WAN Virtualization and Dynamic End-to-End Bandwidth Provisioning Using SDN

Adrian Lara[1], Byrav Ramamurthy[1], Eric Pouyoul[2] and Inder Monga[2]

[1] *University of Nebraska-Lincoln, Lincoln NE 68504*
{*alara,byrav*}*@cse.unl.edu*

[2] *Energy Science Network, Lawrence Berkeley National Laboratory, Berkeley CA 94720*
{*lomax,imonga*}*@es.net*

Abstract: We evaluate a WAN-virtualization framework in terms of delay and scalability and demonstrate that adding a virtual layer between the physical topology and the end-user brings significant advantages and tolerable delays.

OCIS codes: 060.0060, 200.0200.

1. Introduction

Current science projects rely on large-scale collaboration and data transfers. The need for high-speed networking has enabled innovation on high-bandwidth, multi-domain and multi-layer networking. Research and education organizations such as ESnet and Internet2 are currently capable of transmitting data at 100 gigabits per second and are aiming at 400 Gbps across their wide area network (WAN) [1, 2]. In fact, ESnet has been a pioneer in multi-domain bandwidth provisioning with their production service based on On-Demand Secure Circuits and Advance Reservation System (OSCARS) software [2]. This reservation system allows end-users to reserve and provision virtual circuits with guaranteed bandwidth and latency between two edges of the WAN. Furthermore, ESnet recently demonstrated how OSCARS can achieve multi-layer provisioning and optimization [4].

One challenge that continues to exist is building a dynamic end-to-end circuit. By end-to-end, we refer to a circuit that goes all the way from a scientific processing unit to another, instead of a circuit that only traverses the WAN. The difficulty is to go from the WAN to the LAN dynamically and for end-sites to have control without being exposed to the WAN topological complexity introduced by slicing architectures [1]. To address this, Monga et al. [3] proposed OneSwitch, an experimental setup that leverages software defined networking (SDN) and OpenFlow to expose a simple, programmable virtual switch that can be used by the end-user to dynamically create end-to-end circuits. OneSwitch relies on OSCARS to send data across the WAN, but hides this from the external user while exposing the virtual switch abstraction.

In this paper, we first evaluate the performance of OneSwitch (section 3) with multi-tenant usage. We measure the delay introduced by OneSwitch to act as a virtual layer between the physical network and the end-user's OpenFlow controller. We also evaluate the scalability of the solution by investigating how many tenants can use instances of OneSwitch at the same time. Second, we propose several extensions to OneSwitch so that end-users can have a finer-grained control of the network (section 4). Currently, OneSwitch exposes a single layer-2, OpenFlow-compliant switch so that end-users can use an OpenFlow controller to program the instance of OneSwitch and dynamically create end-to-end connections [4]. We are interested in enhancing the capabilities of OneSwitch to create *WAN-specific* extensions to the OpenFlow protocol. We describe how these extensions to the protocol can enable OSCARS to perform multi-layer provisioning more efficiently. Finally, we draw our conclusions in section 5.

2. Background and related work

2.1. WAN virtualization using OneSwitch

OneSwitch [3] abstracts the entire topology of the WAN as a single switch, where each virtual port is mapped to a physical port of a physical switch at the edge of the network. OneSwitch can provide a virtual switch to each tenant. OneSwitch uses OpenFlow to control the switches at the edge of the WAN and hides from the end-user how data traverses the network. At the same time, OneSwitch itself is also OpenFlow-compliant, thus providing the end-user

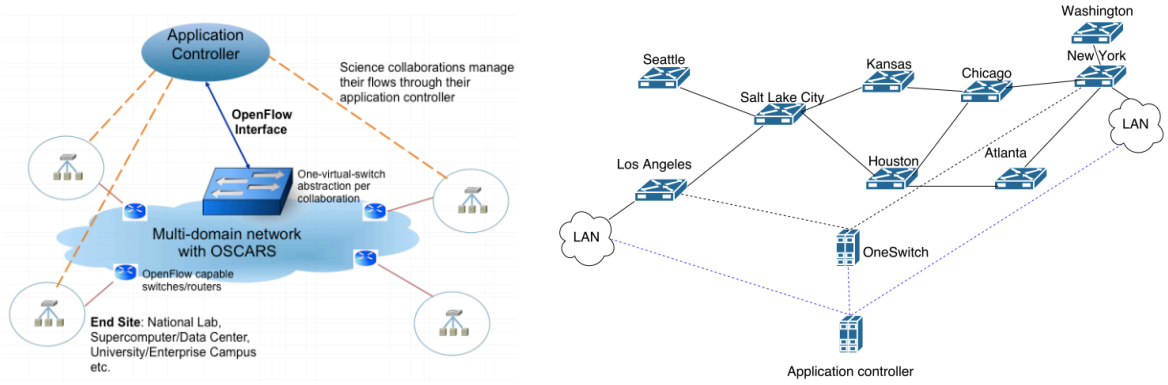

Fig. 1: (a) OneSwitch conceptual diagram, and (b) Experimental topology deployed in GENI.

application with a dynamic and programmable virtual switch. OneSwitch acts as a virtual layer to modify OpenFlow messages as required. The advantages of this model are outlined by Monga et al. [3]. In a nutshell, this abstraction is simple, atomic, programmable and network-wide.

3. OneSwitch performance evaluation

Each instance of OneSwitch acts as a virtual layer between the physical devices of the network provider and the virtual Ethernet switch programmed by the application controller. Naturally, this virtual layer adds a processing delay when translating OpenFlow messages coming from the physical switches to similar messages directed to the application controller.

In this section, we quantify these delays to determine how scalable OneSwitch is and what is the impact on the transmission times of having an intermediate virtual layer. We run our experiments using the GENI testbed and we use a topology based on Internet2's network. Figure 1b shows a WAN with nodes at Seattle, Los Angeles and several other cities. These devices are controlled by a WAN controller that acts as an instance of OneSwitch for multiple application controllers. Although our topology is more complex, with multiple nodes connected to all the edge switches of the WAN, in this figure we show a single application controller connected to OneSwitch and controlling LANs in Los Angeles and New York. To further clarify how OneSwitch works, this application controller is presented with a virtual switch that has two ports. By forwarding traffic from port one to port two, the application controller is successfully sending data from Los Angeles to New York.

Figure 2a shows the round-trip time needed to send packets with and without using OneSwitch. When OneSwitch is not used, we hard code the flow rule at the edge of the WAN so that traffic tagged with a given VLAN is forwarded to the destination. This is the traditional approach in which a network operator manually configures the edge switch of the LAN to forward packets to the WAN with a given VLAN. By using OneSwitch, the application controller can now automatically control both the LAN edge switch and the WAN edge switch. This adds a delay which is needed to push the flow rules during the first packets. However, notice that once the flows have been pushed, the performance is identical in both cases. Therefore, OneSwitch only adds a delay when the controller needs to handle packets often. In practice, few packets end up going to the controller since rules are pushed proactively. We argue that this delay is acceptable given the great benefit in terms of flexibility offered to the end-user.

Figure 2b shows that the delay added by OneSwitch is almost constant when the number of application controllers using instances of OneSwitch increases. OneSwitch handles each end-user application controller using different threads and is thus capable of handling multiple tenants at the same time. To test this, we do not add flow rules, forcing all traffic to go to the application controller. This greatly increases the amount of packets that must be translated by each instance of OneSwitch. Even when this scenario is unlikely, we can still see that OneSwitch scales well.

4. Improving OneSwitch to benefit the end-user and the network provider

ESnet, Infinera and Brocade recently demonstrated a multi-layer implementation of OSCARS using OpenFlow [4]. In that paper, OpenFlow is used to control both Ethernet devices and Infinera equipment at the WDM and OTN layers. Based on this, we argue that OneSwitch can be used to combine dynamic end-to-end circuit creation with multi-layer

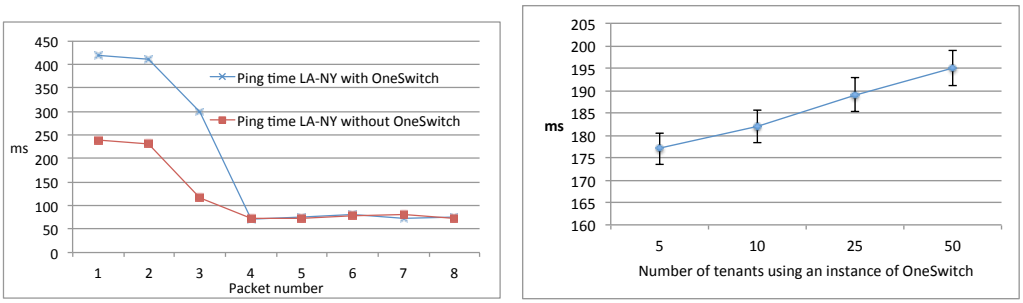

Fig. 2: (a) Round-trip time of packets with and without using OneSwitch, and (b) Delayed introduced by OneSwitch for varying number of tenants.

bandwidth provisioning. We aim at creating *WAN-specific* extensions to the OpenFlow protocol that allow end-users to have a finer-grained control of how the flows traverse the WAN. In the current implementation of OneSwitch, the end-user controls a virtual Ethernet switch using OpenFlow (in other words, the northbound API of OneSwitch is OpenFlow). Next we explain how this channel can be extended to allow the application controller for better network control.

OneSwitch can modify the *features* message used to notify the controller about current capabilities. In this message, a list of bandwidth capabilities can be advertised and the application controller can use an extended version of *packet out* or *flow mod* to request bandwidth guarantees for a given flow. The same thing works for latency. The application controller can request different latency requirements for each flow, based on application-level needs from the tenant.

The advantages for the network provider are also important. On the one hand, per-flow bandwidth requirements simplify network utilization. Currently, OSCARS faces high levels of fragmentation because reservations are made for long periods of time but are not used permanently by the tenants. By knowing the requirements of each flow, OSCARS can decide to send traffic through different routes in order to reduce fragmentation. On the other hand, user input is extremely valuable when performing multi-layer bandwidth provisioning. OSCARS needs a mechanism to ensure that the effort needed to implement a WDM-layer bypass is worth the time. When a user specifies low bandwidth requirements, an MPLS-based path can be used instead. However, when the user specifies that a high-bandwidth and low-latency path is needed, then a bypass through WDM or OTN is worth considering.

5. Conclusion

OneSwitch achieves topology virtualization and allows end-users to program the network without having access to the physical devices. By controlling the entire WAN as if it were a single Ethernet switch, an end-user application controller can create multiple paths between different locations with simple OpenFlow statements. Our evaluation of OneSwitch shows that the delay introduced by the virtual layer is small. It also shows that multiple instances of OneSwitch can co-exist such that several application controllers belonging to different tenants can program their own on-demand networks. We also proposed to extend the OpenFlow protocol between OneSwitch and the application controllers so that end-users have a finer-grained control of the network. The network provider also benefits from this additional input, as it can be used to perform multi-layer provisioning more effectively.

References

1. Internet2 AL2S, http://www.internet2.edu/products-services/advanced-networking/layer-2-services/
2. Energy Science Network. “On-Demand Secure Circuits and Advance Reservation System”, http://www.es.net/services/oscars/
3. I. Monga, E. Pouyoul, C. Guok, “Software Defined Networking for big-data science: Architectural models from campus to the WAN”, in Proceedings of the Supercomputing Conference SC, 2012.
4. H. Rodriguez, I. Monga, A. Sadasivarao , S. Sayed, C. Guok, E. Pouyoul, C. Liou, T. Rosing, “Traffic Optimization in Multi-Layered WANs using SDN”, Proceedings of IEEE Hot Interconnects, 2014, www.hoti.org.