Flow Collision Avoiding in Software Defined Networking

Myat Thida Mon
Faculty of Computer Systems and Technologies
University of Information Technology
Yangon, Myanmar
myattmon@uit.edu.mm

Abstract

With the high volume of traffic in recent network, traffic between the switches has exchanged rapidly among the servers that may conduct the link congestion. In a traditional network, routing protocols are used the shortest path algorithm and congestion will occur in the network links. Equal-Cost Multi-Path (ECMP) algorithm cannot fairly distribute the bandwidth of the links between the flows. These flows suffer long queuing delays and degrades throughput. Flow Collision Avoidance algorithm (FCAA) is an efficient method to alleviate the network congestion by rerouting the flows to guarantee the performance of the network. Congestion occurs when flow demand exceeds link capacity and it leads to the degradation of QoS. The algorithm evaluates the existing flow's demand based on port statistics to handle the collision of the large flow in the network avoiding the congestion. The evaluations indicate that SDN-based FCAA algorithm enforces the required end-to-end QoS for each traffic flow over ECMP.

Keywords: *ECMP; FCAA; SDN; QoS*

I. INTRODUCTION

Software Defined Networking can enable an effective programmable solution to allow flexible network for the current enterprise network. SDN is an architecture that reduces the network operation costs and optimizes the network resource usage. In SDN, network monitoring can be accomplished using OpenFlow statistics and the controller can keep track of available bandwidth on each link. SDN increases higher rates of innovation and it decreases the obstruction for a new technology in large scale network. The SDN represents a network framework with programmatic ways to control forwarding function.

ECMP is a networking strategy for load balancing to allocate flow to get an available path. ECMP is per-flow, hash based traffic distribution. It routes the flows on to the paths over static hashing. It is effective for small flows but it is not operative for large flows due to static hash collision. ECMP is a forwarding mechanism to forward packets along the multiple paths of equal cost with the goal to get distributed link load sharing. These static flows do not consider for resulting collisions degrading overall utilization for existing network. To guarantee the desired overall performance a service, network operators need to offer accommodations for QoS applications.

The SDN Controller gets the commands using OpenFlow statistics and interconnects with the applications including port statistics from the network devices. OpenFlow allows network traffic control from the controller and it also offers many concepts like traffic engineering. OpenFlow is a protocol and OpenFlow switches connect with other switches with OpenFlow ports. OpenFlow messages forward the traffic between OpenFlow enabled switches through the network. Each switch maintains a flow table that contains forwarding information. The controller and switches communicate via OpenFlow messages. OpenFlow controller can arrange data flows by replying the OpenFlow messages from switches. By using the OpenFlow protocol, an OpenFlow controller can insert flow entries into OpenFlow allows network traffic control from the controller and it also offers many concepts like traffic engineering. Based on the traffic statistics along with the network bandwidth information, a controller can construct the network topology. The controller provides network control decision and network devices to forward packets through the traffic statistics.

To get the alternate path for the absence of failures, the existing forwarding protocols are improved. Congestion control is a key factor in ensuring network stability and robustness. In this paper, this paper compares FCAA against ECMP to handle the collision of the large flow in the network avoiding congestion. FCAA can also minimize the network congestion by rerouting the flows over the alternative paths. FCAA algorithm compute alternate light loaded path by forwarding the flow to avoid collision for the large flows.

The rest of the paper is designed as follows:

Related work is explained in Section II. Then this paper discusses flow congestion avoidance scheme in Section III. The evaluation results are discussed in Section IV. In Section V, this paper concludes and future work is provided.

II. RELATED WORK

The proposed method is based on Traffic Engineering. In this section, this paper will discuss the literature works.

In [1], Ian F. Akyildiz, et al. provides an overview of traffic engineering mechanisms to manage data flow efficiently at both the control plane and the data plane in SDN architectures. They also discuss classical TE mechanisms developed for ATM, IP and MPLS networks, and then survey in detail the state-of-the-art in TE for SDN from both academia and industry perspectives.

An efficient method based on SDN discussed for reduction of congestion in data-center networks for rerouting of selected network flows in the switches with the congested links [2].

The authors proposed a method to reduce congestion in SDN data center networks. The controller finds the costs of each link and rerouted the flows through alternate paths with minimum load. The authors in [3] do not consider the overhead for the controller when the controller computes the link utilization and link load of each path.

The authors presented that link utilization is calculated in the SDN controller and recalculated rerouting algorithm is applied to switches which would be configured by using OpenFlow configuration protocol [4].

In [5], the authors proposed an efficient reallocating method for SDN based on genetic algorithm. This paper is compared with ECMP in terms of throughput, packet loss and data transfer to minimize network congestion.

The authors in [6], the problem proposed to obtain improved allocation of resources using the Genetic Algorithm but it is not able to realize the parallel optimization.

The paper in [7] utilizes the hierarchical fat-tree network to efficiently schedule flows based on traffic statistics that deliver multiple alternative paths among hosts and to handle the network congestion via OpenFlow protocol.

The authors in Mahout [8] proposed to monitor and detect the large flow on the terminal host through a shim layer in the operating system rather than directly detecting the switches in the network.

III. Flow Congestion Avoidance in SDN

The concept of SDN enables an efficient network configuration by separating the control logic from the infrastructure network supporting centralization. The control plane controls the behavior of the network by installing forwarding rules to the infrastructure layer. Finding the optimal paths for traffic demands is a problem to define routes dynamically in traffic engineering (TE). Traffic engineering optimizes the performance of the network and avoids the congestion on any one path. It guarantees bandwidth guarantee in the network links. It has the capability to distribute QoS issue traffic when the network congestion occurs.

SDN enables the network control to convert directly programmable and centralized management and the infrastructure layer abstracts the network applications and network services. The SDN makes simpler the network management to reduce operating costs and support innovation. The SDN controller monitors traffic statistics to install forwarding rules into the switches as shown in Figure 1. Traffic measurement in SDN is a vital task to gather statistical data about network flows from the switches in real-time.

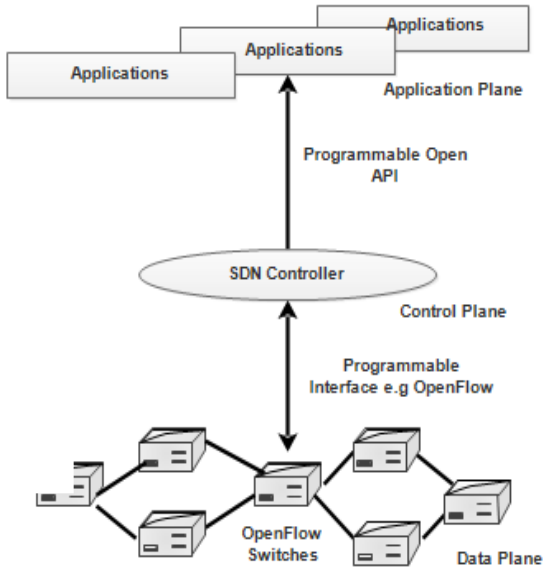

Figure 1. SDN Architecture

The SDN Controller gets the information using OpenFlow statistics. OpenFlow provides access to the infrastructure layer over the network and it forwards the packets between the switches. The controller propagates the forwarding rules to all switches in the infrastructure layer.

Traditional enterprise networks use Equal Cost Multipath (ECMP). It is used to allocate a path for each new flow that depends on hashing. It also performs the static traffic splitting and it can collide on the switches between the large flows. So it can degrade the efficient resources utilization. ECMP cannot handle to select the path dynamically. This paper compares FCAA against ECMP to handle the collision of the large flow in the network avoiding congestion. FCAA considers rerouting flows when the network congestion occurs over the alternative links. It provides the efficient routing scheme to select a new path for the large flow with the minimum congestion when link congestion occurs.

The main contribution of this paper is to compute alternate light loaded path by forwarding the flow to avoid collision for the large flows and the FCAA algorithm that is outperformed the conventional ECMP in fat-tree network.

The controller sends Statistics Request message to the switches to achieve the required port statistics. The switches reply to the controller with Reply message. Then the

controller provides the decision for the new flow to a virtual network composed by queues of appropriate links which meets its performance requirements. The controller computes the light loaded path for the flow and assigns this flow into the switch. When the controller occurs flow entries from switches on the heavy-loaded path, it will shift the flow entries into switches over the light-loaded path.

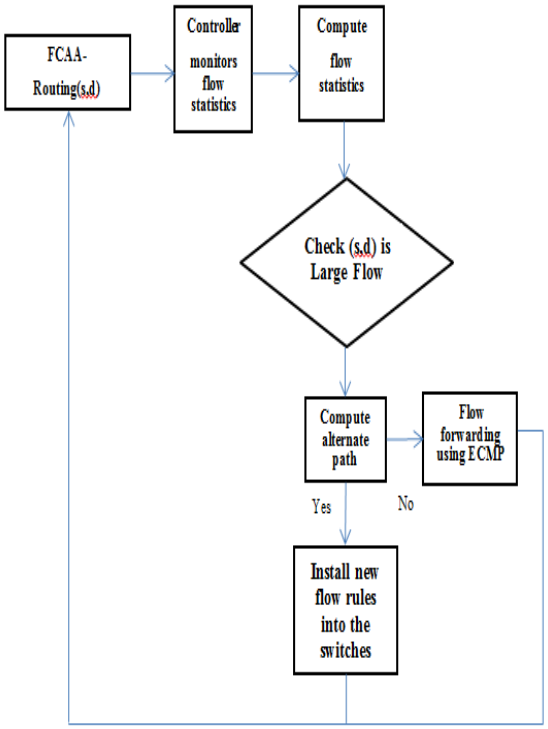

Figure 2. Algorithm Flowchart of FCAA

In the FCAA algorithm, the controller configures the estimated threshold value (10% of link capacity) to switches for classifying the large flow. When the large flow is detected, the proposed system will use the flow collision avoidance algorithm FCAA to deal with the corresponding large flow. The algorithm routes the flows along the lightly loaded path.

The method is to find the light-loaded path for a flow described in the Algorithm as shown in Figure 2. This system uses this information to build up the network G(V, E) where the vertex V corresponds to the switches and the edge E corresponds to the links.

IV. EXPERIMENTAL SETUP

The evaluation was presented to demonstrate the algorithm based on the fat-tree network topology as shown in Figure 3. The VM image has a 64-bits Ubuntu 16.04 installed as the guest OS. The system consider fat-tree of k=4. A fat-tree network has three layers: core, aggregation and edge. Nowadays, the fat-tree network is one of the most widely used topologies. It also consists of two types of elements: core and pods. The Fat-Tree topology consists of 20 switches and 16 hosts. Each pod has k/2 aggregation switches and number of k-port switches. In Ports Statistics Monitoring, the system sets the monitoring period as 5 seconds. And it sets the maximum capacity as 100 Mbps. The metric used to compare the ECMP is the throughput and flow completion time. The Mininet emulator is used to create the virtual networks to model the fat-tree network topology.

The SDN controller in the system is a free OpenFlow controller that runs with the Mininet in the Ubuntu. The algorithm can be applied to the fat-tree network topology as an example. The system uses iperf tool to test the performance. The system runs iperf servers on hosts H1, H2, while two iperf clients are started on each of H9, H10. In the FCAA algorithm, the controller configures the estimated threshold value (10% of link capacity) to switches for classifying the large flow. Lastly, when the measurements of a flow exceed threshold value, the controller determines that the flow is a large flow and it will forward some flows to the light loaded path based on port statistics. In the measurement, this paper has been tested with default window size is 85.3 kB for both tests.

The performance of FCAA is evaluated in terms of throughput. Throughput is calculated using the formula.

$$Throughput = \frac{Amount\ of\ data\ transferred}{Time\ taken\ to\ transfer\ data} \qquad (1)$$

```
********************Estimated Demands********************

10.1.0.1 10.1.0.2 10.2.0.1 10.2.0.2 10.5.0.1 10.5.0.2 10.6.0.1 10.6.0.2

10.1.0.1 | 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
10.1.0.2 | 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
10.2.0.1 | 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
10.2.0.2 | 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
10.5.0.1 | 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
10.5.0.2 | 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00
10.6.0.1 | 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00
10.6.0.2 | 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00
```

Figure 3. Demand estimation between flows

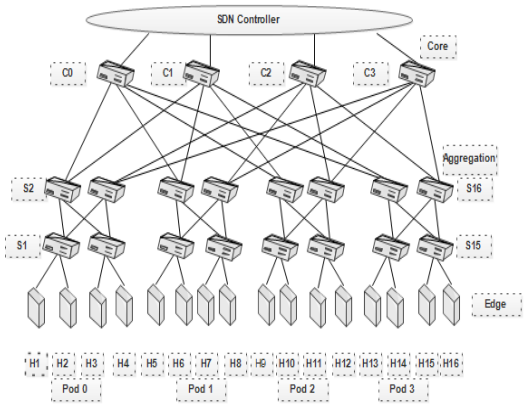

Figure 4. The topology of Fat-Tree network in SDN with k=4

TABLE 1. SIMULATION PARAMETER SETTING

Parameters	Values
SDN Protocol	OpenFlow 1.3
Controller	OpenFlow Controller
Software	Mininet 2.3
Link bandwidth	100 Mbps
Threshold	10 Mbps
Window Size	1 MB

In the testing, flow 1, 2 and 3 from source host H1 and H2 will take different routes to their destination hosts. A host sends packets to another host in the network. The FCAA algorithm is performed by generating the different numbers of flows as shown in Figure. 3. The parameter settings used in the testbed is shown in Table I. FCAA achieves better performance than ECMP. ECMP increases significantly due to congestion. FCAA outperforms ECMP. The system uses Iperf to generate traffic flows among switches in the different pods. Iperf is command line-only tool for the active measurement of the maximum network throughput for IP networks. The system tests the fat-tree network topology using Mininet network emulator as shown in Figure 4.

For example, assuming that flow f from h1 to h9 uses the primary path (S9 → S10 → C1 → S2 → S3), the alternate paths used to keep the link on the main path of flow obtained using FCAA. The algorithm calculates the light loaded path for each congested large flow. Therefore, the congested flow will be rerouted through (S9 → S12 → C2 → S4 → S3).

In this testbed, the throughput between flows from H1 to H15 was measured. In the depicted figure 5, 6, 7, the flow between H1 and H9 are steering the flows in the Flow-1. Then, the two hosts, H2 and H10 are forwarding the flows is shown in Flow-2. The final Flow-3 represents the flow between H3 and H11. The comparison of the experiment results was drawn by using FCAA and ECMP for the throughput of flow1 in Figure 5. According to the testing, the performance of FCAA flows increase quickly from 57.7 to 94.8. ECMP flows increase slightly to 87.4. The results produced by ECMP are receiving poorer due to congestion.

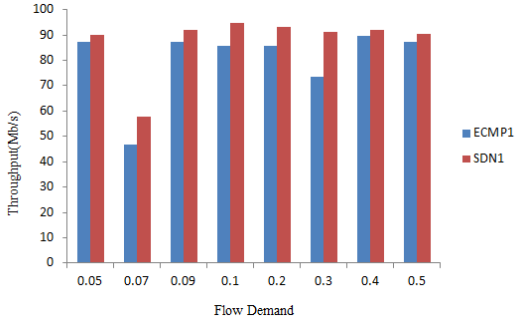

Figure 5. Throught result for Flow1 between Demand estimation

The comparison of the experiment results was depicted by using FCAA and ECMP for the throughput of flow2 in Figure 6. According to the testing, the performance of FCAA flows increase quickly from 82.2 to 95.6. ECMP flows increase slightly to 89.6. FCAA flows have the potential to improve long flow throughput at flow demand 0.1.

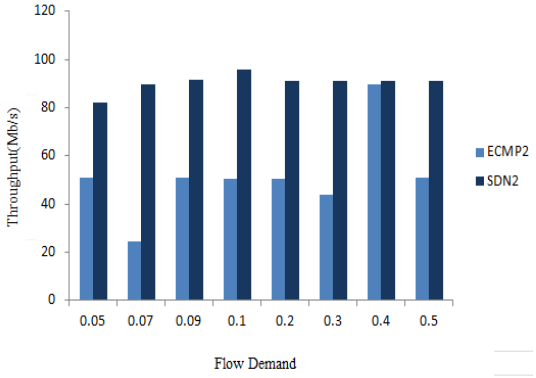

Figure 6. Throught result for Flow2 between Demand estimation

In figure 7, the comparison of the experiment results was presented by using FCAA and ECMP for the throughput of flow3. According to the testing, the performance of FCAA flows increase quickly from 32.7 to 67.7. The results produced by ECMP are receiving lesser due to collision. FCAA flows also can still balance the loads of flows more efficiently and have higher throughput at flow demand 0.1.

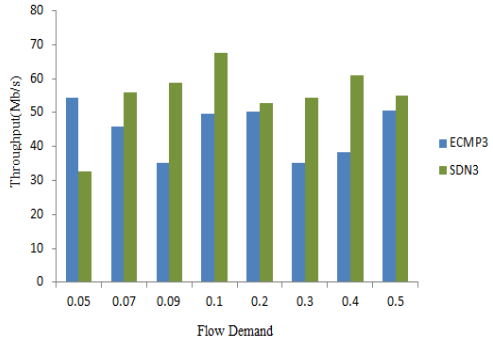

Figure 7. Throught result for Flow3 between Demand estimation

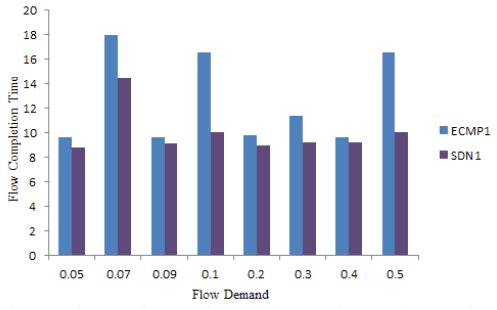

Figure 8. Flow Completion Time between Flows

The test result of the flow completion time FCT for FCAA is shown in Figure 8. The FCT for ECMP increases

from 9.6 to 16.4. In FCAA, FCT increases slightly from 8.8 to 14.5. For flow completion time, FCAA achieves better performance than ECMP.

V. CONCLUSION

In this paper, the traffic-based collision avoidance method was implemented that reduces network congestion according to their flow demands and flow statistics based on SDN. It can deal with the problem of large flow collision and the traditional ECMP. For the collision of large flows, the effect of FCAA collision avoidance was also presented to avoid congestion and focused on the problem of ECMP through fat-tree network topology. For the experiment, the Mininet emulator is used to evaluate the algorithm by comparing ECMP. The performance of the FCAA algorithm has the potential to improve higher throughput than the conventional flow hashing-based ECMP. In conclusion, more research is necessary to calculate optimized paths and to develop collision avoidance approaches to find collision free traffic. As for the future, the system will be planned to improve the congestion avoidance of fat-tree network management and to improve the effectiveness of network resource utilization.

REFERENCES

[1] I.F Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks", 2014, Computer Networks, pp.1-30.

[2] Y. HanS.S. Seo, J. Li, "Software Defined Networking-based Traffic Engineering for Data Center Networks", The 16th Asia-Pacific Network Operations and Management Symposium pp. 1-6. IEEE.

[3] M.Gholami, B. Akbari. "Congestion Control in Software Defined Data Center Networks Through Flow Rerouting", 2015 IEEE, pp . 654-657.

[4] S. Seungbeom, L. Jaiyong, S. Kyuho, J. Hangyong and L. Jihoon, "A Congestion Avoidance Algorithm in SDN Environment", 2016 IEEE.

[5] M.M Tajiki, B. Akbari, M. Shojafar, M., S.H Ghasemi, M.L Barazandeh, N. Mokari, L. Chiaraviglio, M. Zink, "CECT: computationally efficient congestion-avoidance and traffic engineering in software-defined cloud data centers". Cluster Computing, 2018, pp.1881-1897.

[6] L. Yilan, P. Yun, Y. Muxi, W. Wenqing, F. Chi, J. Ruijuan , "The Multi-Path Routing Problem in the Software Defined Network" 2015, 11th International Conference on Natural Computation (ICNC).

[7] Y. Li, D. Pan, "Open Flow based load balancing for fat-tree networks with multipath support", Proc. 12th IEEE ICC13, 2013, pp. 1-5.

[8] A. R. Curtis, W. Kim, P. Yalagandula. "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection", INFOCOM, 2011 Proceedings IEEE, pp. 629-1637.

[8] A. R. Curtis, W. Kim, P. Yalagandula. "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection", INFOCOM, 2011 Proceedings IEEE, pp. 629-1637.