S-Fabric: Towards Scalable and Incremental SDN Deployment in Data Centers

Haoyun Shen
Georg-August-University Göttingen/GWDG
haoyun.shen@gwdg.de

Hao Wang
Georg-August-University Göttingen/GWDG
hao.wang@gwdg.de

Philipp Wieder
Georg-August-University Göttingen/GWDG
philipp.wieder@gwdg.de

Ramin Yahyapour
Georg-August-University Göttingen/GWDG
ramin.yahyapour@gwdg.de

ABSTRACT

Scalable and robust SDN requires the controller to be distributed. In many SDN designs, the distributed controllers are acting as replicas by forming clusters. For large-scale data centers across multiple geographically distributed locations, the controllers have to maintain a synchronized global view. These restrict themselves on single point of failure, low scalability, more communication effort, bad isolation, and rigid deployment.

In this paper, we propose S-Fabric, a novel data center network design, which provides a sliced control plane and a policy-based user-defined data plane. By slicing the network through flows, and assigning a non-replica controller to each slice, S-Fabric achieves flexibility and elasticity, while ensuring isolation and separation. We leverage (but not limited to) a two-tiered spine and leaf architecture, and define forwarding rules for spine, leaf and edge switch respectively. By simplifying the flow table, S-Fabric keeps the number of forwarding rules on spine switches equal to the number of used leaf/edge ports inside a data center. By matching subnets in slices to VLANs on the edge switches, S-Fabric brings backwards compatibility to traditional data centers. S-Fabric enables an incremental deployment of SDN in traditional data centers, since it requires no SDN capability on the spine/core switches.

CCS Concepts

• **Networks** → **Network architectures**, **Data center networks**

Keywords

Network slicing, multi-tenancy, cross data center VM migration, network architecture redesign

ICTCE 2017, October 22–24, 2017, Osaka, Japan
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5315-1/17/10...$15.00
https://doi.org/10.1145/3145777.3145790

1. INTRODUCTION

An important evolution of cloud infrastructure is to deploy software-defined networks (SDN [1]) in data centers. Together with virtualized compute and storage components, all resources run and serve diverse customized applications in an orchestrated way. The global point of view and resource de- / composition makes data centers more capable of scaling up/out on demand, thus cost and energy effective. More flexible data centers can better serve a wide scope of applications, such as Fog Computing, 5G, Internet of Things (IoT), smart city, Industrie 4.0, to name a few.

A usual situation in such software-defined data centers is to migrate services to a place with more spare resources in order to improve resource utilization, while the services can still be located by customers or assembled into a service chain. It is challenging because in many cases the services are network address based, e.g., IP address based, and migration might change the address and cause service interruption or security risk. One solution is to keep the address unchanged and do address tunnelling when migrating the services. Hence, we need to introduce a new address space. Another challenge is, SDN often means centralized control, which is however, against the practical trends of networks, i.e., computing distributed to decentralized network clusters. One example of that is the edge/fog computing, where cloud computing capabilities are pushed from the core to the edge data centers. In this case, a global controller may easily become a bottleneck. So, we should assign control tasks to distributed controllers. This brings new questions: how should we distribute the controllers? It is rigid to assign controller according to physical network sections (physical devices like a subset of switches). We attempt to slice network with flows and assign controllers to flows, while ensuring high scalability.

In this paper, we propose S-Fabric (S for slice), a novel data center network design, which provides a sliced control plane as well as a user-defined data plane. S-Fabric controllers push user-defined forwarding rules to switches, possibility by utilizing protocol such as Openflow, and sets table miss actions to drop. Another key idea of S-Fabric is to isolate and separate the network by slices, which are defined as sets of flows. We then assign flows to tenants. Therefore, a tenant can own multiple slices. Moreover, by fine-grained address formatting and MAC address rewriting, S-Fabric supports address tunnelling and gains high scalability. With these features, S-Fabric supports smooth service migration, exhibits high flexibility and elasticity. By mapping slice to VLAN/Subnet, S-Fabric enables the interoperability with legacy data center infrastructure, thus, an incremental deployment of SDN/NFV/SFC in data centers without SDN infrastructure. Further, S-Fabric also provides northbound interface for orchestration and automation.

The rest of this paper is organized as follows: Section 2 discusses related work. Section 3 presents the concept of S-Fabric and the architecture of network design. Then, Section 4 and 5 respectively illustrates the control and data plane actions of S-Fabric. Section 6 briefly concludes this paper.

2. RELATED WORK

In this section, we study related work and sort out comparative properties, according to two techniques - address tunnelling and network slicing.

Address tunneling

Monsoon [2] brings up a huge layer-2 domain to enhance the ability to dynamically reassign servers running in the data centers. The paper creates a single layer-2 domain that scales up to 100k servers. By encapsulating 3 layers of Ethernet headers, Monsoon forwards packets from servers to the random chosen core and then to destination Top-of-Rack (ToR) switch and then to destination server. A logically centralized server is required to keep track of the mapping between server IP address and corresponding ToR switch MAC address. The servers are required to be modified such that the mapping request to the centralized server are implemented instead of a normal ARP request. Moreover, the scalability of up to 100k servers may not be sufficient for novel use cases such as smart cities, IoT, Industrie 4.0.

VL2 [2] proposes a virtual layer-2 domain that scales up to 100k servers, by encapsulating 3 layers of layer-3 headers to data flows. Servers are assigned with Application Addresses (AA) and network infrastructures such as switches and interfaces are assigned with Location Addresses (LA). All AAs are within one single shared IP subnet to eliminate ARP and DHCP scaling bottlenecks. A VL2 agent is implemented on each server to redirect the broadcast ARP request to a unicast query to the VL2 directory system, which can be a bottleneck of the data center network when the data center scales up.

Portland [4] assigns a pseudo MAC address (PMAC) to an actual MAC address (AMAC) of each virtual machine. A fabric manager deploys AMAC<->PMAC OpenFlow rewriting rules to ingress and egress switches of Portland network, according to a mapping table of VM IP address, AMAC, and PMAC. Portland does not require additional header encapsulation. The fabric manager that keeps track of global AMAC-PMAC-IP mapping, can be a scalability bottleneck and single point of failure.

Crossroad [5] provides a network fabric design which facilitates seamless VM mobility across data centers. The paper extends Portland by assigning each VM a pseudo IP address (PIP) to enhance cross data center packets forwarding. A controller for each data center is still logically centralized. Multi-tenancy support is not defined in the paper.

DCNet [6] introduces a routable MAC (RMAC) to route server to server traffic within a data center. The paper requires no address rewriting but makes changes directly on the end hosts. DCNet assumes all devices belong to one organization and each device should be configured with a IPv6 address, which can be unpractical in many use cases, such as smart cities, cloud provider networks.

SVDC [7] leverages MAC-in-MAC encapsulation by adding a layer-2 header with the MAC addresses of edge switches as the source and destination address, and utilizing VLAN tag to identify the outgoing port. SVDC claims to be more scalable than VLAN, VxLAN and NetLord in large data center scenarios. But still, SVDC utilizes a global controller which has knowledge to all tenants and can be facing synchronization problems in case of geographically distributed data centers.

Network Slicing

FlowVisor [8] defines a slice as a set of flows running on a topology of switches. It assigns one OpenFlow controller for each slice. The FlowVisor servers work as a hypervisor layer between slice controllers and switches, provisioning switch resources for slice controllers. FlowVisor guaranties network isolation between slices and is transparent to controllers and switches. However, by introducing a middle layer, FlowVisor increases the latency at each flow setup. The controllers are no single point of failure to the entire network anymore, but FlowVisor instances become the new ones. S-Fabric brings a solution to this problem, by letting each switch have connection to multiple controllers simultaneously, and performing flow isolation based on destination address.

3. S-FABRIC ARCHITECTURE

In this section, we present the basic concept of S-Fabric. First, we explain the basic terminologies. Then we introduce an example architecture of two interconnected data centers, which apply S-Fabric.

3.1 Global topology

As presented in Figure 1, S-Fabric defines following types of devices in the architecture:

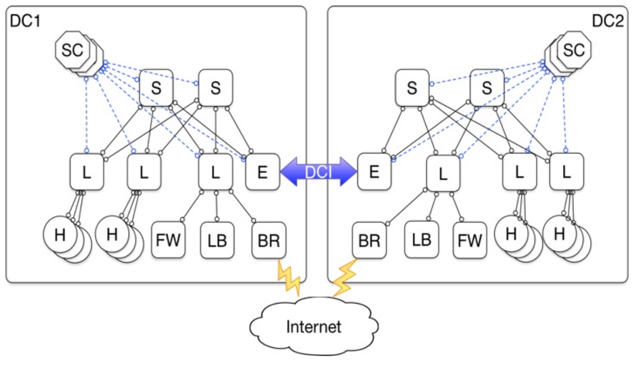

Figure 1. Example architecture of two data centers applying S-Fabric

Host (H)
A host can be a physical server, a virtual machine, or a container in S-Fabric network. A host connects to S-Fabric via a leaf switch. A host can be moved within a data center or across multiple data centers.

Slice Controller (SC)
S-Fabric brings up a sliced control plane by assigning a set of flows running on a topology of switches to a slice. A tenant network consists of a set of slices. Each slice is managed by a slice controller. The flows are classified to a slice based on destination IP addresses in the packets. To allow the backwards compatibility with legacy data center design and reduce the number of flow table entries, using wildcard, one or more IP subnets are assigned to a slice. A slice controller keeps track of the mapping between actual MAC address and location MAC address for each IP address. A slice controller installs the MAC-address-rewriting rules and packets-forwarding rules for flows within the slice, on all relevant switches of each flow. The installed rules are defined before the

initialization of SC, by the orchestrator or manually by administrator, according to SFC's[9] definition of a slice. Furthermore, a slice controller handles ARP requests within one slice. In this way, S-Fabric requires no changes on hosts.

Leaf Switch (L)
A leaf switch is defined as a ToR switch of a set of hosts. To serve the entire service function chain, special hosts with network functions such as firewall (FW), load balancer (LB), boarder router (BR), IDS/IPS etc. are also connected to S-Fabric on leaf switches. A leaf switch has a connection to each spine switch within a data center. ECMP is used to load balance host to host flows at leaf switch to decide on which spine switch to forward the packets to.

Spine Switch (S)
A spine switch connects to all leaf switches and edge switches within a data center. The forwarding tables on every spine switch within a data center are identical. There are no MAC-rewriting rules on spine switches, but only forwarding rules. Thus, it doesn't even matter if a spine switch is SDN or non-SDN.

Edge Switch (E)
An edge switch connects to one or several edge switches within other data centers with data center interconnects (DCI). DCI is normally a set of optical connections, e.g., DWDM, between data centers. In some special use cases, the DCI can also be a normal Ethernet cable, or a l2VPN connection over internet. An edge switch has a connection to each spine switch within a data center.

3.2 Addressing
When migrating address-bounded (mainly IP address-bounded) services from one location to another, we can hardly assume the source and destination are using the same address. We introduce new address space and rewrite this into the actual MAC address. All fabric devices belong to this space. We name this address as location MAC (LoMAC) address. Accordingly, a slice controller keeps track of the mapping between actual MAC address and LoMAC address for each IP address. The LoMAC is defined as

dc_id.switch_id.port_id

Obviously, LoMAC divides connected network fabrics into three levels: data center, switches, and ports. The 20-bit long *dc_id* identifies the data center, thus, support more than one million data centers. *switch_id* is 16-bit long to identify the switches belonging to one data center. That means for each data center more than 65 thousand switches are supported. And the *port_id* is 12-bit long to identify the specific port designated to the LoMAC, which means, for each switch more than 4 thousand ports are supported.

Note, we must not limit the meaning of *dc_id* only to data center. In fact, it can simply represent a cluster of devices with certain processing capacities, e.g., a cluster of sensors and sinks. The same idea applies to the other two IDs. Therefore, LoMAC extends its scope beyond SDN clouds to various applications, from traditional large-scale clouds to many popular novel network concepts.

4. CONTROL PLANE INITIALIZATION
A key factor of control plane in S-Fabric design is slice. As defined above, a slice consists of flows. Which flows a slice can include is done either manually or automatically with a software orchestrator of a fully automated service function chain. This information is in the tenant definition. In our design, we install a controller for each slice in a container (e.g., a VM or a software container like Docker). A light-weighted container is quite meaningful when the number of slices is large. Next, we first state how to initialize the slice information, then based on the initialized slice topology we describe the slice maintained subnet, LoMAC as well as contract table initialization. Besides these, we also explain how SC installs forwarding rules.

4.1 Slice table initialization
Slice information includes three parts: a slice ID, an IP address, and a LoMAC. This LoMAC indicates the physical location of the container. The IP is a unique identifier of the SC and will show up as management IP in the slice tables of its neighbors. A slice table is initialized based on the user-defined business logic. For example, if we define that all slices within a tenant can communicate with each other, then the slice table should be initialized with all other slices of the tenant, with their *slice_ids*, management IPs, and LoMACs. The information comes from the orchestrator or administrative commands. The slice table can only be updated by an orchestrator or administrative commands. Note, it cannot be updated by other controllers. Each entry of this table represents the neighborhood between the current controller and the controller identified with *slice_id*. The neighborhood means these two controllers can send/receive LoMAC requests and responses with each other.

4.2 Slice topology initialization
We define a slice topology during slice initialization. The slice topology indicates the involved switches, subnets/hosts, border router, firewalls, load balancers, NAT routers, IDS/IPS etc.

We use a physical view to represent where the involved service functions are physically located and how they are connected to S-Fabric. We use a logical view to define the logical paths of the service function chain.

Figures 2 shows an example slice by comparing its physical view to its logical view as for SFCs. A, B1, B2, C are hosts in subnet A, subnet B and subnet C. Subnet C is connected via the internet to BR over IPSec VPN.

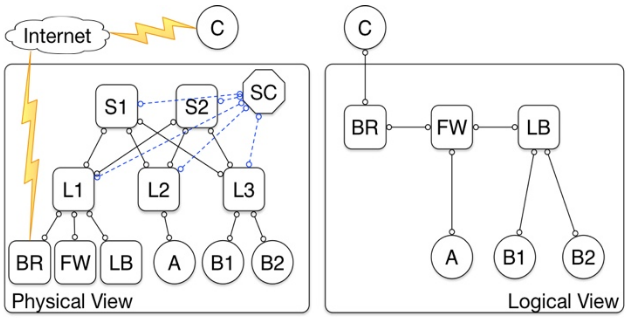

Figure 2. Physical view and logical view of an example slice

4.3 Subnet table initialization
Each SC maintains a subnet table locally. Each entry of this table represents the subnet controlled by the slice controller. The ARP requests within this subnet will all be forwarded to the SC.

The subnet table can only be updated by an orchestrator or administrator instead of other controllers. The updates on a subnet table will result R1 updates in the flow table of the leaf switches.

4.4 LoMAC table initialization
A slice controller keeps track of the mapping between actual MAC address and location MAC address (LoMAC) for each IP address. The LoMAC is used to forward packets to the correct location of the destination.

Each entry in this table represents a mapping between an IP address and its location MAC address, which indicates the physical location of the entity. The entity can either be a controller, a host, a firewall/load-balancer/IDS/IPS, a boarder router, or a switch. Each SC initializes the LoMAC table by adding the necessary entries according to the slice topology. The SC continues the LoMAC Table initialization by adding the LoMAC-AMAC-AIP mapping for each host in each of its subnets.

For a host connected to S-Fabric through a boarder router, such as host C in the example slice, the actual MAC and location MAC of the inside port on boarder router are filled in the LoMAC table.

For a host in another data center, which is connected via edge switch, the actual MAC of the host and location MAC of the outgoing port of edge switch is filled in the LoMAC table.

4.5 Contract table initialization

Each SC maintains a contract table, to keep track of the allowed unidirectional flow from one subnet to another. The contracts are predefined by orchestrator or administrator within tenant definition. In case of the slice of a source or a destination subnet is not local in the contract table, the SC checks the slice table to find out if the neighbor slice is in the same data center, by checking the corresponding *dc_id* in the LoMAC.

Case 1: When the neighbor slice is in the same data center, the SC sends a request to the neighbor slice, querying the LoMAC entries belonging to this subnet. Upon receiving responses, the SC appends the LoMAC entries to it local LoMAC table.

Case 2: When the neighbor slice is in another data center, or connected through a border router over the internet, the SC checks the predefined rules in its local LoMAC table, to make sure the LoMAC of each host in the subnet is the location of a DCI port of the corresponding edge switch.

4.6 Forwarding rules installation

After contract table initialization, the SC installs rules on relevant *leaf* switches, as long as the destination subnet belongs to the local slice in the contract table, or it is a subnet from neighbor slice in another data center.

For the packets with destination IP matching a subnet of another slice in the same data center, the rules will not be installed by the local slice controller, but by the neighbor slice controller who has the destination subnet.

By installing rules only when a valid contract table entry exists, and configuring the table miss action to drop, S-Fabric enables access control based on known flows. Thus, unknown flows are discarded by default.

4.7 ARP resolution

ARP requests are sent from hosts in two cases: for the destination if a host wants to reach anther host in the same subnet; for the default gateway if destination is in another subnet.

All ARP requests are forwarded according to its destination IP address to its corresponding slice controller, or the default controller. Upon receiving the ARP request, the SC checks its local LoMAC table for the destination IP, and replies with the corresponding AMAC. An ARP request will be discarded by default if no match of the destination IP in LoMAC table exists.

5. DATA PLANE

From the data plane point of view, the three types of switches Spine, Leaf, and Edge are assigned to different tasks.

5.1 Leaf operation

A leaf switch is responsible for ARP forwarding, local switching, LoMAC rewrites, access control and forwarding packets to spines.

When packets come from ingress ports, i.e., from hosts or special hosts, a leaf switch performs following actions:

- If the packets type is ARP, rewrite destination MAC to the LoMAC of the responsible SC, and forward it to any spine using ECMP.
- If destination IP is locally connected and the destination MAC is not 0000.0000.0000, forward it to the port indicated by port_id in LoMAC of destination IP.
- If destination IP is locally connected and the destination MAC is 0000.0000.0000, rewrite the destination MAC to the AMAC by matching the destination IP forward it to the port indicated by port_id of the LoMAC of destination IP.
- If the destination IP is not locally connected, rewrite the destination MAC to the LoMAC of next service function in the SFC and forward it to any spine using ECMP.

When packets come from uplinks, i.e., from spines, a leaf switch performs following actions:

- If packets come from an uplink, rewrite the destination MAC to the AMAC according to the LoMAC, and forward it to the host.

Packets with no match are discarded by default.

5.2 Spine operation

A spine switch forwards aggregated packet flows. There are no uplinks on a spine switch. The forwarding rules are defined based on a simple principle:

- Check the location in the destination MAC addresses of the incoming packets, forward the packets to the destination leaf or edge switch.

Thus, a spine can be SDN or non-SDN configured.

5.3 Edge operation

An edge switch is responsible for exchanging packets between S-Fabric enabled data centers or between an S-Fabric data center and a traditional data center where VLANs are used. When packets come from a DCI link, an edge switch performs following actions:

- Match the destination IP address, rewrite the destination MAC to the LoMAC of next service function in the SFC and forward it to any spine using ECMP.

When packets come from an uplink, an edge switch performs following actions:

- Match the destination IP address, rewrite the destination MAC to AMAC of the destination host and forward packets to destination DCI port.

Packets with no match are discarded by default.

5.4 Cross data center traffic

Figure 3 shows a scenario that a host A in data center 1 wants to reach host B in data center 2.

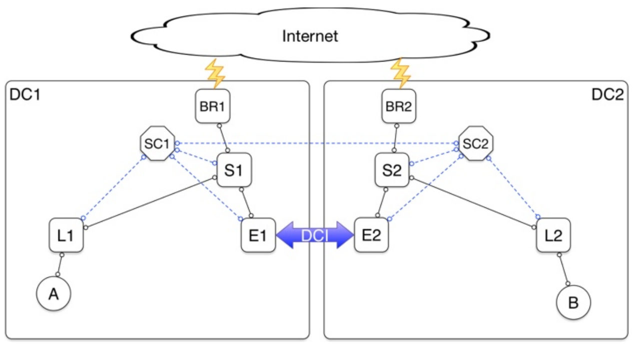

Figure 3. Example of traffic across DCI

In case host A and B are within the same subnet, A sends out an ARP request for B's MAC address. Host A receives AMAC of the DCI port of edge switch E2 as a response from SC1. A then sends packets to L1 with destination MAC as in the ARP response.

In case host A and B are in different subnets, A sends packets with destination IP as host B and destination MAC of 0000.0000.0000 to L1.

Upon receiving packets with destination IP as host B, L1 rewrites the destination MAC to the LoMAC of the DCI port of E1 and forward the packets to spines. E1 rewrites the destination MAC to the AMAC of DCI port of E2 and forwards the packets to E2. E2 rewrites the destination MAC to LoMAC of host B and forwards the packets to spines. L2 receives the packets, rewrites the destination MAC to AMAC of host B and forwards packets to B.

6. CONCLUSION AND FUTURE WORK

In this paper, we presented S-Fabric – a novel data center network design, which is able to provide a flexible, scalable, elastic, secure, robust, backwards compatible, and end host transparent network environment, to achieve an incremental SDN deployment in data center. The S-Fabric architecture fits not only to cloud provider data center networks, but also to a wide range of use cases such as Fog Computing, 5G, Internet of Things (IoT), Industrie 4.0 etc.

In future, we plan to work on implementing our design on a real data center network. This should help us evaluate its performance overhead.

7. ACKNOWLEDGEMENT

This work has been fully funded by the German Federal Ministry of Education and Research (BMBF) in the Celtic-Plus program SENDATE under contract 16KIS0483 and 16KIS0479 (cluster Secure-DCI).

8. REFERENCES

[1] Kreutz, D., Ramos,F.M.V., Veríssimo,P. E., Rothenberg, C. E., Azodolmolky, S. and Uhlig, S. 2015. Software-Defined Networking: A Comprehensive Survey, in *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, Jan. 2015. DOI= https://doi.org/10.1109/JPROC.2014.2371999.

[2] Greenberg, A., Lahiri, P., Maltz, D. A., Patel, P., & Sengupta, S. (n.d.). 2008. Towards a Next Generation Data Center Architecture: Scalability and Commoditization, In *Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow* (PRESTO '08). ACM, New York, NY, USA, 57-62. DOI= http://dx.doi.org/10.1145/1397718.1397732.

[3] Greenberg, A., Maltz, D. A., & Hamilton, J. R. 2009. VL2 : A Scalable and Flexible Data Center Network, In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication* (SIGCOMM '09). ACM, New York, NY, USA, 51-62. DOI= http://dx.doi.org/10.1145/1592568.1592576.

[4] Mysore, R. N., Pamboris, A., Farrington, N., Huang, N., Miri, P., Radhakrishnan, S., … Vahdat, A. (n.d.). 2009. PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication* (SIGCOMM '09). ACM, New York, NY, USA, 39-50. DOI= http://dx.doi.org/10.1145/1592568.1592575.

[5] Mann, V., Vishnoi, A., Kannan, K., & Kalyanaraman, S. 2012. CrossRoads: Seamless VM mobility across data centers through software defined networking. In *Proceedings of the IEEE Network Operations and Management Symposium(NOMS 2012)*, IEEE, 88–96. DOI= https://doi.org/10.1109/NOMS.2012.6211886.

[6] Comer, D., Karandikar, R. H., & Rastegarnia, A. 2017. DCnet: A new data center network architecture. In *Proceedings of the IEEE 7th Annual Computing and Communication Workshop and Conference* (CCWC '7), Las Vegas, NV, 1–6. DOI= https://doi.org/10.1109/CCWC.2017.7868382.

[7] Chen, C., Li, D., & Li, J. 2016. SVDC: A Highly Scalable Isolation Architecture for Virtualized Layer-2 Data Center Networks, in *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1-1. DOI= https://doi.org/10.1109/TCC.2016.2586047.

[8] Sherwood, R., Gibb, G., Yap, K., Appenzeller, G., Casado, M., Mckeown, N., … Appenzeller, G. 2009. FlowVisor : A Network Virtualization Layer.

[9] Kumar, S., Tufail, M., Majee, S., Captari, C., Homma, S. 2017. Service Function Chaining Use Cases In Data Centers, Internet-Draft, draft-ietf-sfc-dc-use-cases-06.