A SDN Based Scalable Network Across Data Centers

Yanwei Xu
Shanghai Engineering Research Center for Broadband Networks and Applications
Shanghai, China
ywxu@bnc.org.cn

Xiaoyuan Lu
Shanghai Engineering Research Center for Broadband Networks and Applications
Shanghai, China
xylu@bnc.org.cn

***Abstract*—The demands for the network across data centers are becoming more and more intense recently. However, there are three problems need to be solved, i.e., the ever-increasing number of actual or virtual machines demands, effectively manage these different machines and maintain the effectiveness. A scalable network across data centers is a very important technology to solve them. This paper proposes a new software defined network (SDN) based scalable network framework and its significant characteristic is to build a scalable layer 2/3 network that across data centers over the internet by using SDN's technologies. Our framework has high performances because: 1) The operation of encapsulating and unpacking packets is not necessary; 2) Routing paths can be pre-computed; 3) Traffic engineering to be implemented.**

Keywords—Testbeds, SDN, OpenFlow, Scalable Layer 2/3

I. INTRODUCTION

For more than 20 years we have been using Layer 3 connectivity powered by dynamic routing protocols to route traffic between data centers, but adoption of virtualization and geo-clustering technologies is forcing us to re-examine our data center interconnect (DCI) models. Harnessing the power of virtualization allows organizations to view and treat their compute resources as a global resource pool unconstrained by individual data center boundaries. Resources can span multiple buildings, metro areas or theoretically even countries. This basically means you can increase collective compute power in a data center that needs it by "borrowing" the resources from a data center that has spare capacity at the moment. This task is achieved by moving virtual machines between the data centers. All major virtualization vendors, such as VMware, Xen and Microsoft support the concept of virtual machine live migration, where you can move live VMs from one physical host (server) to another without powering them down or suffering application connectivity break (there is a short pause, but not long enough for TCP sessions to be torn down). Now the question is, what happens to the network settings -- specifically IP address/Subnet Mask/Default Gateway -- of the VM when it moves from Data Center A to Data Center B? The answer is they remain the same. Well, to be precise, they remain the same when live migration is performed. If, however, the VM is powered down in Data Center A, copied in the down state to Data Center B and then powered up, the server administrator will have to change the IP address on the operating system running inside the VM to match the settings required at the destination Data Center B.

On the network level, now both source and destination data centers need to accommodate the same IP subnets where VMs are located. Traditionally having the same IP subnet appear in both data centers would generally be considered a misconfiguration or a really bad design. Consequently, it also means that Layer 2, aka VLANs, need to be extended between these data centers and this constitutes a major change in the way traditional data center interconnection had been done. Although this explicit need for large-scale testbeds, three critical questions are posed:

- How do we deal with the ever-increasing number of actual or virtual machines demands in large-scale testbeds?
- How do we provide a common virtual management platform for different large-scale testbeds that own different machines?
- How can we maintain the effectiveness of large-scale testbeds?

A scalable layer 2 network that may across the entire Internet will be an effective answer to these problems. Scalable layer 2 network architecture is derived from the need for data center service requirements. And it introduces the flexibility for allowing the insertion or movement of a device in a transparent fashion from the perspective of the IP layer. Virtualization technologies not only increase the density of managed virtual servers in the data center, but also exacerbating the need for scalable layer 2 network.

Software defined network (SDN) [5, 6, 10] may be another effective answer to solve the problem of large-scale network across data centers. SDN is an emerging paradigm in computer networking that allows a logically centralized software program to control the behavior of an entire network. But a pure SDN technology is still difficult to solve those problems mentioned above, the reasons are: (1) the configurations of controller in a pure SDN is very high, thus it is difficult to provide excellent performance for large-scale networks; (2) its scalability will become very poor due to the restrictions of flow table entries.

In order to integrate their respective advantages of scalable layer 2 network and SDN. This paper proposes a new SDN based scalable layer 2 network framework, which has three functional modules: fabric construction, broadcast tree creation and traffic engineering.

This paper is organized as follows: related works are described in section 2. In section 3, The proposed framework and problem definition are presented. Finally, concluding remarks are given in section 4.

II. RELATED WORK

Software-defined networking (SDN) is an approach to computer networking which evolved from work done at UC Berkeley and Stanford University around 2008 [10]. SDN allows network administrators to manage network services through abstraction of lower level functionality. This is done by decoupling the system that makes decisions about where traffic is sent (the control plane) from the underlying systems that forwards traffic to the selected destination (the data plane). The inventors and vendors of these systems claim that this simplifies networking [6]. When a packet arrives at a switch in a conventional network, rules built into the switch's proprietary firmware tell the switch where to forward the packet. The switch sends every packet going to the same destination along the same path - and treats all the packets the exact same way.

SDN requires some methods for the control plane to communicate with the data plane. One such mechanism, Open-Flow, is often misunderstood to be equivalent to SDN, but other mechanisms could also fit into the concept. The Open Networking Foundation was founded to promote SDN and Open-Flow as marketing using the term cloud computing became popular. SDN has attracted lots of efforts from both the research and industry community [4,6,10] after it was proposed.

In the existing SDN studies such as [2], the Controller is responsible for all the communications in the network: processing every packet-in messages and calculating the routing paths for each time of data transmission. This operation manner brings an extremely heavy burden onto the Controller, which highly challenges its efficiency and limits the scalability of SDN in practical networks [10]. This paper employs a totally different operation style for the Controller which highly releases its workload and solves the scalability problem:

- The routing paths of each pair of switches in the network are pre-computed at the booting up stage of the network;
- The ARP requesting messages can be resolved both by the Controller and the broadcasting in the networks, in a load balancing manner.

III. PROBLEM DEFINITION AND THE FRAMEWORK

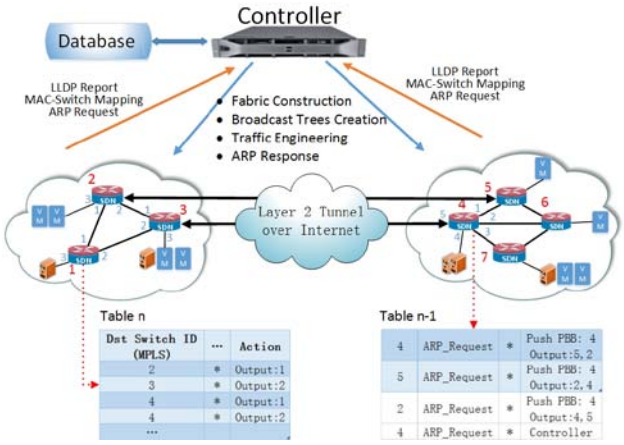

Fig. 1. Framework of the Proposed Network Across Data Centers

AS shown in Fig.1, our network across data centers is constituted by a number of sub-networks. Each sub-network contains a set of OpenFlow switches, each of which is linked with some hosts (physical machines or virtual machines (VM)). Some pairs of switches of different sub-networks have established layer 2 network tunnels (such as the GRE tunnel for connecting sub-networks to form a large-scale layer 2 network. The core of the network is a SDN Controller, which organizes the sub-networks to be a single fabric.

In Fig.1, the network has two sub-networks, where the red numbers indicate the switch IDs and the blue ones are the switch port numbers of each link. There are two tunnels between the two pairs of switches of the two sub-networks.

The main characteristic of the proposed method are as follows:

- All the switches use the LLDP protocol to find their neighbourhoods, the link attributes and the attached hosts, and then report them to the Controller. Hence, the Controller has the full topology of the whole network, which is used for fabric construction and broadcast tree creation at the network booting up stage.
- In each time of unicasting, the ARP requesting packet can be processed by the Controller and the broadcasting in the networks, in a load balancing manner. The ARP responsing message not only contains the MAC address of the destination host, but also the access switch ID of the destination host. The queried switch ID is used to tag the data packets by the access switch of the source host, and then the packets are routed using the destination switch ID by the flow table entries created in the booting up stage.

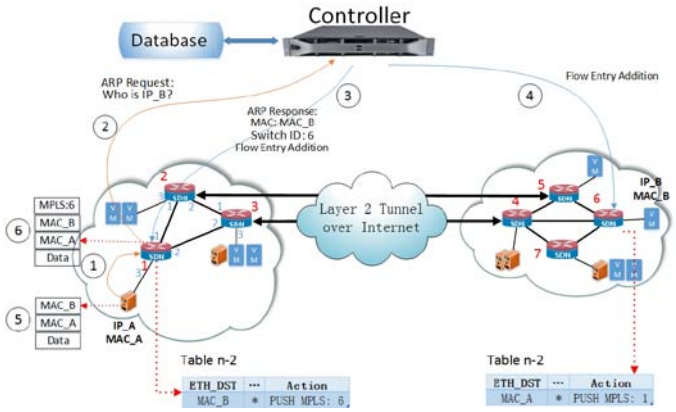

Fig. 2. Unicasting Process

Figure 2 depicts the process of a unicasting. When a host *A* wants communicate with a host *B* at the first time, it send an ARP request message to its access switch. If *B* is directly connected to A's access switch, the ARP request is resolved in the traditional manner; otherwise, the switch will forward the ARP request message to the Controller or the other switches, determined by the corresponding priorities of the flow entries.

If the Controller received the ARP request message, it will search the MAC address of *B* and the ID of its access switch in

its database that storing the topology. And then, the Controller sends an ARP responding message to *A* and two OpenFlow messages to the two access switches of *A* and *B* (switch 1 and 6) for loading the two flow entries as shown in the bottom of Fig.2, whose purposes are to tag the data packets using the destination switch IDs. Note that such flow entries must be stored in the flow table prior to the table for storing the entries that are created in the booting up stage.

After the ARP resolving is finished, the data packets will be tagged by the destination switch by the access switch of the source host. And then, the data packets are forwarded in the switch overlay based on the destination switch ID.

IV. Experiments

We build a testbed running on the OVS [3] based platform to evaluate the efficiency of the proposed network. And the experimental results are mainly focused on the average latencies of establishing a communicating path between two hosts and the end-to-end round-trip time of transmitting a packet.

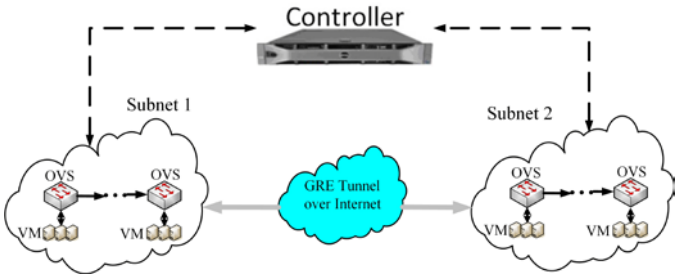

Fig. 3. Testbed of the Experiments

In the experiments, two merits are used to validate the efficiency of the proposed method:

- PCT (Path Construction Time): the time needed for constructing the communicating paths for two hosts.
- RTT (End-to-End Round-Trip Time): the time used for transmitting a packet between two hosts.

Because the number of switches along a path will affect the values of PCT and RTT, for simply focusing on the results of PCT and RTT but without generality, the topology of the testbed is linear, as shown in Fig.3. The testbed is composed of two subnets of two universities in Shanghai, which are connected by the GRE layer 2 tunnel across the public Internet. Each subnet contains at most 100 OVS bridges created by Mininet [1] using the linear topology. The GRE tunnel is created in the two right-most and left-most switches of the two subnets. Each subnet is running on a PC server (Lenovo Think Server R630, CPU is Intel Xeon E5-2630*2, RAM is 16G DDR3-1333). We implement the proposed method based on the OpenDayLight [2] platform. And in the experiments, two other technologies are compared:

- STP: the traditional spanning-tree protocols.
- ODL: the build-in L2 switching of OpenDayLight, which works as the most existing SDN studies that the Controller have to install OpenFlow table entries at every switch along the path between two communicating hosts.

And we use SFabric to indicate the proposed framework.

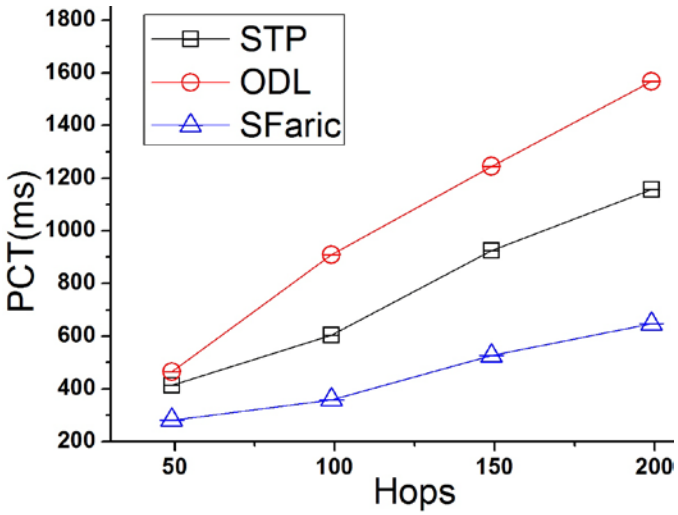

Fig. 4. The comparisons of the average PCTs among STP, ODL and SFabric with different hops

The PCTs, as shown in Fig.4, are obtained while five VMs of the first bridge (the leftmost) are communicating with other VMs connecting to the bridges which are 49, 99, 149 and 199 hops away, respectively. Since SFabric has finished the fabric construction and made every switch know the routes to other switches after network booting up, the Controller just needs to install two flow table entries into the first and last switches while creating a path for a pair of hosts. Compared with other existing OpenFlow solutions that should install the flow table entry into every switch along that path, the path construction time (PCT), mainly spent on ARP resolving and the process of installing the flow table entries, are highly reduced. The PCT of ODL is the longest since it should install the flow table entry into every switch along a path, which is very time-consuming. The PCTs of STP are in the middle due to its broadcasting manner in ARP resolving. Lastly, SFabric just needs to install two flow entries into the first and last switches of a path, so its PCT is the shortest.

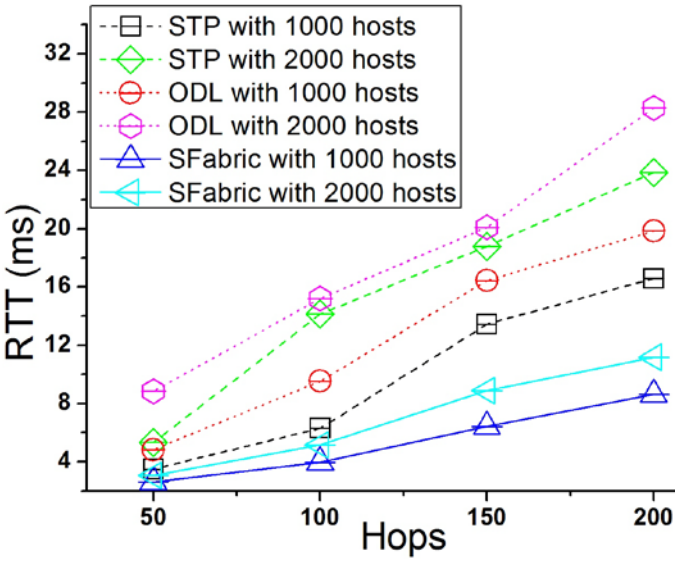

Fig. 5. The comparisons of RTTs among STP, ODL and SFabric with different numbers of hosts

Since the amount of hosts in a network will affect the RTT, we set a scenario where various numbers of hosts exist in a certain subnetwork (i.e., SFabric with 1000 hosts in Fig.5 means there are 1000 hosts exist between the 1st OVS and another OVS when SFabric is used) in order to observe the disparities among the RTTs of STP, ODL and SFabric. From Fig.5 we can see that the RTTs of SFabric with different hosts are very close whereas the RTTs of ODL are much different from each other, which prove the delay influence caused by the number of hosts is limited to SFabric but serious to ODL. This is because the reduction of flow table entries since the time used for finding out the right output at each switch is much less than ODL. Furthermore, as the hosts or switches increase, the gaps among the RTTs of those three solutions are also enlarged correspondingly.

V. Conclusions

In this paper, we propose a large layer 2 network based on SDN technologies for interconnecting data centers. A central controller is used to manage the routing maps among the switches. But different from other SDN solutions, it employs a totally different operation style for the Controller, which highly releases its workload, forwards flows more efficiently and solves the scalability problem.

In the future, we will test our framework in the real network and promote its performance. And the traffic engineering components will be developed and studied in the future. Furthermore, the ARP resolving is only done by the Controller, which will pose a heavy burden to the Controller. How to relieve such burdens would be another future study direction.

Acknowledgment

We would like to thank the 863 plan project under the grant No 2013AA013505.

References

[1] Mininet. http://mininet.org.

[2] OpenDayLight.http://OpenDayLight.org.

[3] OpenVSwitch. http://openvswitch.org.

[4] Programmable flow networking. http://www.necam.com/SDN/.

[5] O.N. Foundation. Software-Defined Networking: The New Norm for Networks. White paper. 2013.

[6] S.Jain, A.Kumar, S.Mandal, J.Ong, L.Poutievski, A.Singh, S.Venkata, J.Wanderer, J.Zhou, M.Zhu, J.Zolla, U.Holzle, S.Stuart, and A.Vahdat. B4: experience with a globally-deployed software defined wan. In ACM SIGCOMM 2013 Conference, SIGCOMM'13, Hong Kong, China, August 12-16, 2013, pages 3--14, 2013.

[7] N.McKeown, T.Anderson, H.Balakrishnan, G.M. Parulkar, L.L. Peterson, J.Rexford, S.Shenker, and J.S. Turner. Openflow: enabling innovation in campus networks. Computer Communication Review, 38(2):69--74, 2008.

[8] A.Voellmy and J.Wang. Scalable software defined network controllers. In ACM SIGCOMM 2012 Conference, SIGCOMM'12, Helsinki, Finland - August 13 - 17, 2012, pages 289--290, 2012.

[9] A.Voellmy, J.Wang, Y.R. Yang, B.Ford, and P.Hudak. Maple: simplifying SDN programming using algorithmic policies. In ACM SIGCOMM 2013 Conference, pages 87--98, 2013.

[10] Y.Zhuang, L.Law, A.Rafetseder, L.Wang, I.Beschastnikh, and J.Cappos. Sensibility testbed: An internet-wide cloud platform for programmable exploration of mobile devices. In 2014 Proceedings IEEE INFOCOM Workshops, Toronto, ON, Canada, April 27 - May 2, 2014, pages 139-140, 2014