Bandwidth Allocation Scheme using Segment Routing on Software-Defined Network

Ohmmar Min Mon, Myat Thida Mon
University of Information Technology, Yangon, Myanmar
ommm@uit.edu.mm,myattmon@uit.edu.mm

Abstract

Nowadays, Internet is the main communication medium but traditional network approach has been difficult to adapt to the fast growing Internet. The emergence of Software Defined Network (SDN) has brought to predict for the solution of this problem. Methods for measuring Quality of Service (QoS) parameters such as bandwidth utilization, bandwidth allocation and delay have been introduced not only for traditional network but also for SDN-based scenarios. Bandwidth allocation is a great significance in various areas of networking. We propose a low-complexity bandwidth allocation algorithm based segment routing for real-time and non-real-time service requests to allocate network resource to ensure the request of services. In this paper, we present available bandwidth allocation scheme based on SDN to provide Quality of Service (QoS) support for various services via experiments under our SDN testbed.

Key Words-Quality of Service, Bandwidth Allocation, Software Defined Networking, Segment Routing

1. Introduction

With the development of the Internet, there are more and more types of services carried by the Internet competing for network resource, most of which require performance guarantees. Nowadays IP networks are very complex to build and manage. Software Defined Networking (SDN) offers a solution for this problem mainly through the following features:

(1) data and control planes are decoupled;
(2) control logic is moved out of SDN switches to an external the SDN controller and
(3) external applications can program the network using the abstraction mechanisms provided by the SDN controller. Controller communicates networking devices to collect information from them and also to push configuration information to them.

Segment Routing (SR) is a new emerging routing technique and can be used in MPLS network that enables source routing. In the segment routing domain, nodes and links are assigned Segment Identifiers (SIDs). SR uses a sequence of segments to compose the desired end-to-end connection path. The segment labels are carried in the packet header and so per-flow state is maintained only at the ingress node. SR allows better control of the routing paths and can be used to distribute traffic for better network utilization. A central controller can take advantage of the possible segment routing by choosing segments based on the traffic. The major advantage of SR is to eliminate the per-flow states from the service provider's core routers. In fact, a path is directly usable by any router; no prior setup/signalization is required, unlike MPLS-TE where a tunnel has to be signaled and maintained using protocols such as the Resource Reservation Protocol Traffic Engineering (RSVP-TE) or Label Distribution Protocol (LDP). It allows the network operator to be forwarded a path from ingress node to egress node. When MPLS is used to instantiate SR tunnels, the MPLS forwarding plane does not change. There are many advantages when we integrate SR in the data plane and in SDN-based control layer technologies. As for scalability and agility, SR avoids the requirement for many labels to be stored in each network device along the path.

In the most recent years, there have been several proposals for monitoring Quality of Service (QoS) parameters in SDN networks. On the other hand, the paper in literature that deals with the problem of bandwidth allocation using SR in SDN. Service providers use this parameter for network management and traffic engineering purposes. In this case, users can send bandwidth allocation requests to the controller if necessary.

The controller receives the requests using the algorithm to calculate routes that can satisfy the demands of those users and allocate the bandwidth for them. This paper presents a solution for bandwidth allocation by monitoring traffic statistics from network devices in an SDN environment, we want to allocate bandwidth on each link in the network and find the highest available bandwidth between two points in the network. We also validate our method on a test environment using the Mininet network emulation tool and the ONOS SDN controller.

Internet Service Providers demand for enhanced network performance, increase the need for network resources. The main challenge is how to achieve optimal path for real time traffic. The second challenge is to

allocate bandwidth on each link for large networks. The major contribution of this system is to present Bandwidth Allocation Algorithm for the traffic that requests for the path with available bandwidth.

The rest of the paper is structured as follows:

Section 2 briefly reviews the related work. Section 3 explains segment routing operations. Bandwidth Allocation Algorithm is demonstrated in Section 4. Section 5 conducts the performance of our mechanism with experimental results on SDN testbed. Finally, section 6 concludes the paper and points out the future research directions.

2. Related Work

The SDN controller can communicate with the switch via the southbound API, where the most used standard is OpenFlow. For NOS platform there is much available open software such as NOX, POX, Floodlight or Ryu. Moreover, there are ongoing industrial projects for controller platforms specialized for data centers, for e.g. OpenDayLight or ONOS [1].

Slavica Tomovic, et.al[2] analyzed performance of bandwidth constrained and bandwidth-delay constrained centralized routing algorithms in large-scale SDN backbone networks. Bandwidth rejection ratio (BRR) and the average route length are used as the two main performance metrics to evaluate the algorithms. The best results of BRR performance were obtained when BWP (Bandwidth Propotion) parameter was set to 0.9.

Hyunhun Cho, et.al[3] constructed an application that calculates optimal paths for data transmission and real-time audio and video transmission. This paper showed that the average of the optimal path is about 100Mbps faster than that of other path.

In [4], the authors discussed the performance of the network by separating application into bandwidth-oriented and latency-oriented application. This system discussed the use case for the bandwidth and latency aware network.

Trong-Tien Nguyen and Dong-Seong Kim [5] present a novel routing scheme called Accumulative-Load Aware Routing for Software-Defined Networks. ALAR algorithm is implemented routing decisions based on Dijkstra algorithm to calculate the link cost values.

In [6], P. Megyesi, A. Botta, G. Aceto, A. Pescapè, S. Molnár presented an approach to measure end-to-end available bandwidth (ABW) in Software Defined Networks (SDN). This system reported the results obtained in different network configurations.

Péter Megyesi, et.al[7] proposed the use of a passive technique for the Available Bandwidth estimation, taking advantage of the NOS in the architecture of SDN. This system analyzed the source of errors introduced by SDN and OpenFlow and considered the possible implementations of ABW estimation based on meter statistics.

Cihat Cetinkaya, Erdem Karayer, MugeSayit, Cornelius Hellge [8] proposed a model to find the most suitable path for DASH services over SDN. It discussed how to add the different types of clients such as standard, HDTV and mobile users and improved the optimization model by considering the client properties.

In [9] Diyar Jamal Hamad, et.al proposed how to get traffic measurement statistics from network devices in an SDN environment. This system shows that the rate received from ports is a little bit higher than the generated traffic because there is background traffic in the network.

In [10], Rakesh Kumar, et.al proposed mechanisms that provide end-to-end delays for critical traffic in real-time systems using COTS SDN switches. This system shows that increasing the number of flows slightly decreases end-to-end delays.

This paper presents to allocate bandwidth on each link in SDN. The results show that bandwidth allocation algorithm in the SDN can optimize the routing paths.

3. Segment Routing Operation

In order to make the current IP and MPLS network more service-oriented and efficient, in 2013, IETF proposed Segmentation Routing (SR) technology[11]. It enables to use non-shortest paths by specifying alternative routes. It is typically associated with a centralized control plane implementation.

SR controller can support the global information of network resources and the global deployment and optimization of resources according to the service requirements of the source nodes, such as traffic engineering, load balancing, etc. The controller is in charge of setting up the edge-to-edge services, by configuring the ingress and egress PE nodes for a given flow.

In segment routing, nodes and links are assigned Segment Identifiers called segments. In the case of a link (i.e. adjacency) segment, the shortest path to the upstream node is taken and then that link is crossed.

There are three actions that are performed on segments by SR-capable nodes [1]. They are associated with operations performed on MPLS labels in MPLS networks. Segment Routing operations are: (a). PUSH (MPLS PUSH) – a segment is pushed on the top of segment stack (b) NEXT (MPLS POP) – an active segment is completed and it is removed from the stack (c). CONTINUE (MPLS SWAP) – active segment is not completed yet and it remains active.

Table 1 shows mapping of each SR control plane operation with a corresponding MPLS operation. The

basic functionality of each corresponding operation in both MPLS and SR are the same.

Table 1. SR operations mapping to MPLS operations

Segment Routing	MPLS
SR Header	Label Stack
Active Segment	Topmost Label
PUSH Operation	Label Push
NEXT Operation	Label POP
CONTINUE	Label Swap

In this paper, the switches update their forwarding tables according to the commands taken from the controller. In the forwarding tables of each switch, the output ports are kept according to the source address, destination address and input port. The switch informs the controller about the requested flow. The controller selects a path considering the requested bandwidth. After selecting an optimal path for the requested flow, the controller sends flow information to the corresponding
switches along the selected path via FlowMod message which is defined in OpenFlow protocol. In order to determine the path, the controller needs to calculate the available bandwidth of the paths. For this purpose, the controller queries the switches periodically via sending OFPC_PORT_STATS messages which is defined in OpenFlow protocol to obtain information about available bandwidth on the links.

Figure 1 illustrates a simple network topology to describe for segment routing operation. When a traffic flow has to be routed along the shortest path to its destination, a segment list including only one label can be used (i.e., the SID of the destination node). In Figure 1, if the target path for an incoming traffic flow is p = {R1, R2, R3, R4} the segment list is SL= {R4}. In this section, this system presents an example of how segment routing works.

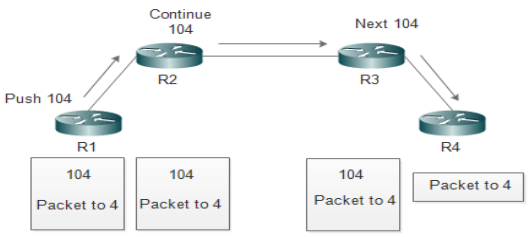

Figure 1. Segment Routing Operations

4. Bandwidth Allocation in SDN

Software Defined Networking offers the chance to speed-up the adoption of QoS control in the Internet. When a user needs certain bandwidth, it sends a bandwidth allocation request packet to the controller. Request packet contains information such as the identification of the user, the destination, how much bandwidth it needs, when it needs the service. When switch received a packet that has no matched flow-entry, it encapsulates the packet and sends the request packet to the controller as a Packet-in message. When the controller received a packet-in message, it checks whether it is a bandwidth allocation request or not. If it is, the controller uses the topology and bandwidth information to calculate the route for bandwidth allocation. Finally the controller responds the bandwidth allocation result to the switch.

4.1. Bandwidth Allocation in Controller and switch

End-to-end bandwidth allocation can be achieved by setting the flow to transmit packets from a host to another with QoS. Firstly the controller tries to route the higher bandwidth flow via a path that can accommodate the required rate. If such a path is not presented, the flow will be dropped. Best-effort flows skip the admission procedure and will always use the shortest path between the end hosts. The controller keeps track of how much bandwidth is allocated in the network. Using this information, the controller decides where to route the path.

Firstly, our application uses this information tobuild up the network topology graph $G(V, E)$ as shown in Figure 2, where the node set V corresponds to the switches and the edge set E corresponds to the links (for further notations see Table 2). Link states of edge E are indicated by vector: (C, u) where 'C' is the link capacity and 'u' is the link bandwidth utilization. The utilization of a link can be calculated by Equation. (1). In this case, bw_u is the bandwidth usage which can be obtained by monitoring the traffic flow of the link.

$$u_E = \frac{bw_u}{C_E} \qquad (1)$$

The available bandwidth of each path can be computed by Equation. (2):

$$bw_P = \min_{1 \le i \le n} (C_E - bw_u) \qquad (2)$$

Our mechanism attempts to avoid traffic congestion when new flow is added to the link. Here, we have to get the path P between source and destination in the network where the available bandwidth is the largest. This can be calculated through the following Equation. (3):

$$abw_{S->D} = \max_{P \in P_{S->D}} \min_{1 \le i \le n} (C_E - bw_u) \quad (3)$$

To get the path P with largest available bandwidth between source and destination, we use Modified Dijkstra algorithm. In this algorithm, the cost of the path, denoted C_P , is measured by the minimum bandwidth instead of the sum of edge of capacities used in traditional Dijkstra algorithm according to Equation (4).

$$C_P = \sum_{i=1}^{n-1} C(bw_u, C_E) \quad (4)$$

In this algorithm, a path from source to destination in G with the maximum bandwidth constructed. Here the array element P records the father of the node v in the maximum bandwidth tree and the array element P records the bandwidth of the path from source node to another node in the maximum bandwidth tree. This algorithm takes O(m+n long n) times to run where m and n are links and nodes respectively in the network G.

Table 2. Notation list

Notation	Description
G(V, E)	The directed graph representation of the topology with node V and edge E
u_E	Utilization of link in the topology graph
C_E	The capacity of link
bw_u	The link usage bandwidth
bw_P	The available bandwidth of each path
$P_{S->D}$	The set of all available paths from S to D

4.2. Bandwidth Allocation Algorithm

The general idea to measure the bandwidth of the path in the network is to provide end to end communication between switches and hosts. This section mentions Bandwidth Allocation Algorithm for the traffic that request for the path with available bandwidth. To explain Bandwidth Allocation Algorithm, consider a network with n nodes. To find the available bandwidth, the controller needs to know the current network topology and links reserved bandwidth.

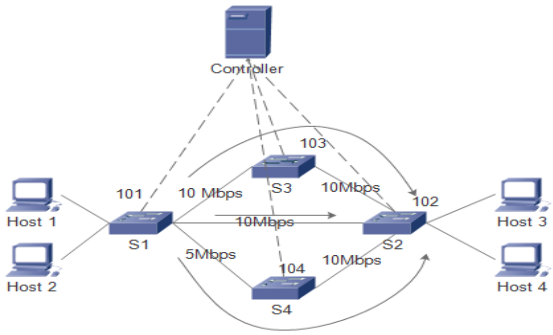

Figure 2. Network topology used in the experiment

This system uses the network topology that describes how bandwidth allocation can be implemented with SR. In the topology, source node is S1 and destination node is S2. When the flows entered the network, for the case of higher bandwidth application traffic S1-S3-S2, S1 would push a SR header with segment list *{*101,103,102*}*, and forward it to S2 as shown in Figure 2.

Best-effort traffic should be steered over a shortest path, which is S1 to S2. S1 would place the segment list *{*101,102*}* in the SR header, and forward to S2.

The algorithm in Figure 3 shows the Bandwidth Allocation Algorithm employed to compute all-pair maximum bandwidth paths. We propose this Algorithm, using the following notation:

- fbw= Feasible bandwidth
- abw= available bandwidth
- Input: Feasible bandwidth
- Output: best available bandwidth

```
Initialize# Find and Use Available Bandwidth
Input: fbw;
Initialize: abw=0;
If  fbw ≥ widest bw
        then abw = fbw
                Else abw = widest bw
Endif
For all n ∈ N do
        If bwn< abw  then bwn =0
Endfor
```

Figure 3. Pseudo code for available bandwidth

4.3. Segment Routing Emulation

We emulated the topology of the network using Mininet, a popular network emulator among Software Defined Network researchers. In Figure 4, we used the pseudo code to emulate the topology. Our implementations of the controller correctly assigned paths for each application. This section includes the setup and verification of a custom SR test application, utilizing Mininet and the ONOS controller. The SR tests contain a SR path selection process according to specific simulated application demands.

```
//Create LeafSpine Topology//
Input: Switch, Host;
Initialize: S=0, H=0, N=4;
For all i=1 to N do
        Si =addSwitch (Si);
        Hi =addSwitch (Hi , IPi);
        Li = createLink (Si);
Topo T = S,H,L;
Return T;
```

Figure 4. Pseudo Code for Creating SR topology

To create the traffic flows, ICMP requests and replies will be used. The system creates a network topology for segment routing tests. And the topologies are created in the ONOS virtual environment. The experiment is employed with the topology to evaluate bandwidth aware routing.

5. Experiment Environment and Results

This section describes the environment under which all the experiments are done. All the experiments are done on a virtual machine (VM) with the configuration using Leaf-Spine architecture. This Leaf-Spine architecture is designed to provide very scalable throughput in a uniform and predictable manner across thousands to hundreds of thousands of ports. Every leaf switch connects to every spine switch in the fabric.

Our evaluation was performed on the topology depicted in Figure 2 that we created within the Mininet framework used for emulating virtual SDN networks. Mininet is a network simulator for simulating SDN scenario in the Linux environment. The network topology is comprised of 4 virtual switches interconnected with an SDN controller, 4 virtual hosts and virtual Ethernet links interconnecting the switches. In our simulation, we write a script based on Python to generate flows that we need. The SDN controller is an Open Network Operating System (ONOS) controller configured to compute the available bandwidth.

The ONOS SPRING-OPEN project VM image has a 64-bit Ubuntu 16.04 installed as the guest OS and the VM is preconfigured to run with two virtual CPUs and two GB of RAM. These are the minimum requirements to run the environment. The PC has Microsoft Windows 8.1 OS and Oracle's VirtualBox hypervisor installed. The system was run with Core(TM)1.6 GHz CPU and 4 GB of RAM.

The ping command can not only provide connectivity results in the network but can also be used for latency measures. Ping provides the capability to determine the size of the packet that you are sending on the network.

The ONOS controller at the network topology is used to load a configuration for the real time traffic in order to emulate routing capabilities on them. Each switch is assigned with a loopback IP address and Segment Routing nodes are identified with Segment Routing ID (SID).

5.1. Evaluation Results

This section shows results of the tests that we run to evaluate Segment Routing network performance. In this system, the controller includes a routing policy based on a maximum bandwidth threshold 10Mbps between the switches. If the used bandwidth is below the acceptable threshold, the controller provides a routing preference for best-effort flow, otherwise for higher bandwidth flow.

On the ONOS controller, it is time to setup the virtualized network in Mininet. The resulting output after command execution is shown in Figure 5.

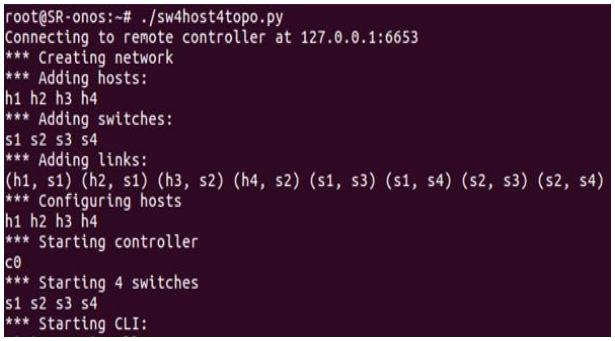

Figure 5. The virtualized network with SR

For traffic engineering with SR, the standard best path selection behavior can be seen by conducting series of ping requests from across the network. In Figure 6, executing ping between hosts in Mininet is done with the following command:

mininet>pingall

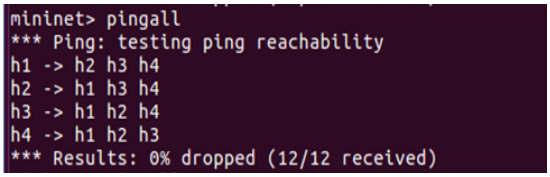

Figure 6. Ping between hosts-Screenshot

The available bandwidth performance is increased in real time traffic than best-effort traffic as shown in Figure 7. It represents utilized bandwidth for each port between switches and hosts with 5Mbps traffic and 10 sec. The results show that real time traffic is more than the best-effort traffic.

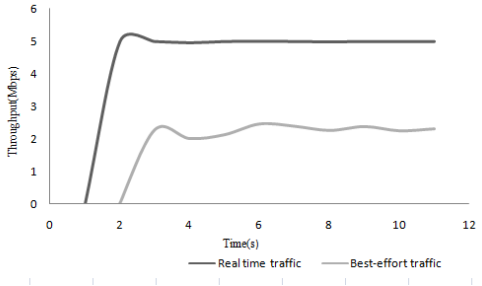

Figure 7. Available bandwidth-test with SR(5Mbps traffic)

Figure 8 also represents how the available bandwidth performance is increased in real time traffic than best-effort traffic. It represents utilized bandwidth for each port between switches and hosts with 10Mbps traffic and 10 sec. Results obtained show that real time traffic is more than the best-effort traffic.

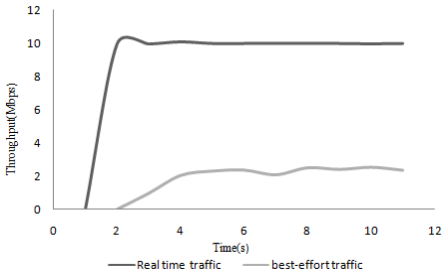

Figure 8. Available bandwidth-test with SR(10Mbps traffic)

The end-to-end delay for both switches is presented in Figure 9. The end-to-end delay is the sum of the delays
experienced at each hop on the way to the destination. Test results in Figure 9 indicate that the average delay is 3.5ms for best-effort traffic and 0.94ms for real time traffic respectively.

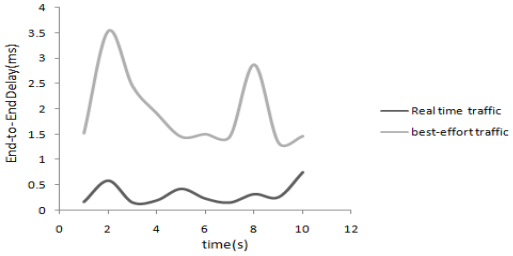

Figure 9. End to end delay-test with SR

The result of the bandwidth allocation algorithm is shown in Figure 10, which provides the distribution of SR path lengths. It is interesting to note that the mean number of hops in a SR path is 2 and the maximum number is 4.

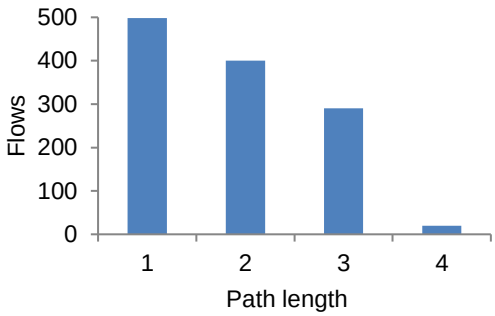

Figure 10. Distribution of SR path length (number of SIDs in the path)

6. Conclusion

In this paper we tackled the problem of Available Bandwidth calculation and monitoring in Software Defined Networks. We proposed a bandwidth allocation algorithm based segment routing for real-time and best-effort traffic. In this paper, we presented bandwidth allocation scheme based on SDN to provide Quality of Service (QoS) support for various services via experiments under our SDN testbed using Mininet for network emulation and ONOS for SDN controller with different kinds of traffic.

In the future work, we will further investigate to introduce other QoS parameters into QoS metrics, such as packet loss and reliability. We also used the Dijkstra algorithm to implement the bandwidth allocation. Then we will plan to extend the investigation of the local fast recovery with a large number of flows.

7. References

[1] B.Pankaj. “ONOS Open Network Operating System An Open-Source Distributed SDN OS”. 2013 Dec, 19, p.34.

[2] S.Tomovic, I.Radusinovic, and N.Prasad, “Performance comparison of QoS routing algorithms applicable to large-scale SDN networks”. In EUROCON 2015-International Conference on Computer as a Tool (EUROCON), IEEE 2015 Sep 8, (pp. 1-6).

[3] H.Cho, J.Park, J.M.Gil, Y.S.Jeong, and J.H.Park, “An Optimal Path Computation Architecture for the Cloud-Network on Software-Defined Networking”. Sustainability, 7(5), 2015 May 5 , pp.5413-5430.

[4] U.Pongsakorn, I.Kohei, U.Putchong, D.Susumu, and A.Hirotake, “ Designing of SDN-Assisted Bandwidth and Latency Aware Route Allocation”. (HPC), 2014 Jul 21, pp.1-7.

[5] T.T.Nguyen, and D.S.Kim, "Accumulative-load aware routing in software-defined networks". In Industrial Informatics (INDIN), IEEE 13th International Conference on 2015 Jul 22 (pp. 516-520).

[6] P.Megyesi, A.Botta, G.Aceto, A.Pescapè, and S.Molnár, "Available bandwidth measurement in software defined networks". In Proceedings of the 31st Annual ACM Symposium on Applied Computing 2016 Apr 4 , (pp. 651-657).

[7] P.Megyesi, A.Botta, G.Aceto, A.Pescapé, and S.Molnár, "Challenges and solution for measuring available bandwidth in software defined networks". Computer Communications, 99, 2017 Feb 1, pp.48-61.

[8] C.Cetinkaya, E,Karayer, M.Sayit, and C.Hellge, "SDN for segment based flow routing of DASH". In Consumer Electronics–Berlin (ICCE-Berlin), 2014 IEEE Fourth International Conference on 2014 Sep 7 (pp. 74-77).

[9] D.J.Hamad, K.G.Yalda, and I.T.Okumus, "Getting traffic statistics from network devices in an SDN environment using OpenFlow". ITaS, 2015 Sep, pp.951-956.

[10] R.Kumar, M.Hasan, S.Padhy, S., K.Evchenko, L.Piramanayagam, L., S.Mohan, and R.B,Bobba, "End-to-End Network Delay Guarantees for Real-Time Systems using SDN", 2017.