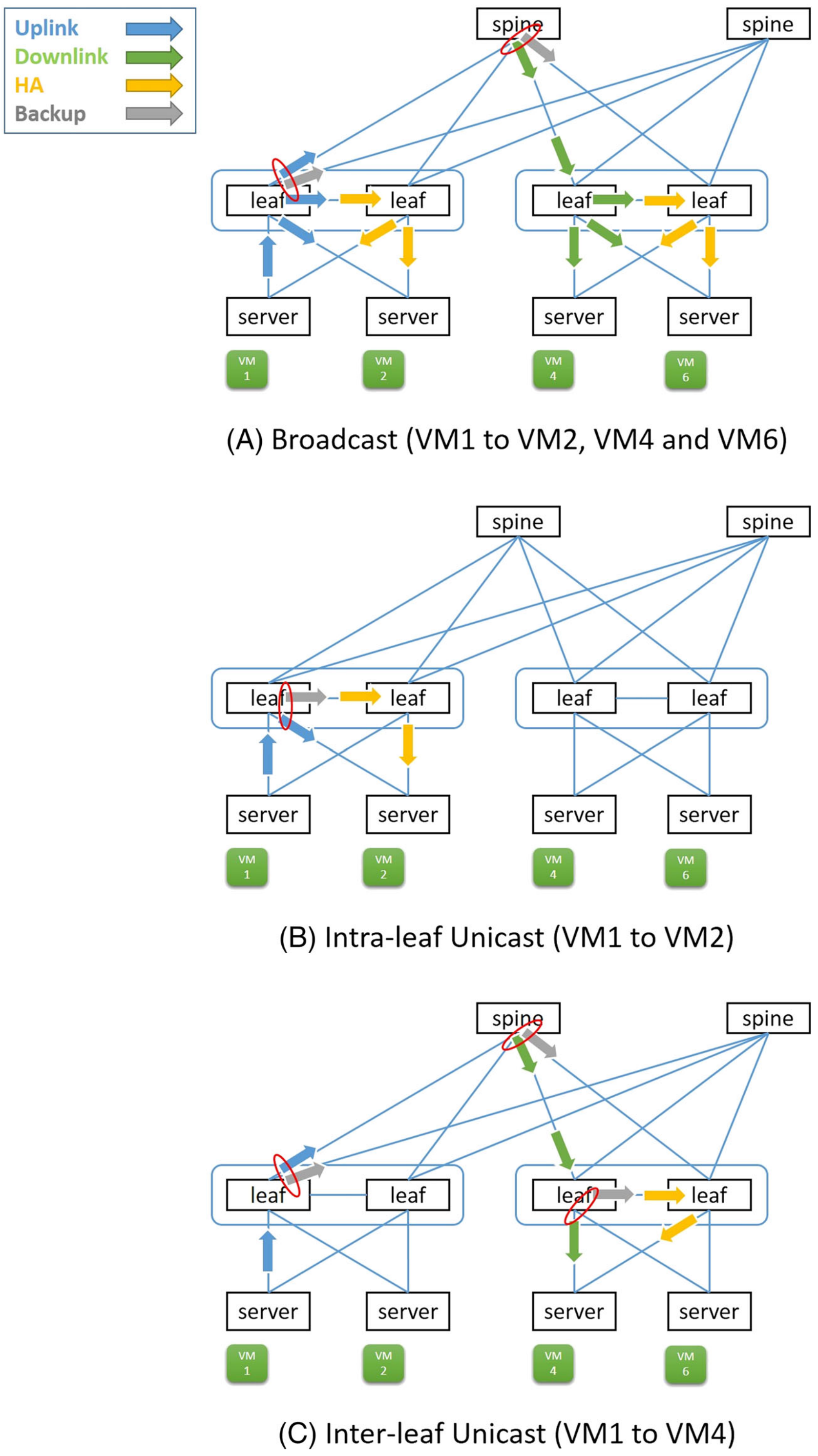
Uplink
Downlink
HA
Backup
spine
spine
leaf
leaf
leaf
leaf
server
server
server
server
VM 1
VM 2
VM 4
VM 6
(A) Broadcast (VM1 to VM2, VM4 and VM6)
(B) Intra-leaf Unicast (VM1 to VM2)
(C) Inter-leaf Unicast (VM1 to VM4)

FIGURE 6 The behavior of broadcast forwarding and unicast forwarding in CBL2

The indirect-type group entry *Inter_Leaf_Group* pushes MPLS tag and sends the packet to the redundant leaf via interleaf link for HA. It should be noted that the MPLS tag pushed by *Inter_Leaf_Group* will be used by the redundant leaf to identify the logical L2 network, since the interleaf link carries more than one logical L2 networks, which may use duplicate VLAN IDs.

The select-type group entry *Path_Group* noted in Section 4.2 pushes MPLS tag and sends the packet to one of the switches in the target leaf group. Since each *Path_Group* in a leaf is mapped onto a circuit, which leads to a remote leaf group, the number of *Path_Group* pointed by the *Multicast_Group* is equal to the number of remote leaf groups.

4.3.2 | Downlink

For a broadcast packet coming from a remote leaf group (via a circuit) with a specified *in_port* and MPLS tag, the MPLS tag will first be stripped and then an all-type group entry "*Leaf_Multicast_Group*" and the *Inter_Leaf_Group* will be applied. The *Leaf_Multicast_Group* sends the packet to all local connected access points of the same logical L2 network.

4.3.3 | High availability

For a broadcast packet coming from the interleaf link with a specified *in_port* and MPLS tag, the MPLS tag will first be stripped and then a corresponding *Leaf_Multicast_Group* will be applied. In our design, an access point can receive broadcast traffic coming from all the other access points in the same logical L2 network via both physical leaf ports, which makes the broadcast forwarding robust to any single-link failure between each server and leaf group.

4.4 | Unicast forwarding

CBL2 forwards a unicast packet based on its *in_port*, VLAN/MPLS tag, and destination MAC address, instead of on the destination MAC only, to ensure that each L2 access is limited to a logical L2 network. Figure 7 shows the table pipeline design for unicast forwarding at leaves. When a unicast packet comes in on any port of a leaf, it will likewise be distinguished by the flow table *DST_Mac_Table* and will be directed to the "*Unicast_Table*" for identifying the destination of the packet and taking corresponding actions. Figure 6B,C shows the behavior of unicast forwarding in CBL2. The forwarding options for packets from different directions are explained below.

4.4.1 | Uplink

For a unicast packet coming from an access point with a specified *in_port* and VLAN tag, a fast-failover-type group entry "*Unicast_Group*" will be applied if the packet's destination MAC address is locally connected (to the same leaf group but on different access point); otherwise, a corresponding *Path_Group* will be applied for sending the packet to one of the leaves in the target leaf group, where the destination MAC address be located on.

The fast-failover-type group entry *Unicast_Group* sends the packet via a primary output port to the corresponding target access point or sends the packet via a backup output port to the redundant leaf when failure occurs on the primary output port.

4.4.2 | Downlink

For a unicast packet coming from a remote leaf group (via a circuit) with a specified *in_port* and MPLS tag, the MPLS tag will first be stripped and then a corresponding *Unicast_Group* will be applied, based on the destination MAC address.

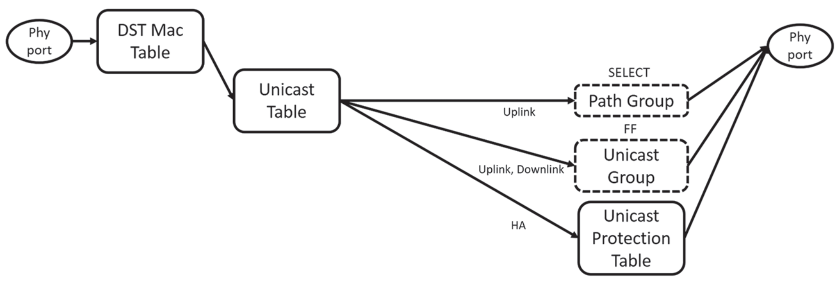

FIGURE 7 The table pipeline design for unicast forwarding at leaves

4.4.3 | High availability

A unicast packet coming from the interleaf link with a specified *in_port* will be directed to the "*Unicast_Protection_Table*," where a flow entry corresponding to the destination MAC address sends the packet directly to the target access point. In our design, an access point can receive unicast traffic coming from all the other access points in the same logical L2 network, via one of the two physical leaf ports depending on their liveness.

4.5 | Dynamic logical L2 expansion

Figure 8 shows the flow chart for adding an access point to a logical L2 network, which enables a newly added access point to send/receive the broadcast traffic to/from all the other access points in the same logical L2 network.

Figure 9 shows the flow chart for adding an MAC to an existing access point, which enables a newly added MAC address to send/receive the unicast traffic to/from all the other MAC addresses under all the other access points in the same logical L2 network.

5 | ANALYSIS OF FLOW ENTRIES CONSUMPTION

In this section, we derive formulas for estimating the consumption of flow entries at leaf switches; the notations are listed in Table 3. For a logical L2 network (L2 broadcast domain), the flow entries consumption on both broadcast forwarding and unicast forwarding at leaf switches is given below.

5.1 | Broadcast forwarding

As noted in Section 4.3, a broadcast packet will be directed to the *Multicast_Table* for processing, based on packet's *in_port* and VLAN/MPLS tag. Specifically, the *Multicast_Table* contains the following types of flow entries:

5.1.1 | Uplink

The number of uplink flow entries at any leaves in a leaf group lg_i^{bd} is equal to $num\_ap_i^{bd}$, the number of locally connected access points.

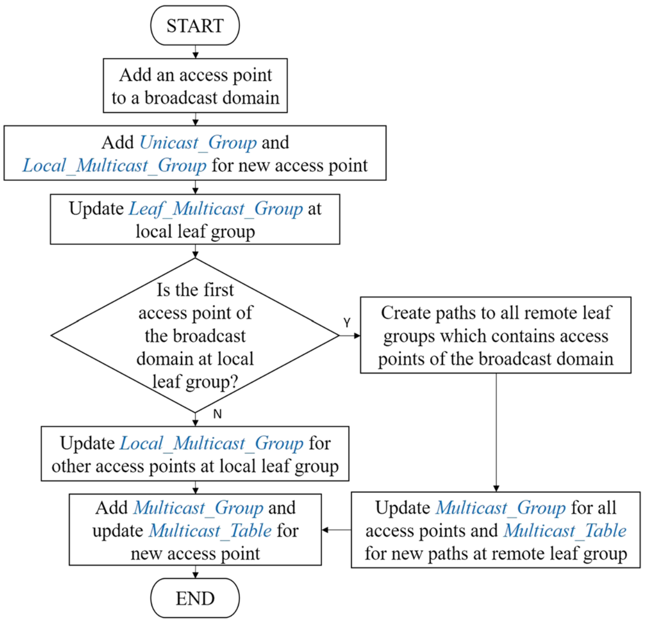
START
Add an access point to a broadcast domain
Add Unicast_Group and Local_Multicast_Group for new access point
Update Leaf_Multicast_Group at local leaf group
Is the first access point of the broadcast domain at local leaf group?
Y
N
Create paths to all remote leaf groups which contains access points of the broadcast domain
Update Local_Multicast_Group for other access points at local leaf group
Add Multicast_Group and update Multicast_Table for new access point
Update Multicast_Group for all access points and Multicast_Table for new paths at remote leaf group
END

FIGURE 8 The flow chart for adding an access point to a logical L2 network

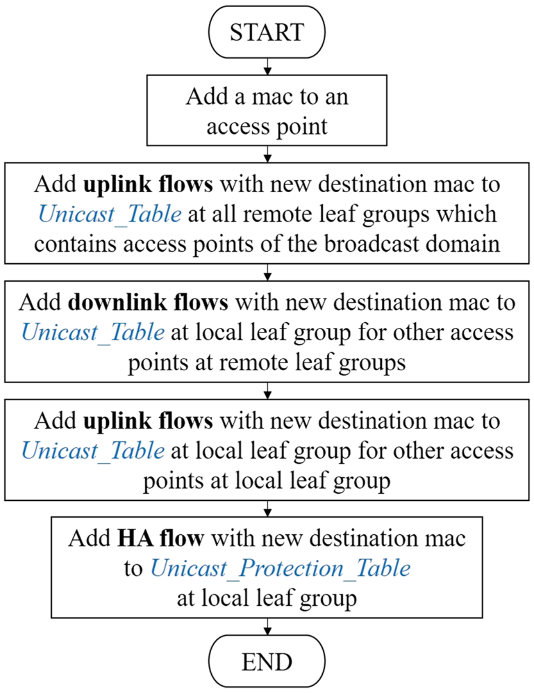
START
Add a mac to an access point
Add **uplink flows** with new destination mac to *Unicast_Table* at all remote leaf groups which contains access points of the broadcast domain
Add **downlink flows** with new destination mac to *Unicast_Table* at local leaf group for other access points at remote leaf groups
Add **uplink flows** with new destination mac to *Unicast_Table* at local leaf group for other access points at local leaf group
Add **HA flow** with new destination mac to *Unicast_Protection_Table* at local leaf group
END

FIGURE 9 The flow chart for adding a MAC to an existing access point

TABLE 3 Notations

Categories	Notations	Descriptions
Spine	num_spine	The number of spines in the leaf-spine fabric
Leaf groups	$LG^{bd} = \{lg_i^{bd} \mid i \geqq 1\}$ num_lg^{bd}	The set of used leaf groups of a logical L2 network The number of elements in LG^{bd}
Access points	$AP_i^{bd} = \{ap_{ij}^{bd} \mid j \geqq 1\}$ $num\_ap_i^{bd}$ $num\_mac_{ij}^{bd}$ $num\_mac_i^{bd}$ num_mac^{bd}	The set of access points of lg_i^{bd} The number of elements in AP_i^{bd} The number of MACs of ap_{ij}^{bd} The number of MACs of lg_i^{bd} The number of MACs of a logical L2 network
Flow entries	$Flow_{MT}^{bd}$ $Flow_{UT}^{bd}$ $Flow_{UPT}^{bd}$	The number of flow entries in *Multicast_Table* at a leaf in lg_i^{bd} The number of flow entries in *Unicast_Table* at a leaf in lg_i^{bd} The number of flow entries in *Unicast_Protection_Table* at a leaf in lg_i^{bd}

5.1.2 | Downlink

The number of downlink flow entries at any leaves in a leaf group lg_i^{bd} is equal to the number of circuits from all remote leaf groups times the number of spines, ie, $(num\_lg^{bd} - 1) * num\_spine$.

5.1.3 | High availability

The number of HA flow entries at any leaves in a leaf group lg_i^{bd} is equal to 1, which handles the broadcast packet coming from the interleaf link.

To summarize the above, the flow entries consumption on broadcast forwarding in *Multicast_Table* at any leaves in a leaf group lg_i^{bd} is given by

$$Flow_{MT}{}^{bd} = num\_ap_i{}^{bd} + \left(num\_\mathrm{lg}^{bd}{-}1\right)*num\_spine + 1. \tag{1}$$

5.2 | Unicast forwarding

As noted in Section 4.4, a unicast packet will be directed to the *Unicast_Table* for processing based on packet's *in_port*, VLAN/MPLS tag, and destination MAC address. Specifically, the *Unicast_Table* contains the following types of flow entries:

5.2.1 | Uplink

The number of uplink flow entries at any leaves in a leaf group $lg_i{}^{bd}$ is equal to the number of flow entries for all locally connected access points to send unicast traffic to all MAC addresses (except the self-owned MAC addresses), ie, $num\_ap_i{}^{bd} * num\_mac^{bd} - \Sigma_j\ (num\_mac_{ij}{}^{bd})$, which is equal to $num\_ap_i{}^{bd} * num\_mac^{bd} - num\_mac_i{}^{bd}$.

5.2.2 | Downlink

The number of downlink flow entries at any leaves in a leaf group $lg_i{}^{bd}$ is equal to the number of circuits from all remote leaf groups times the number of spines and the number of locally connected MAC addresses, ie, $(num\_lg^{bd} - 1) * num\_spine * num\_mac_i{}^{bd}$.

5.2.3 | High availability

The number of HA flow entries at any leaves in a leaf group $lg_i{}^{bd}$ is equal to 1, which directs the unicast packet coming from the interleaf link.

To summarize the above, the flow entries consumption on unicast forwarding in *Unicast_Table* at any leaves in a leaf group $lg_i{}^{bd}$ is given by

$$Flow_{UT}{}^{bd} = num\_ap_i{}^{bd}*num\_mac^{bd}{-}num\_mac_i{}^{bd} + \left(num\_\mathrm{lg}^{bd}{-}1\right)*num\_spine*num\_mac_i{}^{bd} + 1. \tag{2}$$

Since a unicast packet coming from the interleaf link will be further directed to the *Unicast_Protection_Table* in which flow entries forward packets based on their destination MAC address, the flow entries consumption on unicast forwarding in *Unicast_Protection_Table* at any leaves in a leaf group $lg_i{}^{bd}$ is given by

$$Flow_{UPT}{}^{bd} = num\_mac_i{}^{bd}. \tag{3}$$

5.3 | Example

Figure 10 shows an example of the estimate of flow entries consumption at leaf switches. It shows a logical L2 network, which consists of four access points ($ap_{11}{}^{bd}$, $ap_{12}{}^{bd}$, $ap_{21}{}^{bd}$, and $ap_{22}{}^{bd}$), in total two leaf groups ($lg_1{}^{bd}$ and $lg_2{}^{bd}$), and each server connected to each access point has five VMs (five MACs), which belong to the logical L2 network.

As per formulas (1), (2), and (3), we can obtain the flow entries consumption on both broadcast forwarding and unicast forwarding at any leaves in $lg_1{}^{bd}$ and $lg_2{}^{bd}$:

5.3.1 | Broadcast forwarding

$Flow_{MT}{}^{bd} = 5$.

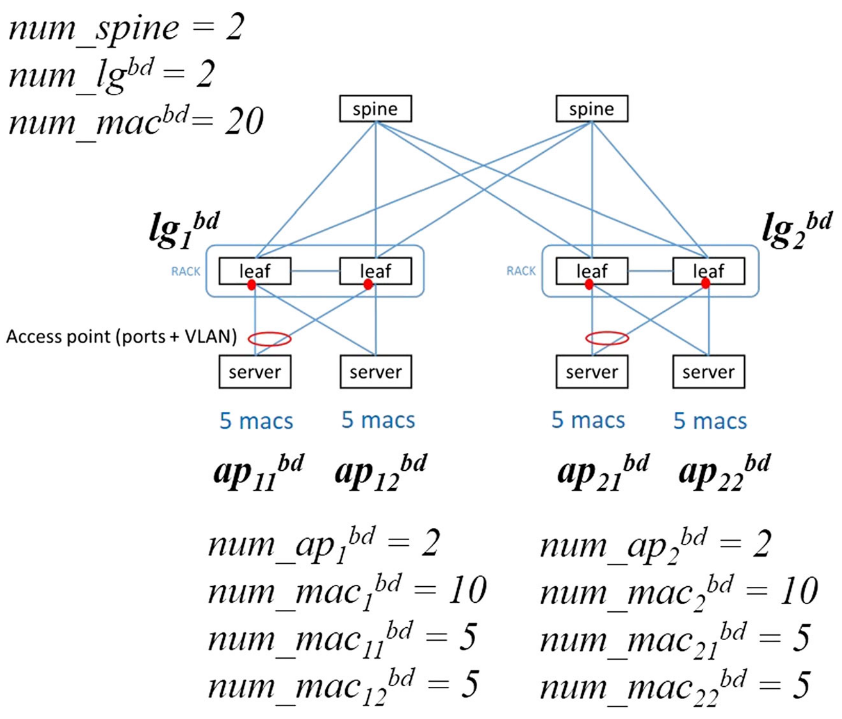

FIGURE 10 An example of the estimate of flow entries consumption at leaf switches

5.3.2 | Unicast forwarding

$Flow_{UT}{}^{bd} = 51.$
$Flow_{UPT}{}^{bd} = 10.$

6 | EVALUATION AND DISCUSSION

We compared the throughput performance with OVS between cases with and without VXLAN tunneling and compared the average address resolution time for ARP requests in a leaf-spine fabric between broadcast handling by control plane and data plane. For HA of CBL2 deployment, we measured the protection switching time in the presence of switch and link failures.

6.1 | Compared with VXLAN tunneling: Direct transmission in OVS improves throughput performance by 58%

In the environment for OVS throughput measurement, as shown in Figure 11, we created two VMs (each with one CPU and 2-GB RAM) in a physical server and used Mininet[23] to run an OVS *s1* (OVS 2.5.2) and a host process *h1* in each VM. The network interfaces of these 2 VMs were connected with VirtualBox host-only networking.[24]

In the case of transmission with VXLAN tunneling, we used OVS commands to build a VXLAN tunnel between two OVSs in the two VMs (with a tunnel ID and the IP addresses of the two VMs) and also to install two flow entries to each OVS for bidirectional forwarding between the host port and the tunnel port (ie, port 1 and port 9 in Figure 11). For the direction from host to tunnel, each packet will be assigned a given tunnel ID and then be forwarded to the tunnel. In addition, we set the maximum transmission unit (MTU) to 9000 bytes on our VXLAN network. Then, we used iPerf[25] to send TCP traffic with a default TCP window size of 85.3 kB from one host process to the other across the tunnel for 1 minute and finally obtained about 2.01 Gbps throughput.

With the same iPerf configuration and MTU size, we obtained about 3.18 Gbps throughput after removing the settings of VXLAN tunnel, which improved throughput performance by 58%. Figure 12 shows the throughput performance for each case.

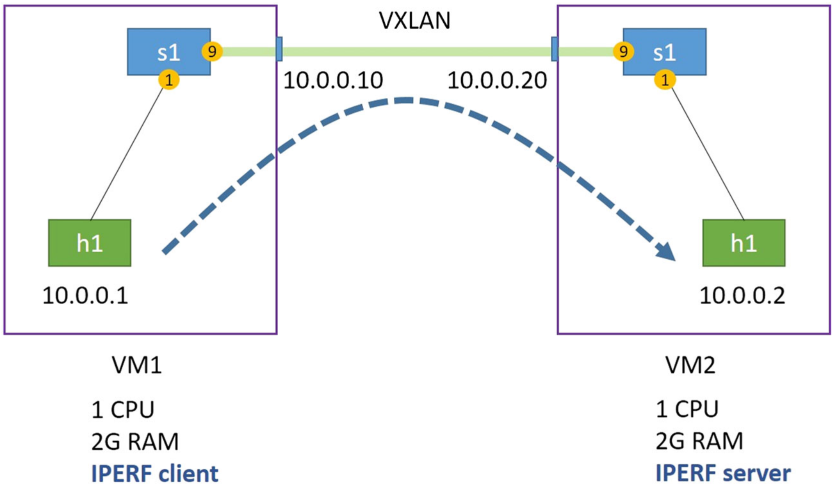

FIGURE 11 The environment for Open vSwitch (OVS) throughput measurement

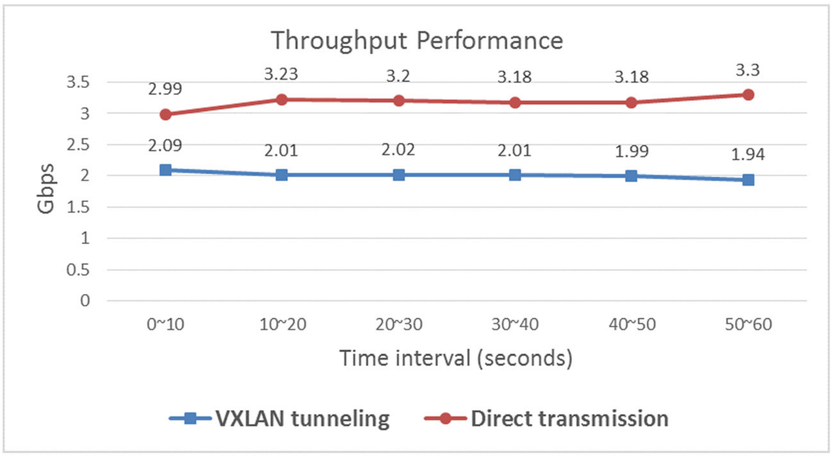

FIGURE 12 The throughput performance of Open vSwitch (OVS) with and without Virtual eXtensible Local Area Network (VXLAN) tunneling

6.2 | Compared with control-plane broadcast forwarding: Data-plane multicasting for ARP reduces address resolution latency from 149 to 0.5 ms

Figure 13 shows our environment for the measurement of address resolution latency. We used Mininet to create a leaf-spine network with one spine and 10 leaves in a VM, which has one CPU and 2-GB RAM. Each leaf connected a host process. In order to measure the address resolution latencies for ARP requests between hosts, we cleaned the ARP tables of all hosts first and then pinged from a host (h1) to the other nine hosts (h2-h10). Since the response time of the first ping includes address resolution time and actual ping round trip time (RTT), which equals the response time of the other ping (about 0.055 ms), we obtained the address resolution latency by subtracting ping RTT from the response time of the first ping.

In the case of control-plane broadcast forwarding, we intercepted ARP requests at leaves and forwarded them to a RYU[26] controller in the same VM by the preinstalled broadcast flow entries (whose output actions forward ARP request packets to the controller) and then the controller redirected the ARP request to the requested VM directly via the PacketOut message[15] in OpenFlow 1.3. In this case, the ARP reply would be forwarded based on packet's destination MAC address by the preinstalled unicast flow entries. On the other hand, for the case of data-plane multicasting, we created a logical L2 network by CBL2 controller application to bridge these hosts (h1-h10) in data plane.

Figure 14 shows the average address resolution latencies for different destination hosts in both cases. As a result of the huge variation in the results of control-plane forwarding, we performed the ping test thrice for each destination host. It can be seen from Figure 14 that the data-plane multicasting reduced address resolution latency from 149 to 0.5 ms on average.

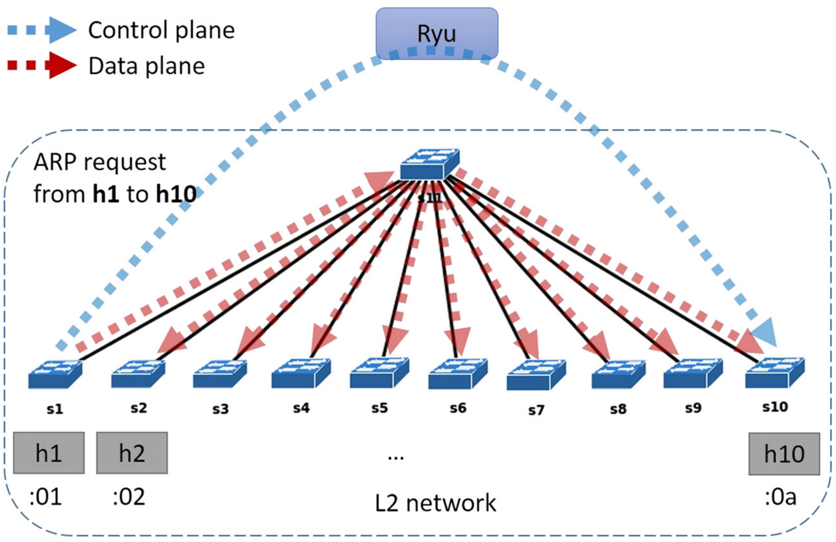

FIGURE 13 The environment for the measurement of address resolution latency

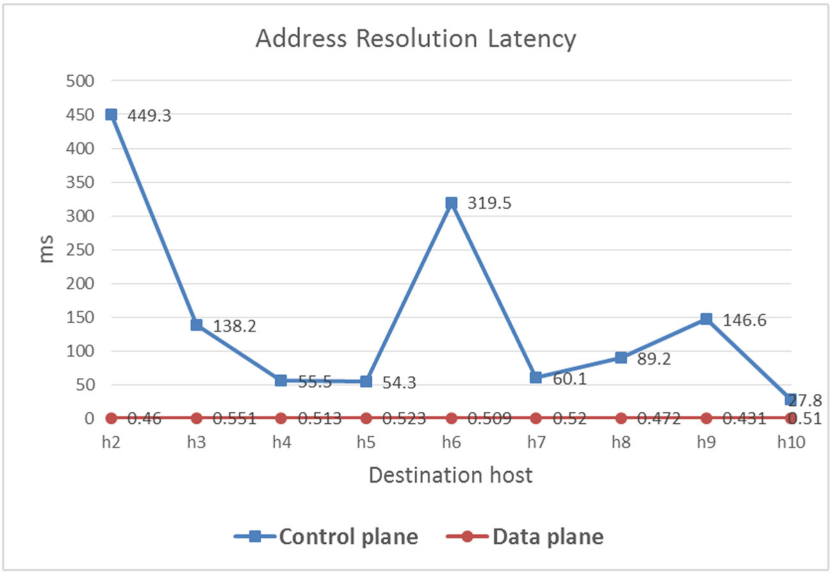

FIGURE 14 The average address resolution latencies for different destination hosts

However, the data-plane multicasting brings about flow entries overhead compared with control-plane broadcast forwarding. As noted in Section 5.1, the flow entry consumption on data-plane multicasting at any leaves in a leaf group lg_i^{bd} is given by $num\_ap_i^{bd} + (num\_lg^{bd} - 1) * num\_spine + 1$, where $num\_ap_i^{bd}$ is the number of locally connected access points and $(num\_lg^{bd} - 1) * num\_spine$ is the number of remote leaf groups times the number of spines. And as for control-plane broadcast forwarding, only one-flow entry is needed at each leaf for intercepting the ARP requests and forwarding them to the controller. Consequently, the flow entries overhead of data-plane multicasting is equal to $num\_ap_i^{bd} + (num\_lg^{bd} - 1) * num\_spine$.

To give an example of flow entries overhead at a leaf for a logical L2 network, we assumed that the leaf connected four spines ($num\_spine = 4$) via 4 40G uplinks (total 160G uplink). We computed flow entries overhead for three cases of different oversubscription ratios, 1:1, 2:1, and 3:1. For oversubscription ratio 1:1, 16 servers ($num\_ap_i^{bd} = 16$) were connected to 16 10G downlinks of the leaf (total 160G downlink). Similarly, 32 and 48 servers were connected to the leaf for oversubscription ratios 2:1 and 3:1, respectively. Figure 15 shows the flow entries overhead of data-plane multicasting at a leaf for a logical L2 network. It can be seen from Figure 15 that the flow entries overhead at a leaf increased linearly with the number of remote leaf groups.

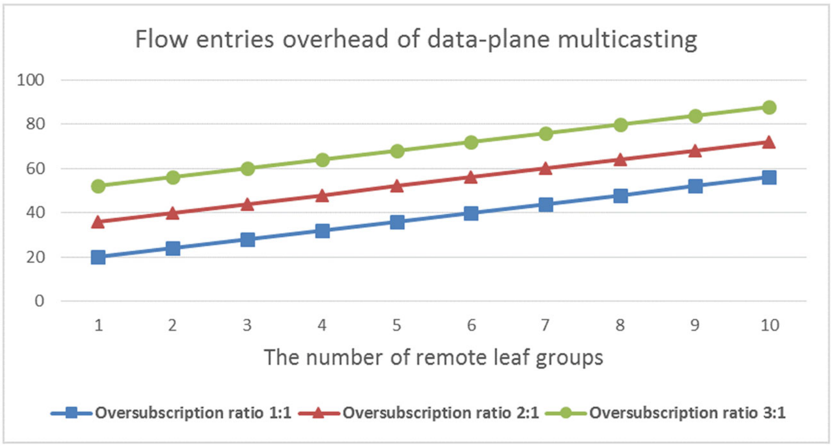

FIGURE 15 The flow entries overhead of data-plane multicasting at a leaf for a logical L2 network

6.3 | CBL2 provides 0.6, 0.4, and 11-ms protection switching time, respectively, in the presence of switch failure, link failure, and port shutdown

To measure the protection switching time in practical deployment, we built a leaf-spine fabric with 40GE ports of 6 NoviSwitch 2128[27] to be controlled by our CBL2 controller application, as shown in Figure 10, and we used commercial testing solution to send unicast packets with the packet size of 1400 bytes at 10-Gbps line rate (about 880 Kpps) from one access point (10GE port) to another access point located on different leaf group within a logical L2 network. We simulated single switch failure (leaf or spine) by disconnecting the power cable and simulated single-link failure by pulling out the optical fiber cables from switches. We also tried to administratively bring the port interface down. The protection switching time could be derived by dividing the number of lost packets by packet rate. The measured results show that CBL2 on average provides 0.6, 0.4, and 11-ms protection switching time, respectively, in the presence of switch failure, link failure, and port shutdown in practical deployment.

According to the test results of convergence on failover[28] in an ACI[20] fabric with 10-Gbps unicast test traffic across leaf switches, ACI provides 571, 286, and 208-ms protection switching time, respectively, in the presence of spine switch failure, leaf switch failure, and intraDC link failure. It can be seen that CBL2 outperforms ACI in path protection, because of the data plane level protection enabled by the fast failover feature of OpenFlow group entries.

7 | CONCLUSION AND FUTURE WORK

In this paper, we proposed a CBL2, which replaces the overlay tunneling enabled by OVSs in servers with underlay label switching performed by a software defined leaf-spine fabric, to enhance the transmission performance in logical L2 networks.

CBL2 achieves scalable L2 with per-port VLAN, which allows VLAN reusing for multiple different logical L2 networks and uses data-plane multicasting to achieve virtual L2 broadcasting. Besides, CBL2 provides HA in the presence of switch and link failures and keeps data plane traffic unaffected in the absence of controller.

Our evaluations indicate that direct transmission in OVS improves throughput performance by 58% compared with VXLAN tunneling and data-plane multicasting for ARP reduces address resolution latency from 149 to 0.5 ms compared with control-plane broadcast forwarding. Our evaluation results also show that CBL2 provides 0.6, 0.4, and 11-ms protection switching time, respectively, in the presence of switch failure, link failurem and port shutdown in practical deployment.

Furthermore, our table pipeline designs in leaf and spine switches support dynamic expansion of logical L2 network without service interruption.

In future work, we plan to deploy traffic engineering in the leaf-spine fabric to minimize overall end-to-end delay for all circuits, by implementing automatic load sharing adjustment on group entries of circuits at spines and leaves based on the collected traffic information.

ORCID

Yao-Chun Wang https://orcid.org/0000-0002-1531-9378

REFERENCES

1. Mell P, Grance T. *The NIST Definition of Cloud Computing*. Gaithersburg, MD, USA, NIST Special Publication: National Institute of Standards and Technology; Sep. 2011:800-145.
2. Amazon Virtual Private Cloud. [Online]. Available: https://aws.amazon.com/tw/vpc/, Accessed on: January 11, 2018.
3. Google Cloud Platform. [Online]. Available: https://cloud.google.com/vpc/, Accessed on: January 11, 2018.
4. Cisco Data Center Spine-and-Leaf Architecture: Design Overview White Paper. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white-paper-c11-737022.html, Accessed on: April 22, 2019.
5. Leiserson CE. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Trans Comput*. Oct. 1985;34(10):892-901.
6. Goth G. Software-Defined Networking Could Shake Up More than Packets. *IEEE Internet Comput*. 2011;15(4):6-9.
7. Fizi FS, Askar S. A novel load balancing algorithm for software defined network based datacenters. *Proc IEEE Int Conf Broadband Commun Next Generat Netw Multimedia Appl (CoBCom)*. Sep. 2016;1-6.
8. Tian J, Qian Z, Dong M, Lu S. FairShare: Dynamic Max-Min Fairness Bandwidth Allocation in Datacenters. *IEEE Trustcom/BigDataSE/ISPA*. 2016;1463-1470, 2016.
9. Mohan P, Divakaran D, Gurusamy M. Proportional Bandwidth Sharing Using Bayesian Inference in SDN-based Data Centers. *IEEE Int Conf Commun (ICC)*. 2016;1-6, 2016.
10. Fang S, Yu Y, Foh CH, Aung KMM. A loss-free multipathing solution for data center network using software-defined networking approach. In: *APMRC, 2012 Digest*. Singapore: IEEE; 2012:1-8.
11. Mahalingam M, Dutt D, Duda K, et al. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. *IETF-RFC*. 2014;7348.
12. Open vSwitch. [Online]. Available: http://www.openvswitch.org, Accessed on: January 11, 2018.
13. Yan, Yaohua and Wang, Hongbo, Open vSwitch Vxlan performance acceleration in cloud computing data center, 2016 5th International Conference on Computer Science and Network Technology (ICCSNT).
14. Data Plane Development Kit. [Online]. Available: http://dpdk.org, Accessed on: January 11, 2018.
15. OpenFlow. [Online]. Available: https://www.opennetworking.org/projects/open-datapath/, Accessed on: January 11, 2018.
16. Chen, C., Liu, C., Liu, P., Loo, B., Ding, L.: A scalable multi-datacenter layer-2 network architecture. In: 1st ACM SIGCOMM Symposium on Software Defined Networking Research, Article No. 8. ACM, New York (2015)
17. Cord. [Online]. Available: https://wiki.opencord.org/, Accessed on: January 11, 2018.
18. Hwang, R.H and Tang, Y.C., Fast failover mechanism for sdn-enabled data centers, 2016 International Computer Symposium (ICS), pp. 171-176, Dec 2016.
19. Raeisi, B. and Giorgetti, A., Software-based fast failure recovery in load balanced SDN-based datacenter networks, Proc. 2016 6th International Conference on Information Communication and Management (ICICM), 2016.
20. Application Centric Infrastructure (ACI). [Online]. Available: https://www.cisco.com/c/en/us/solutions/data-center-virtualization/application-centric-infrastructure/index.html, Accessed on: April 30, 2019.
21. Big Cloud Fabric. [Online]. Available: https://www.bigswitch.com/products/big-cloud-fabrictm/switching-fabric-overview, Accessed on: January 11, 2018.
22. Cisco Application Policy Infrastructure Controller (APIC). [Online]. Available: https://www.cisco.com/c/en_au/products/cloud-systems-management/application-policy-infrastructure-controller-apic/index.html, Accessed on: January 11, 2018.
23. Mininet. [Online]. Available: http://mininet.org/, Accessed on: January 11, 2018.
24. Oracle VM VirtualBox User Manual. [Online]. Available: https://www.virtualbox.org/manual/ch06.html#network_hostonly, Accessed on: April 22, 2019.
25. IPerf. [Online]. Available: https://iperf.fr/, Accessed on: January 11, 2018.
26. Ryu. [Online]. Available: https://osrg.github.io/ryu/, Accessed on: January 11, 2018.
27. NoviSwitch. [Online]. Available: https://noviflow.com/products/noviswitch/, Accessed on: January 11, 2018.
28. Cisco Live BRKACI-3503 Session. [Online]. Available: https://clnv.s3.amazonaws.com/2015/usa/pdf/BRKACI-3503.pdf, Accessed on: April 30, 2019.

How to cite this article: Wang Y-C, Lin Y-D. Circuit-based logical layer 2 bridging in software-defined data center networking. *Int J Commun Syst*. 2019;32:e4128. https://doi.org/10.1002/dac.4128