10th ICSE

2019

PROCEEDINGS

Volume 1

* Engineering Education
* Electrical and Electronics
* ICT
* Manufacturing and Automation

Organized by

Yangon Technological University

Sponsored by

JICA

December 7-8, 2019

ICSE2019-ICT-21

Statistics Measurement of Network Traffic in Software Defined Networking

Ohmmar Min Mon#1, Myat Thida Mon#2

Faculty of Computer Systems and Technologies, University of Information Technology,
Yangon, Myanmar

1 ommm@uit.edu.mm

2 myattmon@uit.edu.mm

Abstract - **The recent developments in existing network infrastructure need high bandwidth capacity and multiple paths between high-end switches. Software-Defined Networking (SDN) provides QoS for network flows. Statistics measurement is an essential task in traffic engineering. Traditional equal cost multi-path (ECMP) routing used to statically steer flows among multiple equal-cost paths but it cannot guarantee optimal resource utilization. This static hash collision can degrade throughput and high latency for the flows. Flow Path Computing algorithm (FPCA) can minimize the network congestion by rerouting the flows over the alternative paths in SDN. The FPCA algorithm evaluates the existing flow's demand based on port statistics and the existing flow is shifted from the congested path to the alternative light loaded path. In this paper, this paper also takes a look at statistics measurement in SDN to improve the performance of HTTP and FTP traffic over ECMP. The evaluations indicate that FPCA algorithm enforces the required end–to-end QoS for each traffic flow.**

Keywords - **Software Defined Network, ECMP, QoS, Port statistics, OpenFlow**

I. Introduction

Software Defined Networking is introduced to provide programmable solutions and to bring network flexibility by separating the control plane and data plane. The demand for multimedia applications is increasing on the enterprise network. Statistics measurement solutions in SDN provide reliable traffic measurement of traffic engineering parameters such as delay variation, bandwidth to detect to an enormous range of network applications. To guarantee the desired service performance with peak demands, network operators provide accommodations for high QoS applications. To efficiently report such a challenge, the increasing of new applications and services need severe performance demands.

The active research topic SDN has more advanced features while using traditional network function. The SDN architecture can support high QoS applications such as voice, video and real-time applications. The SDN Controller gets the commands from the application layer and interconnects back to the applications including port statistics and events from the network devices. 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 forward packets through a well-known protocol such as OpenFlow. OpenFlow allows network traffic control from the controller and it also offers many concepts like traffic engineering.

To select the optimal path for the absence of failures, the existing forwarding protocols are improved. Conventional enterprise networks use the Equal Cost Multi-path (ECMP). ECMP is one of the most general solutions for traffic engineering to increase the fast protection performance of interior gateway protocols. These static flows do not account for resulting collisions degrading overall switch utilization for existing network utilization. Congestion control is a key factor in ensuring network stability and robustness. In this paper, this paper compares FPCA against ECMP on network parameters such as throughput for HTTP and FTP traffic and the FPCA algorithm guarantees the demanded QoS stage.

The rest of the paper is designed as follows:

The related work for the statistics measurement is discussed in Section II. Then this paper discusses the traffic measurement scheme and the FPCA algorithm used to measure the statistics 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], an efficient algorithm exploits the SR assignment by finding the optimal solution with the shortest list of SIDs by using Shortest Path First algorithm to allocate SR path. It implements equal cost multi-path routing (ECMP). This paper simplifies the control plane operation by representing the available network segments for calculating the segment list to reduce the available payload area.

The authors proposed an application to allocate the optimal paths between the networks and to offer the end-to-end connection among the users at the end of the network. This paper is applied to the SDN routing computation SRC to select the optimal path for the multi-domain environment in [2]. This SRC application does not consider when there is no abrupt change in the network like traffic growth.

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 [3].

D.J.Hamad, et al in [4], implements fine-grained traffic engineering procedures using the statistics on the switches that calculate the link utilizations and link capacities on the links. This system collects link usage information in terms of the number of bytes and they do not consider the traffic congestion for each link when generated traffic is greater than the link bandwidth from the ports in the network.

This method controls the congestion in SDN based OpenFlow using the traffic statistics of the network devices and flows are rerouted through paths with more free resources. The authors in [5] consider two parameters such as throughput and average packet delay reduction.

The authors in [6] provide bandwidth guarantees for priority flows for efficient use of network resources. To be able to satisfy the bandwidth requirement, the Dijkstra algorithm is used to reduce traffic routing for providing QoS flows of the network.

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 [7].

In [8], the authors proposed an efficient congestion avoidance method based on the flow requirements. This paper is compared with ECMP to prevent congestion and the waste of resources by allocating resources.

The authors in [9], BCMPO problem proposed to obtain more flexible control and better allocation of resources using the Genetic Algorithm but it is not able to realize the parallel optimization.

Ian F. Akyildiz, et al. [10] 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.

III. Traffic Measurement scheme

Software-Defined Networking is a new management approach for planning networks that separates the control plane and forwarding plane of a network. For traffic engineering (TE), the problem of finding the optimal paths for traffic demands is to define routes dynamically. Traffic engineering is the avoidance of congestion on any one path and ensures bandwidth guarantee across the network. TE has the capability how to distribute QoS issue traffic when the network congestion occurs.

In SDN, the forwarding plane lies under the control plane and all the routing decisions are made by the control plane. The SDN controller monitors traffic metrics to install forwarding rules into the switches. The result is sent back in a form of a new entry to flow table in the switch for subsequent flows. The application layer lies above the control plane. The SDN controller in the control plane communicates with the application layer by North Bound API and to the data plane by South Bound API like OpenFlow as shown in Figure 1. Statistics measurement in SDN trusts on gathering statistical data about network flows from the switches in real-time. Statistics measurement is a crucial task in traffic engineering. It includes three main tasks: topology measurement, statistics measurement, and performance measurement. OpenFlow is a well-known open protocol that offers the control plane to data plane. It allows network administrators to modify the flow-table by permitting the network to be changed programmatically by the network applications and services. So, the network administrators can implement QoS by collecting network statistics. Real-time applications are provisioning to support network changes and to allocate network resources.

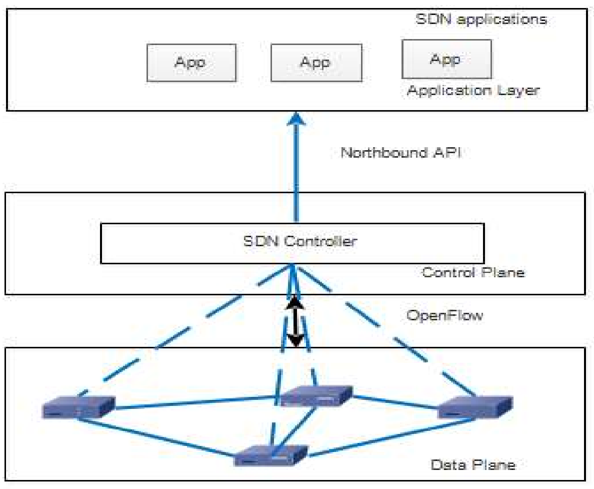


Fig. 1 Architecture of Software Defined Network

Traditional enterprise networks use Equal Cost Multipath (ECMP). A key limitation of ECMP is that two or more large, long-lived flows can collide on their core switches and the utilization of bandwidth in these links may exceed the network threshold. ECMP cannot handle to select the path dynamically. This paper describes the proposed SDN-based statistics measurement for rerouting between source and destination. In this system, the route selection of the flows is done by considering the paths from the controller to the switches. Congested links are identified as over-utilized links that cause network congestion and packet loss.

The main contribution of this paper is to implement the SDN-based statistics measurement with OpenFlow and the FPCA algorithm that is outperformed the conventional ECMP in fat-tree network. And then the traffic through these congested links is rerouted through the light-loaded alternate path and the total network throughput increases.

The performance of the system is measured by throughput for HTTP and FTP traffic. When the routing module receives a Packet-in request, the controller checks the paths that satisfy the flow. If the flow demand is exceeded 10% of link bandwidth, it computes the light loaded path based on port statistics and the existing flow is shifted from the congested path to light loaded path.

The method for finding the QoS path for a flow is described in the Algorithm. This system uses this information to build up the network G(V, E) as shown in Figure 3, where the vertex V corresponds to the switches and the edge E corresponds to the links. In this case, the free bandwidth is defined the free bandwidth in Equation. (1).

$$Free_{bw} = \max(LinkCapacity - PortSpeed * \frac{8}{1000}) \quad \text{-----(1)}$$

In Figure 2, the pseudo-code is presented for flow path computing algorithm (FPCA). Then f_D represents flow demand, bw_{max} defines the maximum bandwidth, bw_f represents feasible bandwidth and C_h defines the congested path.

```
Input:  link bandwidth, Topology: G(V,E)
Output: Number of selected path
p= selected path;
Initialize: L.capacity =$bw_{max}$;
Begin
Loop

For each $p_{src\rightarrow dst}$ in link do

        if $bw_f + f_D \leq$ L.capacity then

        $p \leftarrow bw_f + f_D$

        return p
        else
        p= $C_h$(hash),
        return p = $p_{src\rightarrow dst}(C_h)$
end Loop
end
```

Fig. 2   Flow Path computing algorithm

IV. Experimental Setup

The evaluation was performed on the topology depicted in Figure 3. The VM image has a 64-bits Ubuntu 16.04 installed as the guest OS. The system was run with Core(TM) (i5) 1.6 GHz CPU and 4 GB of RAM. These are the minimum requirements to run the environment. To emulate the algorithm, this paper use the topology of Fat-tree network with k=4, which consists of K-port switches with k pods. Each pod is composed of two layers of k/2 switches. In addition, each aggregation is composed of (k/2)2 core switches with one port attached to each k pod. Each edge switch is directly connected to (k/2) hosts and the remaining k/2 port of edge switches is connected to a (k/2) aggregation switch. The testbed consists of 16 hosts interconnected using a fat-tree of 20 (4-port) switches as shown in Figure 3.

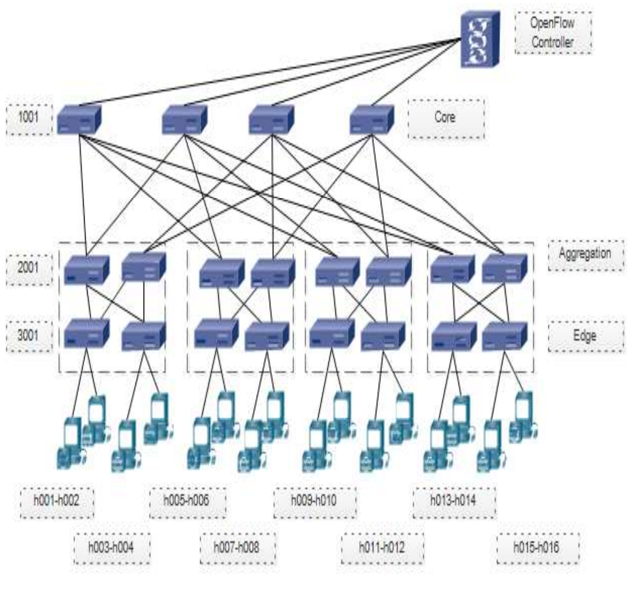


Fig. 3 Experiment with Fat-tree Network topology

The evaluation was performed on the Mininet testbed to create a network topology with OpenFlow virtual switches. The network topology is based on Fat-tree topology to provide better performance in throughput for each traffic flow. All flows are generated with iperf between iperf servers and clients. Throughput results from the flows are generated via iperf's statistics. The algorithm is implemented in Mininet using Python to create flows. In the FPCA algorithm, the threshold of link bandwidth utilization is 10% of link bandwidth. If the flow demand of a switch exceeds the specified threshold, it will forward some flows to the light loaded path based on port statistics and finally, the throughput of each traffic flow will be improved. In the measurement, this paper has been tested with flow size 23 MB and default window size is 85.3 kB for both tests.

Experimental results are based on the different parameters for the network topology. Two types of traffic are tested in the experiments: HTTP traffic and FTP traffic. The FPCA algorithm is performed by generating the different numbers of flows as shown in Figure. 4, 5 and Table I.

TABLE I
Number of Flows

No. Flows	**Source->Destination**
HTTP Traffic	H005->H001, H006->H002, H007->H003, H008->H004
FTP Traffic	H005->H001, H006->H001, H007->H003, H008->H003

Table II shows the list of the results of four HTTP flows (h001-h005, h002-h006, h003-h007, h004-h008) in this experiment. Table III shows the list of the results of FTP flows (h001-h005, h001-h006, h003-h007, h003-h008) in this experiment.

TABLE II
Throughput of Http Traffic

Data Flow	**100 MB**	**200 MB**	**300 MB**	**400 MB**	**500 MB**
FPCA	91.2	182.4	272.8	365.6	449.6
ECMP	57.6	163.2	232.8	359.2	434.4

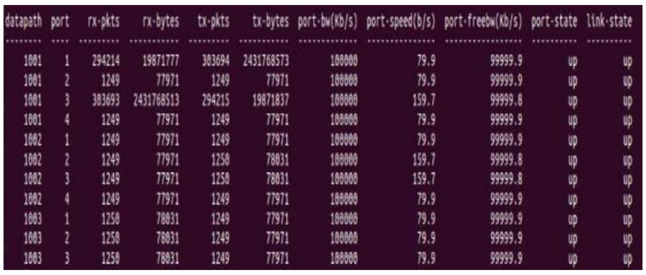
TABLE IIII
Throughput of FTP Traffic

Data Flow	**100 MB**	**200 MB**	**300 MB**	**400 MB**	**500 MB**
FPCA	91.3	183.76	273.19	361.44	449.36
ECMP	90.6	138.16	193.04	356.32	436.16

The list of the results of flow completion time for four HTTP flows (h001-h005, h002-h006, h003-h007, h004-h008) is shown in Table IV. FPCA achieves better performance than ECMP. ECMP increases significantly due to congestion. FPCA outperforms ECMP.

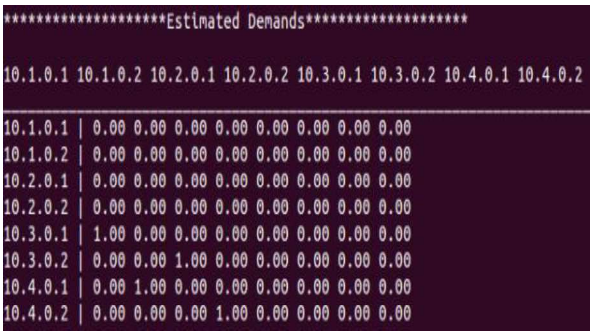
TABLE IV
FLOW COMPLETION TIMES BETWEEN FPCA AND ECMP

Flows	ECMP (ms)	FPCA (ms)
4 (HTTP)	3.71	2.0
4 (FTP)	3.75	1.99



```
datapath port  rx-pkts    rx-bytes  tx-pkts    tx-bytes port-bw(Kb/s) port-speed(b/s) port-freebw(Kb/s) port-state link-state
-------- ---- --------- ---------- --------- ---------- ------------ -------------- ---------------- ---------- ----------
    1001    1    294214   19871777    303694 2431768573       100000           79.9          99999.9         up         up
    1001    2      1249      77971      1249      77971       100000           79.9          99999.9         up         up
    1001    3    303693 2431768513    294215   19871837       100000          159.7          99999.8         up         up
    1001    4      1249      77971      1249      77971       100000           79.9          99999.9         up         up
    1002    1      1249      77971      1249      77971       100000           79.9          99999.9         up         up
    1002    2      1249      77971      1250      78031       100000          159.7          99999.8         up         up
    1002    3      1249      77971      1250      78031       100000          159.7          99999.8         up         up
    1002    4      1249      77971      1249      77971       100000           79.9          99999.9         up         up
    1003    1      1250      78031      1249      77971       100000           79.9          99999.9         up         up
    1003    2      1250      78031      1249      77971       100000           79.9          99999.9         up         up
    1003    3      1250      78031      1249      77971       100000           79.9          99999.9         up         up
```

Fig. 4 Port Statistics from the controller



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

10.1.0.1 10.1.0.2 10.2.0.1 10.2.0.2 10.3.0.1 10.3.0.2 10.4.0.1 10.4.0.2
------------------------------------------------------------------------
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
10.3.0.1 | 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
10.3.0.2 | 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00
10.4.0.1 | 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00
10.4.0.2 | 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00
```

Fig. 5 Demand estimation between flows

The fat-tree network topology as shown in Figure 3 is created by using Mininet emulator. For example, assuming that flow f from h005 to h001 uses the primary path (3005 → 2005 → 1001 → 2001 → 3002), the alternate paths used to protect every link on the primary path of flow obtained using Algorithm. Flows interfere locally at the Aggregation switch due to a hash collision. The algorithm tries to select the best rule that can protect the flow without interfere other backup paths. Therefore, the congested traffic will be rerouted through (3005 → 2006 → 1003 → 2002 → 3002).

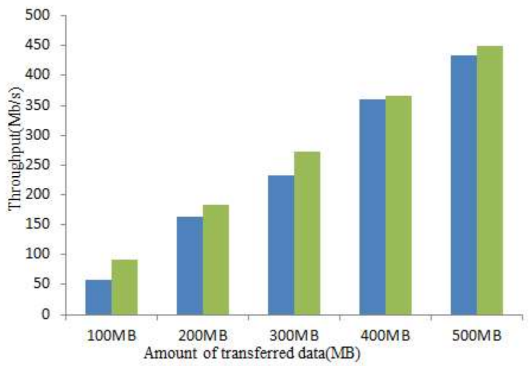


Fig. 6 Throughput between HTTP traffic

This paper compares the throughput per flow under a different number of OpenFlow switches deployed using the proposed FPCA and ECMP. The test result of throughput between HTTP traffic is shown in Figure 6. HTTP traffic in ECMP decreases 3.38% than FPCA. The FPCA receives packets at desired around 2 seconds. ECMP flows increase significantly 3.71s due to congestion. Throughput results of HTTP in FPCA in all traffic with different bandwidths also achieves better performance than in ECMP.

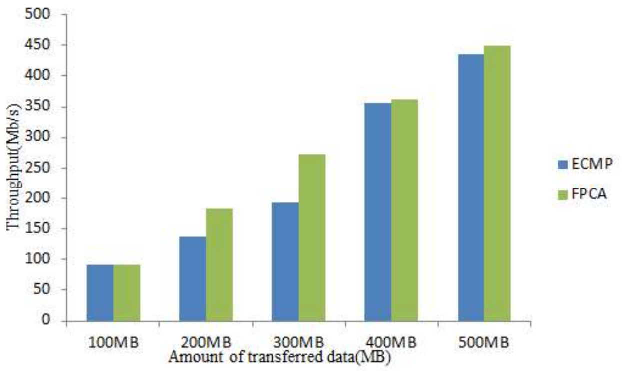


Fig. 7 Throughput between FTP traffic

The test result of throughput between FTP traffic is shown in Figure 7. FPCA also achieves better performance than in ECMP. ECMP decreases by 2.93% than in FPCA. The FPCA receives packets at desired around 2 seconds. ECMP flows increase significantly 3.75s due to congestion. Throughput results of FTP in FPCA in all traffic with different bandwidth also achieves better performance than in ECMP. So the topology of FPCA in SDN architecture is suitable for fat-tree networks than ECMP.

V. CONCLUSIONS

In this paper, this paper focused on the problem of ECMP and congestion control through fat-tree network architecture. If the existing flow's demand is exceeded 10% of link bandwidth, the flow is rerouted to the light loaded path using port statistics. This paper has implemented for statistics measurement in SDN to improve the performance of HTTP and FTP traffic over ECMP. This paper was considered with three metrics: (i) flow demand, (ii) port speed and (iii) free bandwidth. The experimental results proved that 3.38% and 2.93% throughput improvement for HTTP traffic and FTP traffic compared with ECMP. Based on the simulation results, the performance of the FPCA algorithm outperforms the conventional flow hashing-based ECMP. This paper would like to enrich the congestion control of fat-tree network management in the future.

REFERENCES

[1] S. Salsano, L. Veltri, L. Davoli, P.L. Ventre, and G. Siracusano, "PMSR—Poor Man's Segment Routing, a minimalistic approach to Segment Routing and a Traffic Engineering use case". NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium, 2016, pp. 598-604.
[2] 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, no. 5, May 2015, pp. 5413-5430.
[3] 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.
[4] D. J. Hamad, K. G. Yalda and I. T. Okumus, "Getting traffic statistics from network devices in an SDN environment using OpenFlow". ITaS, September 2015, pp. 951-956.
[5] M.Gholami, B. Akbari. "Congestion Control in Software Defined Data Center Networks Through Flow Rerouting", 2015 IEEE, pp . 654-657.
[6] S. Tomovic, I. Radusinovic and N. Prasad, "SDN control framework for QoS provisioning", 22nd Telecommunications forum TELFOR 2014, Serbia, Belgrade, November 2014.

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

[8] 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.

[9] 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).

[10] 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.

Contact
Conference Secretariat Office
Main Building(1/1-3), YTU, East Gyogone,
Insein Township,Yangon, Myanmar
Email : icse@ytu.edu.mm, icseoffice@gmail.com
Tel : +95-1-9663291

ISBN 978-99971-0-791-6
9 789997 107916 >

PROCEEDINGS of The Tenth International Conference on Science and Engineering, 2019 (Volume 1)

10th ICSE

2019

PROCEEDINGS

Volume 1

* Engineering Education
* Electrical and Electronics
* ICT
* Manufacturing and Automation

Organized by

Sponsored by

December 7-8, 2019