Large Flow Detection and Delay Measuring for Multipath Routing over SDN

Hnin Thiri Zaw, Aung Htein Maw
University of Computer Studies
Yangon, Myanmar
h.thirizawucsy@ucsy.edu.mm, ahmaw@ucsy.edu.mm

Abstract

The traditional single path routing can cause imbalanced link utilization and is not efficient for all traffic types such as long-lived large flows. Moreover, it can lead to low network throughput and high network latency. Traffic engineering (TE) is a key solution to solve these problems of single path. The main purpose of TE is to optimize the network resource utilization and improve network performance by measuring and controlling network traffic. One of the TE approach for large flows is multipath routing which distribute traffic load among available multiple paths. To be intelligent multipath routing based on traffic types, we also need to detect large flow in network. In this paper, we present two main folds: (1) large flow detection approach by using sFlow analyzer in real time, and (2) measuring end-to-end delays of available paths between source node and destination node where large flow occurred in SDN environment.

Keywords*: Traffic Engineering (TE), Multipath, sFlow, End-to-end delay.*

1. Introduction

Nowadays, traditional network infrastructures are more and more complex as the growth of networking scale rapidly. Generally, current network devices can also be seen as two logical layers: control plane to make decisions according to set of rules and data plane to forward packet only based on decisions of control plane. Coupling between these decision control plane and forwarding data plane makes inflexibility to maintain the traditional packet forwarding devices such as routers, switches, access points and so on.

To overcome the limitations of current network infrastructure, researchers has been approached to Software Defined Network (SDN) technology for future network rapid innovation. As shown in Figure 1, the main concept of SDN is to separate the control plane from network devices and physically place it as an external entity which is called a controller which has the eyes of global network view as a graph. OpenFlow[1] is the first communication interface to exchange data between controller and forwarding devices in SDN architecture. It allows direct access to and manipulation of forwarding device for both virtual (hypervisor-based) and physical. The controller runs the network operating system and many networking applications such as load-balancer, firewall, policy management and network management applications can be developed over it. Therefore controller can make decisions centrally for all network devices.

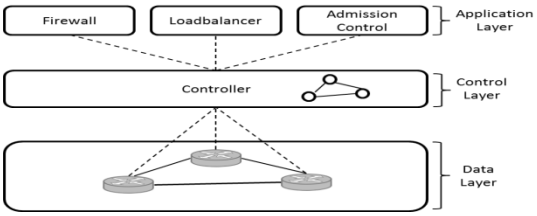

Figure 1. SDN Architecture

The main process of multipath routing is to split the received packets of same flow through a set of forwarding paths. In wired network, traffic splitting can also applied by using intermediate router which can be any node along a path or several network interfaces. The advantages of multipath transferring can balance the traffic load and hence (i) avoid congestion in bottleneck nodes and (ii) makes efficient use of underutilized path which is caused by traditional single path routing. Therefore, multipath approach is very useful and important for long-lived large flow in network. Although it has many benefits, there are still certain challenges in parallel transferring the packets of the same flow over multi path because of different delivery time to from one source to destination. It can also be called different end-to-end delays. As a consequence, delay variations (jitters) becoming increased and degrade

the quality of service for delay sensitive real-time traffic (e.g. video conferencing, online gaming and so on). Therefore, not only flow characteristics such as packet arrival rate but also end-to-end path delay time should take account into multipath routing based on its algorithm objectives. So, detection of large flow in network and end-to-end delay metrics plays important role to be efficient traffic engineering.

In this paper, we propose large flow detection in SDN environment by using sFlow-RT analyzer and estimation of end-to-end delay for data plane. Our simulation is based on ONOS as a controller and Mininet emulator.

The remainder of this paper is organized as follows. Section 2 presents related work overview. Section 3 gives the explanation about sFlow-RT[5] analyzer. In Section 4, delay measurement model is presented. Section 5 describes experiment scenario of large flow detection and delay measuring. Section 6 presents conclusion and future works.

2. Related Work

Some related works proposed multipath routing methods in SDN environment are as follows.

The authors of [2] presented multipath with load balancing and admission control, which differs from our approach is that their controller perform traffic statistics monitoring in every 5 seconds and our monitoring analyzer works in every 1 second. Another work of flow splitting in OpenFlow network [3] was proposed three different load dependent multipath models. Their proposed methods are used when traffic overload, otherwise traffic is sent over single primary path, which differs from our approach is that it uses OpenFlow features for traffic monitoring and dynamic flow splitting and we use sFlow-RT analyzer to detect large flow in real time. Design and functionality test of chunked video streaming over emulated multi-path OpenFlow network was proposed in [4]. The authors presented method to receive video streaming packets from ongoing multipath at the egress Open vSwitch before receiver by using middle box to reorder packets, which differs from our approach is that we use to differentiate large flow in network to apply multipath routing. These proposed system were described above are not consider end-to-end path delay cost metric, which is contrast to our research work. Large flow detection and delay measuring method are mainly focused in this paper to apply in multipath routing.

3. sFlow-RT Analytics Engine

sFlow is a real time traffic analyzer for Software-Defined Networking (SDN). It can be used to monitor network traffic in both physical and virtual devices (eg. Open vSwitch)[5]. sFlow uses sampling technology to analyze traffic statistics and based on collector and agent architecture in Figure 2. In this figure, sFlow collector receives a continuous stream of sFlow datagrams from its agents which are embedded in devices periodically. Then, the collector analyze the statistics of traffic and converts them into metrics which are specified in keys of flow definition by user and can be queried using REST API which can be accessible by any language such as JavaScript and Python. sFlow-RT application (eg. large flow detection and DDOS mitigation) can be developed as external application by using REST API or embedded application by using JavaScript API. The output metrics are represented by JSON format that is human-readable text to transmit data objects which consisting of attribute-value pairs.

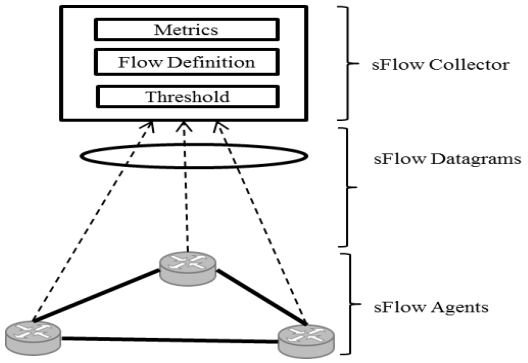

Figure 2. sFlow Architecture

4. Measuring End-to-End Delay

End-to-End delay refers a time taken for a packet travelling across from one source end to another destination end in the network. It consists of nodal processing delay, queueing delay, transmission delay and propagation delay. For multipath routing, difference end-to-end delays among multiple paths can lead to increase delay variations (jitter) in packets and degrade QoS performance for delay-sensitive real-time applications. Therefore, to be efficient multipath approach, end-to-end delays of multiple paths are needed to measure before splitting traffic load.

In our delay measurement, we measure delays of paths by sending out probe packets. According to figure 3, Total delay (T_{Totoal}) calculation

is applied by generating probe packet from ONOS controller. The first probe packet (faked source MAC and faked destination MAC) is sent through source switch to destination switch along the path and back to controller. Packet arrival time ($T_{arrival}$) and sent time (T_{sent}) from this first packet can be learned. Then another two probe packets are generated from controller to source switch and destination switch respectively. From these two packets, round-trip-time of controller to switches can be found. Therefore, the equation for total end-to-end delay cost can be derived following:

$$T_{End-to-End} = T_{arrival} - T_{sent} - \frac{T_{ONOS-to-S1}}{2} - \frac{T_{ONOS-to-S2}}{2} \quad (1)$$

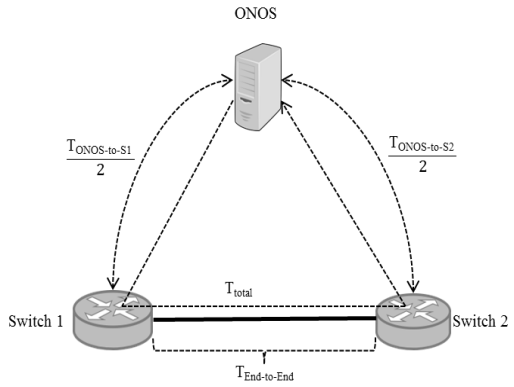

Figure 3. Delay Measurement Architecture

5. Experimental Setup

In our testbed, we use ONOS controller (version 1.8) among other kinds of SDN controllers (eg.NOX, POX, Ryu, FloodLight) because of its performance, high level abstractions and API. ONOS is distributed system which is designed for scalability and high availability. It serves as network operating system with separation of control and data plane for service provider network. Our topology is created by using Mininet emulator (version 2.2.1) which can create virtual network and provide hundreds and even thousands of virtual hosts, and OpenFlow (version 1.0). Two virtual machines is used for our testbed, sFlow analyzer and Mininet topology is running on one VM and ONOS controller is running on another. We use two experimental testbeds to demonstrate and evaluate results. One is leaf and spine topology in figure 4 to show sFlow-rt analyzer configuration. Another testbed in figure 8 is to evaluate results of delay measurement module.

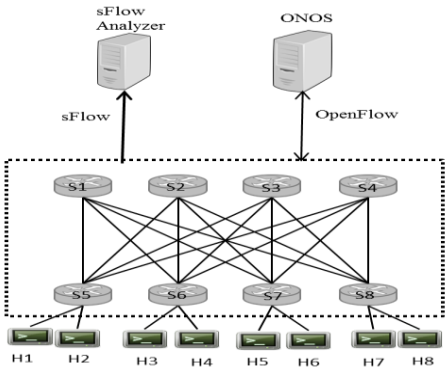

Figure 4. Leaf and spine topology(Testbed 1)

Leaf-spine topology is based on two-tier topology which is mostly used in data center infrastructure. The leaf layer includes switches to provide connectivity of end devices. The spine layer provides connectivity of leaf switches. In our leaf-spine testbed topology in figure 4, we use 8 switches and 8 hosts. Bandwidths of all links in this network topology are set 10 Mbps. ONOS controller ran on a laptop powered by Core i5-5200U CPU @ 2.20GHZ with RAM 4GB. Mininet and sFlow ran on Laptop PC powered by Core i5-5200U CPU @ 2.20GHZ with RAM 4GB.

```
def configSFlow(spine, leaf, collector,
ifname):
    sflow = 'ovs-vsctl -- --id=@sflow
create sflow agent=%s target=%s
sampling=10 polling=20 --' % (ifname,
collector)
    for s in range(1,spine+1):
      sflow += ' -- set bridge s%s
sflow=@sflow' % s
    for ls in range(1,leaf+1):
      sflow += ' -- set bridge s%s
sflow=@sflow' %       (spine+ls)
```

Figure 5. sFlow agents configuration

```
setFlow('pair',{'keys':'macsource,macdes
tination,ipsource,ipdestination,tcpsource
port,tcpdestinationport,link:inputifindex,
link:outputifindex','value':'bytes'});
setThreshold('elephant',{'metric':'pair','v
alue':1000000/8,'byFlow':true,'timeout':1
```

Figure 6. Defining flow and threshold

For sFlow analyzer, we configure sFlow agents on all open vswitches in topology according to figure 5. In this figure, interface.

Name (ifname) of switch is used as sflow agent address and target address is sflow collector address. As we set link bandwidth 10Mbps, sampling rate is 1 in 10 packets and polling interval is 20 seconds. Then, we define flows in our large flow detection script as presented in figure 6 which are used to match packets that share common attributes. In flow definition, a flow called pair that captures MAC addresses, IP addresses, TCP ports, interface indexes, and calculate bytes per second for each flow. After that, we also define threshold which is applied to metrics. We set threshold value 1MB in this script. When the rate value of a flow exceeds the threshold, it will notify as 'elephant' flow. Our ONOS application can access JSON output from sFlow analyzer by calling REST API: /activeflows/{agent}/pair/json in every one second. This REST API list top active flows, and remove all duplicates for flows reported by sources. In our experiment, we use iperf tool as traffic generator for TCP flow.

Delay measurement module is implemented in our ONOS application. Here, we need to find a set of available shortest paths before measuring delays. In ONOS, we can use topology service to find shortest paths which uses *Dijkstra algorithm*. It represents the available shortest paths as a set of links. After that, the delay measurement module is responsible for measuring delays of available paths between source and destination nodes where is large flow occurred. As soon as sFlow notifies about large flow, it sends probe packets with fake MAC source and destination addresses to each path as described section 4. Firstly, this module send probe to source switch, then another probe is sent through along the path, and final probe is sent to destination switch. Sent-time for every packet is encapsulated in each payload and receive-time is when the controller receives it back. Actual delay is calculated by using equation (1).

Figure 7. Large flow notification from sFlow

We generated TCP large flow from H8 (IP address: 10.0.3.1) to H2 (IP address: 10.0.0.2) by using iperf. At that time, sFlow detects large flow from H8 to H2 and notify this event in figure 7 which presents MAC addresses, IP addresses, TCP ports and interface indexes. This information is represented in JSON format via REST API. ONOS application is reading this REST API in every one second. As soon as ONOS application gets JSON from sFlow, it finds available shortest paths between H8 and H2. By using these shortest paths, delay measurement module send probe packets to each path. The results of delay measurement module are evaluated in section 5.1.

5.1 Experimental Results

We tested delay measurement results with different probe interval scenario. Probe intervals are 1000 ms and 500 ms. We also tested in different single path topologies: 4 switches, 8 switches and 12 switches as shown in figure 8.

(1) S1---S3---S4---S2
(2) S1---S3---S4---S5---S6---S7---S8---S2
(3) S1---S3---S4---S5---S6---S7---S8---S9---S10---S11---S12---S2

Figure 8. Single Path Topologies

All topologies have two hosts (H1 and H2) which are connected with S1 and S2 respectively. They are placed as udp server and clien. Iperf tool is used to generate udp traffic during 100 seconds via increasing bandwidth rates (20Mbps to 100Mbps). The results are showed by average delay and maximum delay. We showed average delay results as Test1 and maximum delay as Test2.

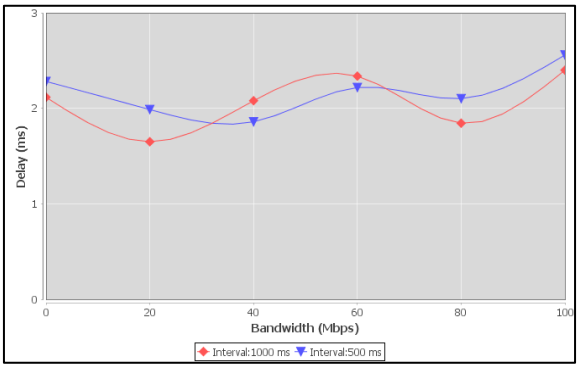

Figure 9. Average Delay for Hop Count: 4

According to Figure 9, 10 and 11 in Test1 case, there are no significant changes for average delays with different probe packet interval scenario.

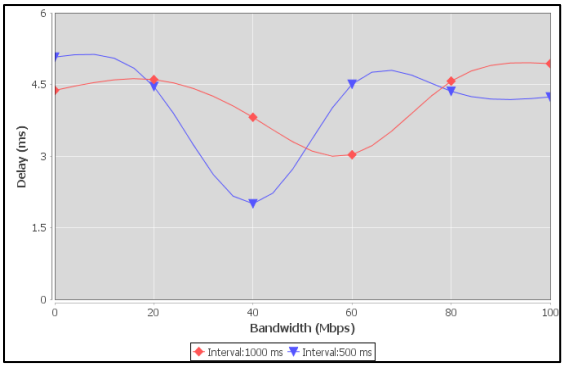

Figure 10. Average Delay for 8 Hop Count: 8

As shown in Figure 12, 13 and 14 for test case 2, we observe that maximum delays of probe interval 500ms are increasing with nearly two times than 1000ms probe interval after 40Mbps later. This is because of the fact that probe packet processing rate with interval 500ms is increased to compute delays in controller than interval 1000ms.

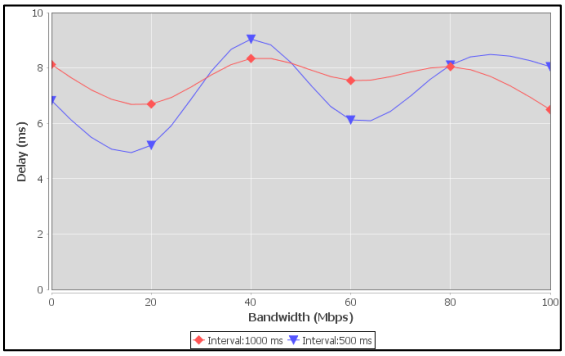

Figure 11. Average Delay for Hop Count:12

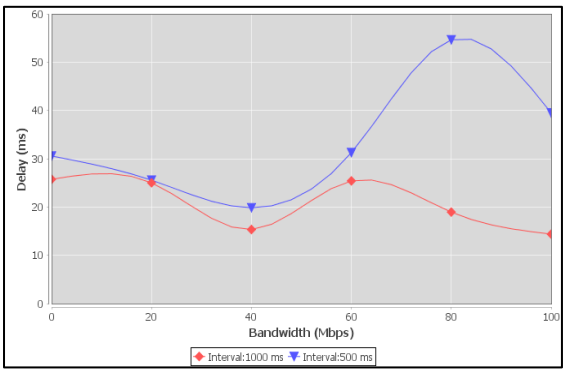

Figure 12. Maximum Delay for Hop Count:4

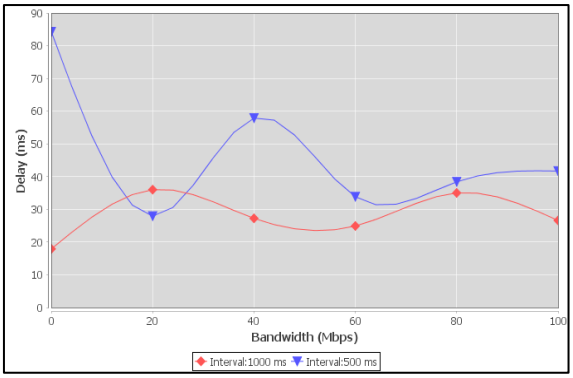

Figure 13. Maximum Delay for Hop Count:8

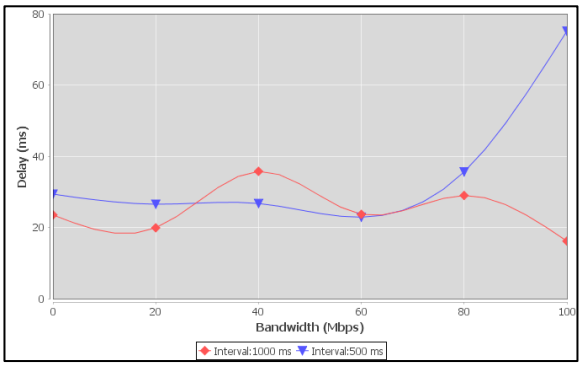

Figure 14. Maximum Delay for Hop Count:12

6. Conclusion

Delayed detection of large flow can degrade overall network performance, so the bandwidth utilization of network plays an important role of traffic engineering. In this paper, we presented large flow detection method which can detect and retrieve JSON about large flow in one second. So, this detection method can be applied quickly in order to address large flow problem. Moreover, we also described about how to measure current delays of paths by ONOS application and simulation results of delay measurement module are evaluated. According to results, delay measurement method is not efficient for less than probe interval 1000ms. These results are based on monitoring single datapath testbed. As future work, we will test this delay measurement module in monitoring two or more datapaths.

References

[1] O. M. E. Committee, “Software-defined networking: The new norm for networks,” *ONF White Paper,* Palo Alto, US, April 2012.

[2] Ramdhani, M.Fajar, S.Naning Hertiana, and B.Dirgantara. "Multipath routing with load balancing and admission control in Software-Defined Networking (SDN)." *Information and Communication Technology (ICoICT), 2016 4th International Conference on*. IEEE, May 2016, pp.1-6.

[3] Braun, Wolfgang, and M.Menth. "Load-dependent flow splitting for traffic engineering in resilient OpenFlow networks." *Networked Systems (NetSys), 2015 International Conference and Workshops* on. IEEE, March 2015, pp. 1-5.

[4] P.M. Thet, et al. "Design and functionality test of chunked video streaming over emulated multi-path OpenFlow network." Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2015 12th International Conference on. IEEE, June 2015, pp. 1-6.

[5] Peter Phaal, March 2013 [Online]. Available from: http://blog.sflow.com/2013/03/ecmp-load-balancing.html

[6] Mininet Community [Online]. Available from: http://mininet.org/

[7] Domżał, Jerzy, et al. "A survey on methods to provide multipath transmission in wired packet networks." *Computer Networks* 77 , Feb 2015, pp. 18-41.

[8] Azizi, Mounir, Redouane Benaini, and Mouad Ben Mamoun. "Delay measurement in OpenFlow-enabled MPLS-TP network." *Modern Applied Science* 9.3, Jan 2015, pp. 90-101.

[9] Akyildiz, Ian F., et al. "A roadmap for traffic engineering in SDN-OpenFlow networks." *Computer Networks* 71 , Oct 2014, pp.1-30.