ISBN 978-981-14-1455-8
Proceedings of 2019 the 9th International Workshop on Computer Science and Engineering
WCSE_2019_SPRING
Yangon, Myanmar, February 27-March 1, 2019, pp. 142-147
doi:10.18178/wcse.2019.03.024

Available Bandwidth Based Application-aware Engineering in SDN

May Thu Zar Win [1+], Yutaka Ishibashi [2] and Khin Than Mya [3]

[1,3] University of Computer Studies, Yangon, Myanmar
[2] Nagoya Institute of Technology, Japan

Abstract. Software-defined networking (SDN) has many advantages to support traffic engineering (TE) rather than the traditional network because of its decoupling of control and data plane, global centralized control, and enabling innovation through the network programmability. This paper focuses on the application-aware TE in SDN based on the estimation of end-to-end available bandwidth (ABW). First, we propose an application-aware TE framework for SDN which combines with sFlow-RT analyzer. The proposed framework performs three main tasks: traffic classification, traffic measurement, and traffic management. As the prerequisite of port number based traffic classification, we use sFlow-RT which provides traffic metrics via REST API. Collecting traffic statistics and calculation of ABW for each link based on the provided metrics are managed by traffic measurement module. Traffic management performs flow rerouting based on calculated ABW and the application's nature. We validate our proposed framework using the popular network emulation environment, Mininet and the widely used NOS called ONOS. Results of the experiment show increased throughput and decreased packet loss compared with other forwarding application.

Keywords: Software-defined networking, traffic engineering, application-aware engineering, available bandwidth.

1. Introduction

As the rapid growth of cloud computing and Internet of Things (IoT) technology, the traditional network architecture cannot handle the complexity of control protocols and internetworking of a large number of smart devices [1]. In cloud data-center networks, the equal treatment of all classes of applications is not the proper way to meet user's application-level requirements because the resource requirements of network applications are varied [2]. Application-aware engineering treats the application traffic depending on their own resource requirements such as bandwidth, network delay, and jitter [3]. Traditional networks also struggle to perform this complex application-aware engineering (AE) task. Therefore, an emerging architecture, software defined networking (SDN) has become a popular solution for companies to control their networks using a more cost-effective software solution, rather than with more expensive hardware methods.

Traffic engineering (TE) and security are still interesting research areas for SDN applications in the fields of an industrial environment. TE is an important network application which manages and measures the network traffics and determines the optimal routing solutions to optimize the network resources utilization and to meet requirements of the network QoS [4]. In this paper, we primarily focus on the application-aware traffic engineering in SDN by taking account of ABW as the QoS parameters. The main tasks of this paper are as follows: (i) the port-based application traffic classification is performed by using sFlow-RT analyzer, (ii) the proposed application-aware engineering dynamically collects ports statistics of source and destination switches according to the JSON information that is sent by sFlow-RT, and (iii) the application-aware also

+ Corresponding author. Tel.: + 95-425027402;
E-mail address: maythuzarwin@ucsy.edu.mm

calculates end-to-end ABW based on port statistics and reroutes best available path to meet the application's requirements.

The rest of this paper is organized as follows. The related works for calculating ABW and application-aware engineering describes in section 2. We propose the system design of ABW based application-aware engineering in section 3. Section 4 presents experimental methods of the system. We discuss experimental results in section 5. Finally, section 6 concludes the paper.

2. Related Works

Measurement techniques for QoS parameters (packet delay, packet loss ratio, and bandwidth utilization) are typically classified into active and passive techniques. Active techniques send probe packets into the network and evaluate how the network traversal affects the network status [5]. This technique can cause a temporary network congestion [6]. Passive techniques estimate QoS parameters by setting multiple measurement points in the network to monitor network statistics. These passive techniques are too complex to deploy in traditional networks. By taking the advantages of SDN's global centralized control, there are some papers to estimate ABW by using the passive techniques.

Megyesi et al. [7, 8] proposed the ABW estimation in SDN by using OpenFlow messages to track the bandwidth utilization of every link in the network and calculated the ABW on each path in the network based on the statistics information. In [7], they used the Floodlight controller and they also explained a proper trade-off is required between accuracy, polling rate, and network delay constraints. In [8], they focused on the source of errors for estimation of ABW measurement and highlighted that these errors are due to lack of a local timestamping mechanism in OpenFlow.

Singh et al. [9] estimated end-to-end ABW on any given path not only by composing link-wise ABW but also validating with a bandwidth measurement tool called Yaz. The ABW measurement is worked well by taking the traffic statistics from SDN controller [7-9], but the controller keeps querying statistics from all the switches in the network that may lead to overload network traffic. Therefore, this paper uses OpenFlow statistics to estimate the end-to-end ABW and reduces excessive network traffic by querying only the selected switches statistics that are provided by sFlow-RT analyzer.

There are some of the previous studies for application-aware traffic engineering in SDN. Cheng et al. [2] proposed application-aware routing scheme for SDN-based cloud data centers by taking into account of delay variation and link load but they only focused on LARAC-based routing scheme. Jarschel et al. [3] proposed DPI-based application-aware network for YouTube video streaming and rerouted the flow to least congested links based on monitoring of the YouTube's buffer level threshold and stalling. Jeong et al. [10] performed application-aware traffic engineering by using the port number and DPI-based traffics classifier to identify application or service flows and then they distributed the identified flows to different priorities queues in each switch port. Their evaluation results demonstrated that the initial flow treatment delay for DPI of the first flow rule increases but decreases the propagation delays in the congestion scenario. In this paper, we perform application-aware traffic engineering scheme in SDN combined with sFlow-RT analyzer.

3. Design and Implementation

In this section, we explained the design and implementation of our proposed system. The software defined network architecture mainly consists of three layers: application layer, control layer, and data plane layer. These three layers of SDN architecture and details components of the proposed system are described in Fig. 1.

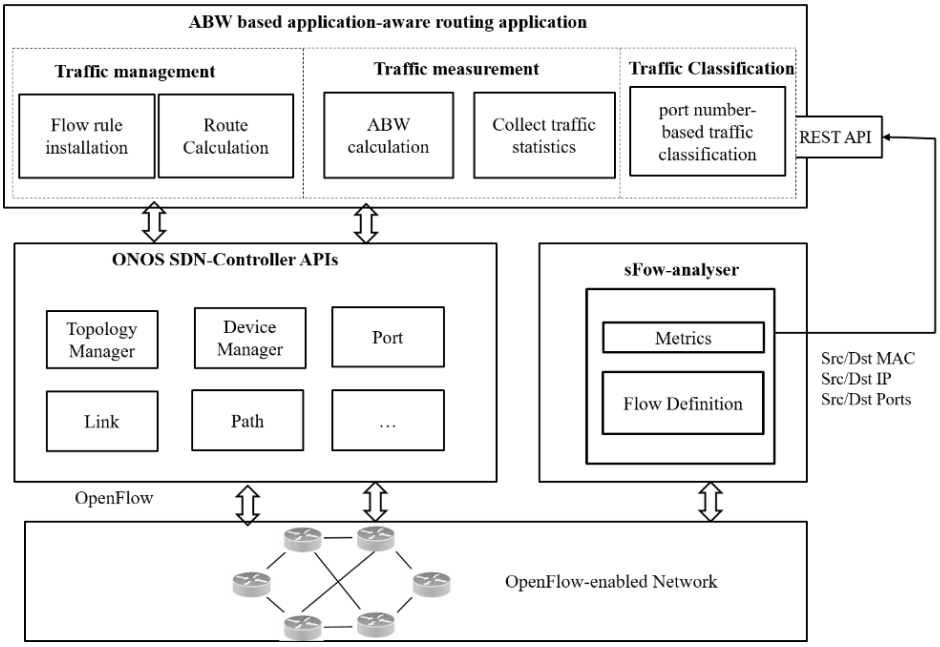

Fig. 1: System design.

3.1. Traffic Measurement

Classification of application traffics, collecting switches port statistics, and measuring ABW is the responsibility of the traffic measurement module. In this paper, we use sFlow-RT as a prerequisite for traffic monitoring and port-number based traffic classification modules. The sFlow analytics engine obtains a continuous telemetry stream from sFlow Agents embedded in network devices and converts them into actionable metrics, accessible through the REST flow API [11]. Figure 2 explains details of proposed system for the example of FTP traffic. The algorithm is written with Java in the application layer of ONOS controller.

```
Algorithm   : Application-aware routing
Procedure   ABW_Calculation
begin:
  Input      : srcIP,dstIP,srcPort,dstPort from sFlow-RT
  Output     : max_ABW_path
//classify application by validating its port with well-known port (e.g, FTP=20, RTP=5004)
   if ( srcPort ==20 OR dstPort==20) then
       print(That is FTP traffic and calculate ABW)
//get available paths of given source and destination IP by using ONOS API
       available_paths = getAllPaths (srcIp,dstIp)
//get available links along the path
       for path in available_paths
              available_links = getAllLinks (path)
//collect statistics for each link, calculate ABW by using statistics and select maximum ABW
              for link in available_links
                     collect statistics for each link
                     calculate ABW for each link
                     calculate max_ABW_path
              end for
        end for
   else
       print(perform default routing)
   end if
end
```

Fig. 2: Proposed ABW-based application-aware algorithm.

Taking the advantage of SDN's topology visibility, we can obtain many traffic statistics related with switch or router's port such as packets Received (pktRx), packet Transmitted (pktTx), bytes Received (bytesRx), bytes Transmitted (bytesTx), packet Received Drop Rate (pktRxDrp) and packet Transmitted Drop Rate (pktTxDrp). The received bytes of one switch was more than the bytes transmitted by other switch at the ports through which they are connected [9]. Therefore, we calculate the link load $L_i(t)$ for the source and destination port of the i^{th} link for a given path at time t through the following equation:

$$L_i(t) = srcPort\_bytesTx\ (t) + dstPort\_bytesTx\ (t) \quad (1)$$

$srcPort\_bytesTx\ (t)$ is the source port statistics of transmitted bytes count at time t and $dstPort\_bytesTx\ (t)$ is the destination port statistics of transmitted bytes count at time *t*. After calculating all the link load along a given path, the available bandwidth of i^{th} link at time t can be derived as equation (2).

$$ABW_{link_i}(t) = C_i(t) - L_i(t) \quad (2)$$

where $ABW_{link_i}(t)$ is the available bandwidth for the i^{th} link at time t, $C_i(t)$ is the capacity of the i^{th} link and $L_i(t)$ is the link load for i^{th} link at time t. Then, ABW of a given path is obtained through the following equation:

$$ABW_{path}(t) = \min_{link_i \epsilon Path} ABW_{link_i}(t) \quad (3)$$

The ABW path is the minimum ABW of links along a given path, and $link_i \epsilon Path$.

3.2. Traffic Management

Although the default forwarding application uses the shortest path for all the flows, we choose the path between the source and destination hosts in the network by taking account of the maximum ABW value. To route the best optimal path, the maximum ABW of the path between the source and destination host is calculated by equation (4).

$$ABW_{src \to dst}(t) = \max_{path \in path_{src \to dst}} \min_{link_i \in Path} ABW_{link_i}\ (t) \quad (4)$$

After calculating the best path, the new flow entries are added to respective devices along with the path by using FlowRuleService which is supported from the ONOS controller. The traffic selection fields of each flow entry are source and destination MAC address, TCP/UDP ports, and protocol type. When the bandwidth-aware application traffic (e.g, FTP traffic) enters to the network, sFlow-RT analyzer detects and sends the application traffic's metrics (i.e, srcIP, dstIP, srcMAC, dstMAC, srcPort, dstPort) to the proposed application-aware application and then steer the route to the maximum ABW path.

4. Experimental Method

In this section, evaluation environment measurements are presented. Tests were conducted by Mininet, network emulator on Ubuntu host with the ONOS SDN controller and sFlow-RT analyzer. In this environment, laptop PC (i.e., Core i5-4210U CPU @ 2.20GHZ with RAM 8GB, Ubuntu 16.04 OS) are used for evaluating the performance. Figure 3 sketches the classical Y topology which is frequently used as a test topology of application-aware engineering.

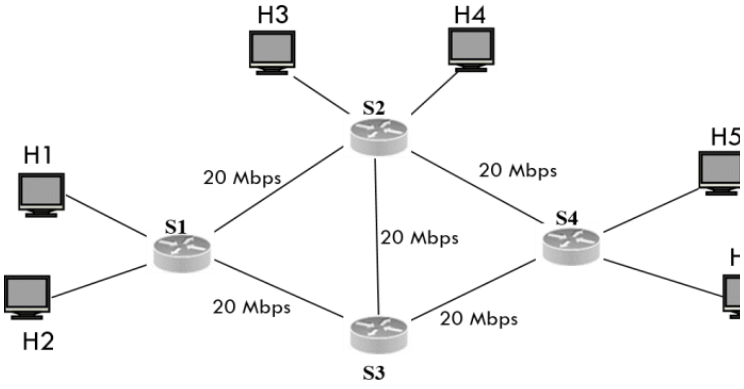

Fig. 3: Mininet test topology.

5. Experimental Results

In this experiment, we generated different bit rates of UDP traffics between switches s1 and s4 and carried out the following three test called *test 1, 2, and 3* here. In test 1, H1 sends 5 Mbps of UDP traffic to H5 for 20 seconds. In test 2, H1 sends 15 Mbps of UDP traffic to H5 and H2 parallel sending 5 Mbps to H6. In test 3, H1 sends 15 Mbps to H5 and H2 parallel sending 10 Mbps to H6.

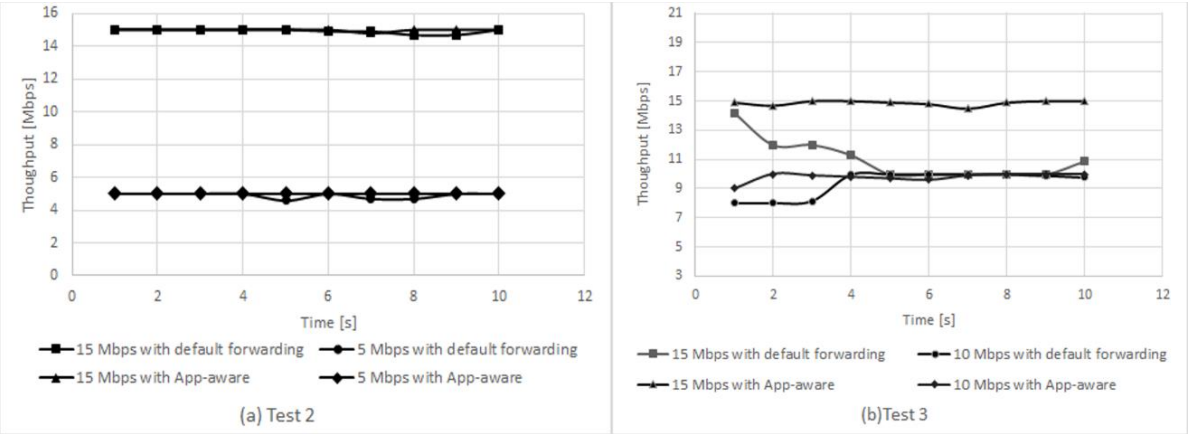

Fig. 4: Throughput results for test 2 and test 3.

In Fig. 3, we see that there are two equal cost paths between switches s1 and s4 (p1= {s1, s2, s4}, p2= {s1, s3, s4}). The idea of the proposed method is to select the maximum ABW path, however, the default forwarding only selects a single shortest path. In test 1, the throughput results of default forwarding and app-aware are almost the same because the amount of traffic (5 Mbps) is smaller than the current link capacity (20 Mbps). Therefore, we omit the throughput results of test 1 due to page limitation. In Fig. 4, we only present the throughput results for test 2 and test 3. If the traffic volume is equal to the current link capacity, there is a little difference between default forwarding and app-aware. In Fig. 4 (a), we see that the throughputs result of these two methods are almost the same.

In Fig. 4 (b), we find that the throughput results of 15 Mbps and 10 Mbps for the default forwarding are significantly lower than those for the proposed app-aware because when H1 sends 15 Mbps to H5, the default forwarding selects path p1. Thus, the path p1 already consumed 15 Mbps and left 5 Mbps. Parallel to this, when H2 sends 10 Mbps to H6, the default forwarding selects again the path p1 even the alternative path p2 provides a better ABW (20 Mbps). Therefore, nearly 23% of packet loss occurred in default forwarding. When the route decision is made by the proposed method, it selects a maximum ABW path for each traffic rather than the default path, for example, after H1 sends 15 Mbps to H5 via the path p1, the paths p1 left 5 Mbps. Therefore, when H2 sends 10 Mbps to H6, the app-aware selects the maximum ABW path p2 (20 Mbps). However, the proposed method needs the significant time to calculate ABW and install flow rules, the proposed method guaranteed fast packet transmission and packet loss. When the traffic volume is larger than the link capacity, the proposed method significantly outperforms than the default forwarding.

6. Conclusion

In this paper, we proposed an application-aware engineering scheme based on the ABW measurement. Port-based application traffics classification is performed with the help of sFlow. The proposed app-aware routing reduced the network traffic excessive by querying the selected switches instead of querying the whole network topology. The experimental results demonstrated that the total throughput of the proposed method outperforms the default forwarding when the traffic volume is larger than the link capacity, the method also reduces packet loss than the default forwarding.

For the future work, we will study the other QoS parameters like delay variation and packet loss to improve the application-level requirements. Moreover, we will compare our method with other path selection methods and we will also study our method with the complex network topologies and other NOS.

7. References

[1] D. Sinh, L.V. Le, B.S.P. Lin, L.P. Tung. SDN/NFV-A new approach of deploying network infrastructure for IoT. *Wireless and Optical Communication Conference (WOCC), IEEE*, June 2018, pp. 1-5.

[2] L.C. Cheng, K. Wang, Y.H. Hsu. Application-aware routing scheme for sdn-based cloud datacenters. *Ubiquitous and Future Networks (ICUFN), IEEE*, July 2015, pp. 820-825.

[3] M. Jarschel, F. Wamser, T. Hohn, T. Zinner, P. Tran-Gia. SDN-based application-aware networking on the example of youtube video streaming. *Second European Workshop in Software Defined Networks (EWSDN), IEEE*, October 2013, pp. 87-92.

[4] Z. Shu, J. Wan, J. Lin, S. Wang, D. Li, S. Rho, C. Yang. Traffic engineering in software-defined networking: Measurement and management. *IEEE Access*, Vol 4, June 2016, pp. 3246-3256.

[5] M. Jain, C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with tcp throughput. *Trans. IEEE/ACM*, Vol. 32, 2003, pp. 537–549.

[6] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, L. Cottrell. Pathchirp: Efficient available bandwidth estimation for network paths. *Proc. of Passive and active measurements workshop*, April 2003.

[7] P. Megyesi, A. Botta, G. Aceto, A. Pescapé, S. Molnár. Available bandwidth measurement in software defined networks," *Pro. of the 31st Annual ACM Symposium on Applied Computing*, April 2016, pp.651-657.

[8] P. Megyesi, A. Botta, G. Aceto, A. Pescapé, S. Molnár. Challenges and solution for measuring available bandwidth in software defined networks. *Computer Communications*, February 2017, pp.48-61.

[9] M. Singh, N. Varyani, J. Singh, K. Haribabu. Estimation of end-to-end available bandwidth and link capacity in SDN. *International Conference on Ubiquitous Communications and Network Computing*, *Springer*, August 2017, pp. 130-141.

[10] S. Jeong, D. Lee, J. Hyun, J. Li, J.W.K. Hong. Application-aware traffic engineering in software-defined network. *Network Operations and Management Symposium (APNOMS), IEEE*, 2007, pp. 315-318.

[11] https://www.sflow-rt.com.