Scaling-up Flow Path Computation for a Large Number of Virtual SDN/NFV Infrastructures

Hiroaki Yamanaka       Eiji Kawai       Shinji Shimojo
National Institute of Information and Communications Technology
Tokyo, Japan
{hyamanaka, eiji-ka, sshinji}@nict.go.jp

1. BACKGROUND

Wide-area SDN/NFV infrastructures (i.e., the OpenFlow networks with middleboxes) require high capital and operational expenditures (CAPEX and OPEX) due to long-haul links, and many middleboxes and switches. Sharing wide-area SDN/NFV infrastructures owned by an infrastructure provider (InP) with multiple application service providers (ASPs) improves the cost-effectiveness. ASPs control SDN/NFV infrastructures to provide communication services to their end-hosts. To enable ASPs to provide their services, ASPs need to control SDN/NFV infrastructures independently. Especially, ASPs dynamically select the sequence of the middleboxes, to which the traffic is steered, to realize the service chaining considering the load balance of the middleboxes. Hence, building isolated logical SDN/NFV infrastructures (i.e., virtualization of SDN/NFV infrastructures) is necessary.

In our assumed virtualization model, middleboxes implemented by virtual machines are allocated to unique ASPs while a physical OpenFlow network is virtualized (i.e., logically isolated, but physical links are shared) and virtual OpenFlow networks are provided to ASPs. On the virtual OpenFlow networks, ASPs set flows between middleboxes for communications of the end-hosts. An InP implements an OpenFlow network virtualization technique (e.g., AutoVFlow [1]) to translate messages from ASP controllers to handle flows in the physical OpenFlow network.

An InP dynamically selects paths for flow settings by ASPs to maximize the accommodated flows because an InP has an incentive to improve the cost effectiveness of its own physical infrastructure. An ASP views the OpenFlow switches connected to the middleboxes or the end-hosts, and the virtual links between the switches in the virtual OpenFlow network. An ASP sets flows to steer the traffic to the middleboxes to realize the service chaining. When an ASP sets a flow on the virtual link, an infrastructure owner computes a physical path between the switches based on the available bandwidth of the physical links. To accept many flow settings, an InP needs a scalable path computation method.

2. MOTIVATION

In existing distributed SDN controller platforms, the functionality of handling OpenFlow switches is distributed; however, path computation is implemented in a centralized manner. Thus, path computation is not scalable. Onix [2] distributes controller instances horizontally and vertically. An Onix instance at the lower layer handles the subset of switches and behaves as a single switch for an Onix instance at the upper layer. An upper Onix instance handles a global network, which has an abstracted topology shown by the lower layer instances, and computes paths in the network. Onix instances at the lower layer compute paths in the subset of switches. In the Onix architecture, computational load is concentrated on an Onix instance at the upper layer. ONOS [3] horizontally distributes controller instances while network applications run atop the ONOS controller instances. SDN applications directly manipulate the ONOS instance, hence, there is no centralized point to handle the network. However, ONOS itself does not support distributed path computation.

For improving the scalability of path computation, a data store that manages the available bandwidth of physical links is a bottleneck. Path computation cannot be implemented in parallel because even multiple computation instances need to lock the common data store. When computing paths on a global network, a path computation instance locks the data store of the available bandwidth of all physical links during computation to accurately find paths. As a result, even if there are multiple path computation instances, they cannot simultaneously compute paths. Note that an eventually consistent data store provides the high read/write performance by utilizing data store replications, however, a path computation instance may compute paths based on the incorrect available bandwidth.

3. PROPOSAL

Our proposal selects the data of the available bandwidth of the physical links to be locked to improve the scalability of path computation. We assume that the data store is strong consistent and it can be locked for the available bandwidth data of the individual physical links. Hence, path computation instances do not affect each other while they lock on the data of disjoint physical links. Each path computation instance computes physical paths for flows between the switches connected to end-hosts or middleboxes. When computing a path, the path computation instance selects a path from the candidate links, which are the part of the physical links being selected as a path with the high probability. If the candidate physical link is locked by another computation instance, the computation instance selects a path from the remaining physical links. Especially, in wide-area SDN/NFV infrastructures, most flows are for middleboxes in close locations. Hence, the overlapping candidate physical links for path computation instances, which compute paths between different switches, are small parts of all physical links. Conflicting of locks on the data by the
path computation instances has small impact for results of selected paths. Furthermore, our method can improve the scalability of path computation.

3.1 Architecture

The architecture is composed of the AutoVFlow proxies, the path computation instances, and the global strong consistent data store, as shown in Figure 1. The AutoVFlow proxies [1] behave as virtual OpenFlow switches for ASP controllers. They autonomously implement the virtualization while providing integrated virtual OpenFlow networks for the ASP controllers. When an ASP controller sets flow entries to the virtual OpenFlow switches through the AutoVFlow proxies, the path computation instances compute paths in the physical OpenFlow network for the flows. Then, the AutoVFlow proxies set flow entries to physical OpenFlow switches according to the path. During computation, the path computation instance locks the data of the part of the physical links and updates the available bandwidth data of the physical links of the computed path.

3.2 The data of the physical links to be locked

When computing a path between the switches that are connected to end-hosts or middleboxes, the path computation instance locks the data of physical links determined as follows and selects a path within the physical links. We assume that a short path composed of the links with the high link capacities is selected with the high probability by a path selection algorithm. Selecting a short path minimizes the bandwidth consumption in the whole network. Furthermore, selecting high bandwidth links with the high probability avoids to consume a low bandwidth link (i.e., to lose the path containing the low bandwidth link) within early time. We propose the method to find the physical links that would be selected in paths with the high probability.

The idea of the method is to find the switches, through which the paths between the start switch $s$ and the goal switch $g$ frequently go, and extracts the physical links around the found switches. Let $C = (c_{ij})(1 \leq i, j \leq n)$ be the transition probability matrix, where $n$ is the number of the all physical switches. $c_{ij}(i \neq j)$ is proportional to the bandwidth capacities of the links $ij(1 \leq j \leq n, i \neq j)$ and $c_{ii} = 0$, and $\sum_{1 \leq j \leq n} c_{ij} = 1$. Path selection, which chooses the next switch in a path based on the probability that is proportional to the bandwidth of the links to the adjacent switches, can be modeled as random walk on the Markov chain $C$. Let $\pi = (\pi_1, \ldots, \pi_n)$ be the stationary distribution of $C$, then $\pi_i$ can be interpreted as the probability that the switch $i$ is selected at the arbitrary time. For switches $s$ and $g$, the method identifies the switches, whose probabilities in the stationary distribution are in top $\epsilon \%$ of the switches within the distance $d$ from switch $s$ or $g$ in the topology of the physical OpenFlow network. The physical links in the shortest paths from the identified switches to switches $s$ and $g$ are determined to be the physical links. The shortest path is on the weighted graph of the physical OpenFlow network, where the weight is the reciprocal number of the bandwidth capacity of the physical link. The physical links are pre-determined, and the path computation instances lock the data of the determined physical links when computing a path between switches $s$ and $g$.

4. Demonstration

In the demonstration, we will show that our proposal improves the scalability of path computation. We prepare two networks of the same topology and the bandwidth capacity of the links (Figure 2). For one network (i), multiple path computation instances compute paths with locking the data of the all physical links. For the other network (ii), multiple path computation instances compute paths with locking the data of the selected physical links determined by our method. On each network, there are two virtual SDN/NFV infrastructures. The ASP controller frequently sets flows in a virtual OpenFlow network. In the other virtual OpenFlow network, we run a streaming application, and the other ASP controller sets flows for the streaming traffic. We will observe that path computation instances wait (i.e., the data of all physical links is locked) for others many times in the network (i). On the other hand, path computation instances less wait in the network (ii).

5. References

