Congestion Control Using OpenFlow in Software Defined Data Center Networks

Masoumeh Gholami
The Faculty of Electrical and Computer Engineering
Tarbiat Modares University
Tehran, Iran
masoomeh.gholami@modares.ac.ir

Behzad Akbari
The Faculty of Electrical and Computer Engineering
Tarbiat Modares University
Tehran, Iran
b.akbari@modares.ac.ir

***Abstract*— this paper studies congestion control issue in data center networks and proposes a potential solution based on OpenFlow protocol. A main feature of the emerging data center networks is their performance in hosting different cloud applications and services. Since congestion management is necessary to effectively utilize numerous data center applications, in this paper we present an efficient method to control the congestion in the software defined data center networks based on OpenFlow protocol. In our proposed method, congestion in the network links is recognized by centrally checking the port statistics of the OpenFlow enabled switches and after that some of the flows in any congested link are rerouted through paths with more free resources by the OpenFlow controller. We executed our proposed method by Mininet, and the experimental results demonstrate the efficiency of the proposed method in decreasing the congestion and enhancing the performance of the network.**

***Keywords—* OpenFlow; data center; congestion control; fat-tree; software-defined network.**

I. Introduction

Nowadays, data centers services have been increasingly used in campus, enterprise, and companies for running a diverse range of programs and applications. Data centers have multiple networks such as storage area network (SAN), local area network (LAN) and clustering high performance computing (HPC) network. Each communication infrastructure struggles to meet the objective of their traffic. For example SAN and HPC need to achieve high speed lossless packet forwarding and LAN need to carry best effort traffic. The presence of different types of communication patterns in data centers leads to an increase in the cost of power consumption, system cooling, infrastructure maintenance, as well as administration management of data centers [1-3]. To solve this issue, IEEE 802.1 Data Center Bridging (DCB) work group [4] is enhancing Ethernet as a single unified network of data centers which can satisfy the performance requirements of single unified network such as losslessness and low transmission delay. However, the proposed scheme still lacks a set of quality service requirements where one of the indispensable requirements is Ethernet congestion management. Regarding the network traffic characteristics of data centers [5], Ethernet congestion management must greatly reduce transmission delay and bandwidth consumption, and significantly optimize application throughput and performance of the network. Therefore, providing high throughput at network is fundamental to the performance of data center for congestion management.

Hence, to eliminate the transient congestion, IEEE 802.1Qbb work group [6] provides a link surface current control mechanism known as PAUSE mechanism which can temporarily stop the link when the buffer is full. Even though, this mechanism can guarantee losslessness, it results in congestion spreading to other network nodes which will extremely degrade the serious performance of the network due to the longitudinal congestion. The IEEE 802.1Qau work group [7] has developed a new two-layer end-to-end congestion management known as Quantized Congestion Notification (QCN). In this way, whenever congestion occurs in a switch, a feedback message is transmitted to the source that causes congestion to reduce its data transmission rate. Upon receiving the feedback message, the related source decreases its transmission rate by activating the rate limiter. This approach leads to alleviating congestion without congestion spreading over the network. Nevertheless, when congestion occurs in the network owing to contention of end hosts or severely large bursty traffic, reducing source transmission rate is unavoidable. Furthermore, QCN scheme relies on parameters setting and network configuration failing to guarantee high throughput and low latency for all kinds of network.

Meanwhile, a typical data center topology called *fat-tree* often has the possibility to deliver huge bandwidth through providing multipathing between any pair of hosts [8]. However, the full utilization of bandwidth proves inefficient due to unbalanced traffic distribution which causes heavy congestion among core/ aggregation switches in the network. Besides, this issue leads to a long delay and high packet drop in all flows owing to congestion causing degraded network performance. Therefore, it is essential for congestion management in data centers to achieve high throughput and low latency for efficient communications. To date, a considerable amount of work has been dedicated to congestion management using sampling frame in order to detect congestion and feedback message to control it. However, these solutions should measure congestion according to buffer queue length and prevent it through the control messages so that they lead to extra network overhead and increase its complexity. To tackle this, to the best of our knowledge, there has not been any previous study on congestion control in data center using

Software Defined Networking (SDN) technology [9]. Since congestion management can tremendously benefit numerous data center applications, we study congestion control in fat-tree data center in this paper [10].

SDN is an innovative network technology that offers high interoperability and cost-efficient ways of user control and network programmability in data center networks. The main difference between a traditional networking and SDN based networking is separation of data plane and control plane in SDN based networks. In SDN technology, the whole network is centrally managed by a dedicated controller which interacts with network switches using the OpenFlow protocol. A common OpenFlow network is composed of three components: the OpenFlow controller, OpenFlow switches, and hosts. Each of the switches maintains a flow table that contains routing information. The controller and switches communicate via OpenFlow messages. There are a series of actions that the OpenFlow controller can perform by sending messages to the switches, such as updating flow tables or probing switch statistics. By analysis of the reply messages from the switches, the OpenFlow controller can detect network congestion and reroute the network paths, through OpenFlow protocol, in order to decrease the congestion [11].

In this paper, an efficient approach based on SDN is introduced for reduction of congestion in data center networks. The main characteristics of the proposed approach are 1) enhanced operational efficiency, 2) dynamic decision making about the congestion and 3) response efficiency. In this approach, the congestion detection and routing component are the main modules of OpenFlow controller. The main property of our proposed approach is rerouting some selected flows in the switches using the congested links. Due to practical constrains such as data center real platform and real traffic, we simulated a data center network with Fat-tree topology using Mininet [12] emulator and Floodlight controller [13] to evaluate our proposed congestion control approach. Our experimental results show that by deploying our proposed approach, performance metrics such as throughput and end-to-end delay are considerably improved.

The remaining part of the paper proceeds as follows: Section II presents our proposed approach for congestion control. In section III, experiment of the proposed approach is conducted. Subsequently, the performance of the proposed approach is presented and discussed. Finally, the conclusion and further works are presented in Section V.

II. METHOD

Traditionally, a variety of methods have been used to assess congestion control in data center such as BCN, QCN etc. Each has its advantages and drawbacks. A major problem with the classical method is that it is not flexible and dynamic relative to the network status. Hence, the SDN approach is one of the most flexible ways for congestion control and to evaluate the effectiveness of performance in the network. In this section, we present an efficient approach based on SDN that is able to control congestion as well as dynamic decision making under congestion conditions in a fat-tree data center. Our proposal is described with five components, including network topology creation, host management, congestion detection, congestion control, re-routing.

A. *Topology creation component*

This component aims to discover, storing link status to make network current topology. Using Link Layer Discovery Protocol (LLDP), this component sends LLDP packet on all ports for link detection. By the use of received information from switches, the network current topology is stored for the controller and the information is also available for re-rerouting component.

B. *Host management component*

Host management component maintains all identified hosts over the network. This component stores the requirement information including MAC address of source and destination, IP address of source and destination, OpenFlow switch ID connected hosts and the number of ports of OpenFlow switch connected hosts. Using this information, re-routing component computes the proper route for large flow identified, whenever congestion occurs in the network.

C. *Congestion detection component*

The aim of this component is to query and store the statistics from all OpenFlow switches. These statistics are then used by the control component to identify large flows and the re-route component to compute the load on various links. This component gathers statistics from each OpenFlow switch (per table, per flow and per port) by polling them at fixed intervals. For congestion detection, the controller sends STATS_REQUEST message assigned to the PORT, FLOW and TABLE in the network as the statistics of each switch. Then, the switch responds to the controller with STATS_REPLY message. The controller examines transmitted bytes on the port of each switch at fixed intervals. Once the transmitted bytes are 70% higher than the link capacity, the congestion conditions occur in the port. Then the controller using congestion control component identifies the large flow which causes congestion and this flow path is altered with the least loaded shortest path.

D. *Congestion control component*

This component aims to identify one or more large flows which cause congestion by the use of gathered statistics. In order to identify large flow size and also life time of flow considered. Namely, L is defined as flow size at time estimated and b_t is defined as byte totally of flow received by the use of the switch at time, I defined as time interval that it is considered 5ms:

$$L_t = (b_t - b_{t-i})/I \tag{1}$$

After identifying one or more large flows, the flows path should be changed with an alternative path. Through the controller through the OFP_FLOW_MOD message, the current table of the switches is updated.

E. *Re-routing component*

The control component is responsible for rerouting. It computes the minimum loaded shortest paths between a set of the shortest paths based on the statistics collected from the switches by the congestion detection component.

Once the large flow is examined in the network, the controller performs re-routing process based on the network topology. For further re-routing, first, it computes all the possible shortest paths with Dijkstra algorithm [14]. Moreover, to measure the least loaded path, the statistics from the congestion detection component will be used to compute it. The total cost of each path $p_i \in P$ is defined as $y_i = a_i + b_i$,

$$a_i = \sum_k^N w_k\,, \qquad b_i = \sum_k^N \frac{L_k}{C_k} \tag{2}$$

Where a_i is the total fixed costs of each link in the path $l_k \in p_i$ and b_i is the total variable costs of $l_k \in p_i$. Each l_k has a predetermined fixed weight w_k, a link load of L_k bps computed from the statistics, and link capacity C_k bps. Currently, all link weights are set to 1. The value of L_k is estimated from the change in byte count for all flow entries in a switch between the two most recent statistics from the congestion detection component. After computing paths y_i from all $p_i \in P$ the controller then picks the path that has the minimum y_i. If the amount of statistics collected is insufficient, a route is randomly picked from P. Through the OFP_FLOW_MOD message, the current table of the switches is updated.

III. Experiment Results

In this experiment, we employ emulator Mininet to assess the performance of our proposal method. One of the most significant features Mininet supports is SDN. Mininet creates virtual networks including core, switch, user codes etc just on a single machine. Because of the simple communing and customizing the most of components, this emulator can be applied in various development, training, and research projects. After creating the network from OpenFlow switches, we need to control these switches. Here, we evaluate Floodlight controller which is an open-source controller with modular and flexible controller platform at its core. This OpenFlow controller is implemented purely in software and is included in its own Java Virtual Machine (JVM). Floodlight is a community for the promotion and/or proposing standardization of SDN Northbound APIs, so that services that use an OpenFlow controller can be written quickly and efficiently.

In order to assess performance, we build up a k=4 fat-tree network which includes 20 OpenFlow switches, 16 hosts as can be seen from Fig. 1.

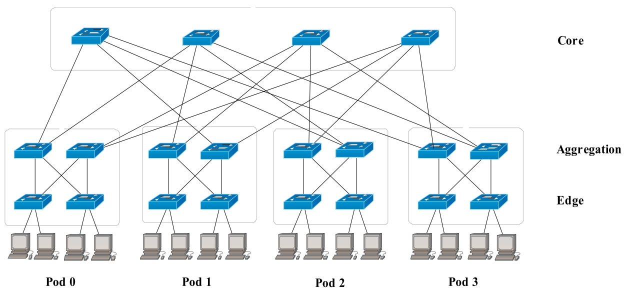

Fig. 1. The topology of Fat-tree with k=4.

To generate traffic on the network which is the main challenge of the network, we used the highly precise Ipref instrument [15], which suits analytical and industrial purposes. Ipref is an open source code, which can be used for the evaluation of the traffics generated through TCP and UDP. This instrument reports different types of statistical scales such as throughput delay dispersion, and datagram removal. Using the Ipref instrument, the packets with determined size are transmitted in a given time and a specific share of load proportional to the linkage rate. We carry out two experiments to evaluate the effect of important network parameters, such as maximizing throughput and minimizing latency on the performance of the network. We compare performance of our proposed method with a scheme without congestion detection and with congestion detection. In the following section, each of the experiments will be described.

A. Experiment 1:k=4 fat-tree network

We tested the impact of parameters on the network with k=4 fat-tree network. In the test, we ran scenario with packet size 64 byte, load amount 10% to 99%, duration 1 minute. Then we compared throughput and average latency difference. Fig. 2 and 3 show throughput (Mb/s) and average delay (ms), respectively.

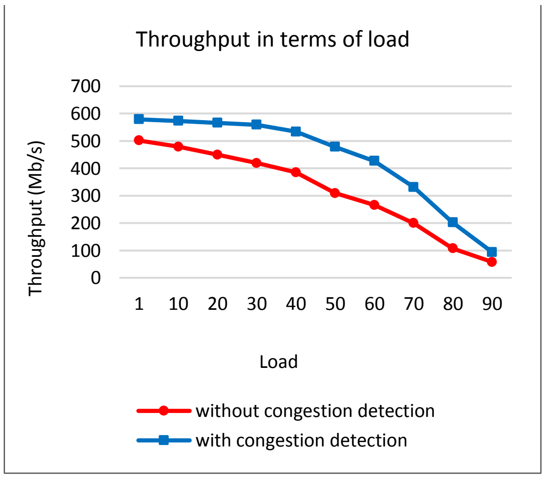

Fig. 2. Throughput considering different amounts of load on the network.

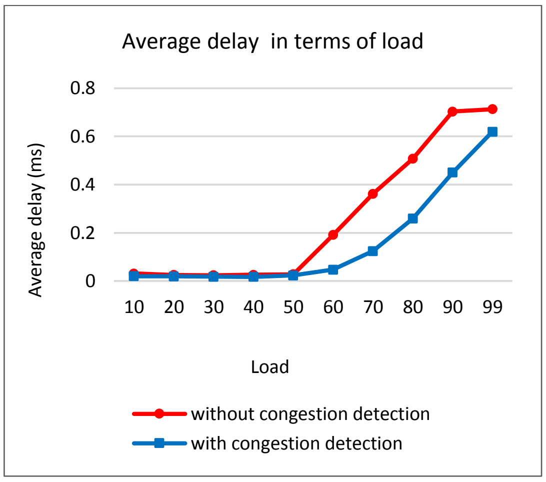

Fig. 3. Average delay with load size.

It must be noted that the computed delay is not two-way and just the time difference between the source and destination is considered. In addition, since characteristics of the links are considered to be the same, the average is measured as point-to-point for the entire delays.

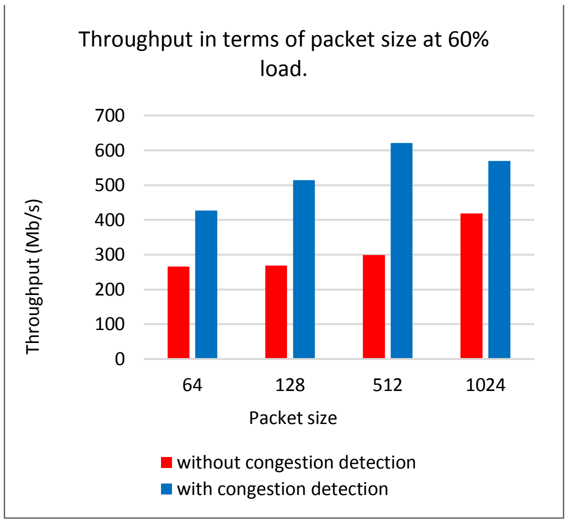

Fig. 4. Throughput for different packet sizes at 60% load.

Fig. 4 indicates the throughput for different packet sizes where load percentages are maintained at 60%, while the packet size varies in the range of 64 to 1024 byte.

B. Experiment 2:k=4 fat-tree network

In this scenario, we ran parameters similar to previous scenario with only the packet size considered to be 1024 byte. Then we compared network behavior in this scenario. As Fig. 5 and Fig. 6 show, there is a significant difference between the two scenario namely with increasing packet size, our method simply acts on congestion conditions.

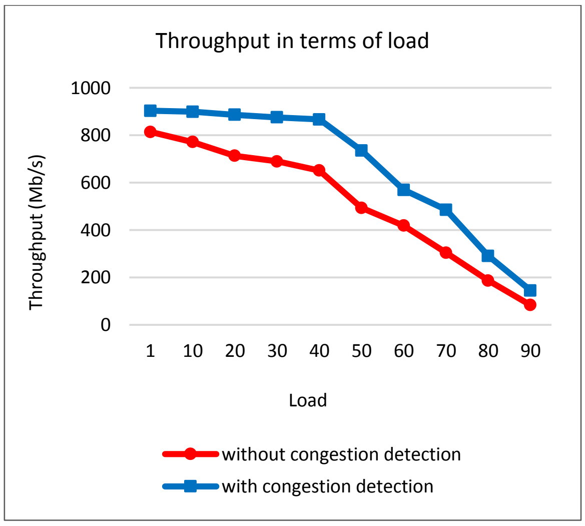

Fig. 5. Throughput considering different amounts of load on the network.

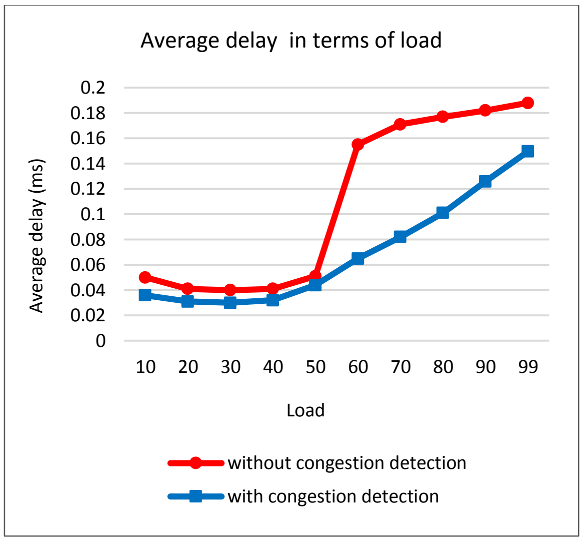

Fig. 6. Average delay with load size

IV. Discussion

This study set out with the aim of assessing the importance of congestion control using SDN in the data center network. The most obvious finding to emerge from the analysis is that by increasing packet size efficiency proposed method the performance of the network did not reduce. One of the reasons for the phenomenon is that the line space is considered rather independent from the packet size, where the increase in packet size results in less congestion. So, the effect of congestion control procedure is less noticed. Nevertheless, the proposed method not only does not reduce the network efficiency, but also indicates a considerable rise in load peak.

Another important finding was that the congestion delay, whereby the packet size increases, the overall delay is reduced and more delayed congestion occurs. Thus, the enhanced efficiency of the network is observed using the proposed method at higher percentages of load insertion on the network.

The findings of the current study do not support the previous research. However, in the view of qualitative data we attempt to compare the proposed method with the previous research such as QCN. The previous research attempted to control congestion through source using rate transmission reduction whereas our proposed method centrally deals with the problem with congestion by the use of OpenFlow switch without involving hosts. In the following, we discuss and compare advantages and disadvantage of this method with our proposed method in the following aspects [3].

1) *Overhead:* The overhead of QCN is fairly low and unpredictable. Whilst the overhead of proposed method is low and predictable namley the sizes of messages for requesting and replying port statistics on each port are 8 bytes and 104 bytes respectively [16].

2) *Rate of convergece to fair state:* QCN is slow in convergence to fair state due to AIMD- like algorithms can obtain fairness just in the long term. But the proposed method can reach the perfect fair state owing to centralization.

V. Conclusion

In the present paper, we introduced congestion control for software defined data center network. Our proposed method considered using significant features of SDN in order to optimize performance of the network. We analyzed the effectiveness of our approach by the use of a range of tests, showing that this method has potential to improve particularly in the throughput enhancement and average packet delay reduction. Another key point of the proposed method is its higher generalization potential, which originated from the minimum possible assumption on the problem conditions. This capability of the system results in the simple integration of the proposed method to the data centers SDN platforms.

Further research could usefully exploit machine learning methods for improvement of the detection and management components of our scheme. Moreover, the proposed method could examine in the large scale of the network using distributed controller due to overhead on the single controller.

Acknowledgment

The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] J. R. Santos, Y. Turner, and G. Janakiraman, “End-to-end congestion control for infiniband,” in INFOCOM. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, vol. 2, pp. 1123–1133, 2003.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” in ACM SIGCOMM Computer Communication Review, vol. 38, no. 4, pp. 63–74, 2008.

[3] Y. Zhang and N. Ansari, “On architecture design, congestion notification, TCP incast and power consumption in data centers,” Communication. Survey. Tutorials, IEEE, vol. 15, no. 1, pp. 39–64, 2013.

[4] IEEE. 802.1 - Data Center Bridging Task Group. Available at: http://www.ieee802.org/1/pages/dcbridges.html/.

[5] T. Benson, A. Akella, and D. A. Maltz, “Network traffic characteristics of data centers in the wild,” in Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, pp. 267–280, 2010.

[6] IEEE. 802.1Qbb, “Priority-based Flow Control,” 2011. [Online]. Available: http://www.ieee802.org/1/pages/802.1bb.html.

[7] IEEE. 802.1Qau, “Congestion Notification.” [Online]. Available: http://www.ieee802.org/1/pages/802.1au.html.

[8] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, “Portland: a scalable fault-tolerant layer 2 data center network fabric,” in ACM SIGCOMM Computer Communication Review, vol. 39, no. 4, pp. 39–50, 2009.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling Innovation in Campus Networks,” SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Mar. 2008.

[10] H. Kim and N. Feamster, “Improving network management with software defined networking,” Communications Magazine, IEEE, vol. 51, no. 2, pp. 114–119, 2013.

[11] Y. Jarraya, T. Madi, and M. Debbabi, “A survey and a layered taxonomy of software-defined networking,” Communications Surveys Tutorials, IEEE, vol. PP, no. 99, pp. 1–1, 2014.

[12] Mininet. http://mininet.org/.

[13] Floodlight. http://www.projectfloodlight.org/floodlight/.

[14] L. Feng, L. Dongmei, and C. Weihong, “Improved Dijkstra Algorithm Based on Quad Heap Priority Queue and Inverse Adjacent List,” J. Image Graph., vol. 12, p. 7, 1999.

[15] D. A. S. T. D. at the National Laboratory for Applied Network Research (NLANR). Iperf. http://iperf.sourceforge.net/, Mar 2012.

[16] OpenFlow Switch Consortium, “OpenFlow switch specification, version 1.4.”, 2013.