Using SDN Congestion Controls to Ensure Zero Packet Loss in Storage Area Networks

Shie-Yuan Wang, Li-Min Chen, Shih-Kai Lin, and Liang-Chi Tseng
Department of Computer Science
National Chiao Tung University
Hsinchu, Taiwan
shieyuan@cs.nctu.edu.tw

***Abstract*—Today, storage area networks (SAN) are widely used in data center networks to connect servers to data storage devices. In such a network, ensuring zero packet loss is very important for a server's network file system (NFS) to quickly access data across the network. Otherwise, the time spent on detecting packet losses and then retransmitting the lost data will severely harm the data access throughput and delay performance of the network file system.**

Recently, the SDN technology has been widely used in data center networks. In an SDN network, the SDN controller has a global view of the states of the overall network. Exploiting this capability, we designed SDN-assisted rate-based and credit-based congestion control schemes to ensure zero packet loss in the network. We used both greedy traffic and bursty NFS traffic to compare the performances of the two schemes under many different situations. In this paper, we report their relative strength and weakness and show that the SDN-assisted credit-based scheme can ensure zero packet loss while maintaining a high link utilization.

I. Introduction

Today, storage area networks (SAN) are widely used in data center networks to connect servers to data storage devices. To increase the efficiency of data transmission over SAN, Fiber Channel over Ethernet (FCoE) [2] [3], a modern technology used in SAN, transmits data directly over Ethernet without using the IP and TCP headers [4]. Because TCP is not used in FCoE to detect packet losses and retransmit them, FCoE assumes a lossless Ethernet. To guarantee zero packet loss, Data-Center-Bridging-capable (DCB) switches [5] [6] need to be used in SAN to support FCoE. SAN can use other technologies such as iSCSI [7] to transport data. Since iSCSI uses IP and TCP protocols, TCP can be used to detect packet losses and then retransmit the lost data without assuming a lossless Ethernet. However, detecting packet loss and retransmitting take much time, which severely harm the data access throughput and delay performance of the network file system.

For the above reasons, providing zero packet loss in the network is very important for SAN. Although DCB-capable switches can achieve this goal by sending a PAUSE control packet to a sending switch to ask it to stop forwarding packets, they can only support class-of-service-based flow control between a sending switch and a receiving switch. Flow control is employed upon a class queue rather than a flow queue in the switch. Since there are only eight class queues supported and no flow queues are supported in a DCB-capable switch, stopping the packet forwarding of a class queue may cause the head-of-line blocking problem for some flows belonging to that class. (That is, packets of the flows that are not congested in the receiving switch are unnecessarily blocked in the sending switch from being forwarded to the receiving switch.)

To overcome this DCB problem, we designed SDN-assisted rate-based and credit-based congestion control schemes. The emerging SDN technology [8] is being used in data center networks. In an SDN network, the SDN controller has a global view of the states of the overall network. The controller knows how many flows are contending for the bandwidth of a bottleneck link and thus can appropriately allocate bandwidth to avoid congestion. The controller also knows the current buffer usage of a port, thus can temporarily stop some hosts from sending more packets to avoid buffer overflow. In the two propsed SDN-assisted schemes, the SDN controller controls the rates of the sending hosts or the credits (number of bytes) that a sending host can use to send data to ensure zero packet loss. Since the control is employed on the sending hosts rather than between neighboring switches (which is used in the DCB approach), our proposed schemes do not suffer from the head-of-line blocking problem with DCB-capable switches.

In the two proposed schemes, we assume that each host has a virtual OpenFlow switch, which is OpenVswitch (OVS) in our case [9], running to implement the control from the SDN controller. Currently, it is very common to run a virtual OpenFlow switch on a host to enable multiple virtual machines (VMs) running on the host to connect to the physical network. Therefore, this assumption can be easily fulfilled. To support the rate-based control, the OVS running on a sending host periodically receives commands from the SDN controller to adjust the sending rates of flows on that host. To support the credit-based control, the OVS running on a sending host requests credits from the SDN controller when it has data to send. Only when it receives enough credits can it send out its data. Effectively, our credit-based control scheme functions like a window-based control scheme that limits the amount of data that can be sent into the network at any moment.

In this paper, we evaluated the performance of the rate-based and credit-based schemes using greedy NFS traffic and bursty NFS traffic (both are transported over UDP). Greedy

978-3-901882-89-0 @2017 IFIP

NFS traffic is used to represent sending a very large file to the NFS server. Bursty NFS traffic is used to represent sending a directory with many small files to the NFS server. We logged the real NFS traffic between a NFS server and a NFS client to obtain the distributions of disk I/O time on the NFS server. The disk I/O time is the elapsed time from when the NFS server receives a NFS read/write command to when it has finished the command and sends back a NFS reply. We used the obtained realistic NFS traffic patterns in our experiments to make our evaluation results more realistic.

In the rest of paper, Section II discusses related work. Section III presents the design and implementation of the two proposed schemes. Realistic NFS traffic patterns are presented in Section IV. Experimental settings are explained in Section V. Section VI evaluates the performance of the two schemes. Lastly, the paper concludes in Section VII.

II. Related Work

Data Center Bridging (DCB) is a set of standards designed for data center environments to avoid packet losses due to buffer overflow. In IEEE 802.1Qbb, it defines Priority Flow Control (PFC) based on the PAUSE frame defined in IEEE 802.3x. PFC provides a link-level flow control mechanism that can be used to independently control the packet forwarding of each traffic class. This mechanism ensures zero packet loss under congestion in data center networks. Its pause frame mechanism allows the traffic of a specific class to be stopped without affecting other classes. However, because only eight traffic classes are allowed, flows belonging to the same class may suffer from the head-of-line (HOL) blocking. In our proposed SDN-assisted schemes, we eliminate this HOL problem.

In [11] [12], the authors proposed rate-based control schemes to control network congestion. In [13], the authors proposed a credit-based control scheme to avoid network congestion. In addition to these papers, in the past, several papers have proposed rate-based or credit-based control schemes. However, few of them are assisted by the SDN controller as the SDN technology emerged in just a few years.

There is a recent work [14] that used the SDN technology to deal with congestion in data centers. They used a rerouting mechanism to avoid transferring packets through bottleneck links. However, the direction of their work is different from ours. Although many related work also try to avoid network congestion, reduce queuing delay, and increase link utilization, they do not put "zero-packet-loss" as theie most important goal. In contrast, our SDN-assisted rate-based and credit-based control schemes strictly enforce "zero-packet-loss" in the network. While most related work were proposed for general networks where packet losses are acceptable, our schemes are specifically proposed for SAN, where packet losses should not occur.

III. Design and Implementation

In SAN, because multiple servers may access the storage device simultaneously, the single bottleneck scenario is the most common problem (i.e., all traffic merges at the switch connecting to the storage device.) Our credit-based scheme is designed for this kind of problem. Our design of rate-based and credit-based schemes involve three different entities — end hosts, switches, and the SDN controller. In the following, we present each of them in detail.

A. End Host

There are two kinds of end hosts: client and server. In our evaluations, multiple clients sent their data to the server simultaneously. The server measured the aggregate throughput and counted how many packets were lost (if there was any).

On each client, we run a OVS. To support the rate-based control, the SDN controller periodically sends this OVS a message to adjust the sending rates of all flows originating from that host. To support the credit-based control, when the OVS has accumulated enough data, it requests credits from the SDN controller to send out the data. Only when the OVS receives enough credits can it send out the data.

B. Switch

We assume that all switches are OpenFlow-enabled switches. The SDN controller gathers and computes information of every traffic flows such as their recent bandwidth usage, bandwidth usage of each flow on the bottleneck link and the current buffer occupancy of output ports. Such information is used to adjust the sending rates of traffic flows or control the number of credits that can be allocated to sending hosts.

C. SDN Controller

There are different ways to avoid network congestion. If there are multiple paths in a network, one can apply a rerouting mechanism [14]. However, when there is a bottleneck link that cannot be bypassed, how to avoid packet losses caused by buffer overflow on that link is important and it is the main topic in this paper.

The SDN controller knows the global state of the network and thus has the capability to ensure zero packet loss in the network. We proposed two kinds of SDN-assisted congestion control schemes: rate-based and credit-based.

1) Rate-based SDN Controller : In this design, each end host needs to use the transmission rates assigned from the SDN controller to send packets. The maximum sending rate cannot be higher than the assigned bandwidth at any moment. The SDN controller ensures that the total of all assigned rates on each link does not exceed the link bandwidth, so that no packets will be dropped due to buffer overflow.

Receiving PacketIn [1] messages sent from SDN switches for new arriving flows, the SDN controller knows how many traffic flows are contending for the bandwidth of each output port. By periodically collecting packet counter statistics, the SDN controller knows the current bandwidth usage and buffer occupancy of each output port. With this information, it computes the available link bandwidth along the path of the new flow. The controller then assigns the transmission rate to the sending host of the new flow. After that, the sending

host can start sending the data based on the rate. The SDN controller maintains the rates of every traffic flows. Thus it can quickly computes an initial sending rate for a new flow without causing network congestion.

Using a period of 100 ms, the controller periodically monitors the buffer occupancy of all output ports and adjusts the current assigned rates. If the buffer occupancy is less than a threshold (50 packets in our experiments), to maintain a high link utilization, the traffic flows passing the output port increase their transmission rates by an equal amount of 10 packets per period. On the other hand, if the buffer occupancy exceeds a threshold (100 packets in our experiments), to avoid buffer overflow, the traffic flows passing that output port decrease their transmission rates by 10 packets per period.

When a flow finishes its transmission, the sending host will notify the SDN controller so that the controller can equally allocate the released bandwidth to remaining flows. On the other hand, if a new flow's rate needs to be greatly increased from its initial sending rate to its fair share, the rates of existing flows should be decreased. In our scheme, the rate adjustment is based on the current bandwidth sharing percentages of existing flows. Suppose that existing flows should totally release a certain rate Y packet/sec to the new flow and a specific flow is responsible for x% of the total bandwidth usage of all existing flows, then this flow should decrease its current rate by Y*x%. Based on these designs, the link bandwidth can be dynamically re-allocated among a new set of flows.

2) Credit-based SDN Controller: The credit-based SDN controller strictly controls the number of packets in the network. End hosts must acquire credits from the SDN controller before they can send out data. One credit allows 1.4 KB of data to be sent into the network. The controller ensures that the total number of credits does not exceed the maximum buffer space of the bottleneck output port. Therefore, no packets will be dropped due to buffer overflow. For fairness, the SDN controller limits the maximum number of credits that a host can acquire.

Credits are one-time use only. When a sending host wants to send a data block, it must obtain enough credits first. For example, to send a 256 KB data block, about 183 credits should be acquired first. If a sending host runs out of credits, it must acquire enough credits from the SDN controller so that it can continue to transmit data. The number of credits in the SDN controller is limited. It is close to the maximum buffer space of the bottleneck output port. In our experiments, because the maximum buffer space of the bottleneck output port is set to 1,000 packets, the number of credits in the SDN controller is set to 950 in our experiments.

When there are no more credits to dispatch to sending hosts, the SDN controller will block all credit requests from hosts. Only after the bottleneck switch returns free credits to the SDN controller, can those blocked requests be granted. The time to return free credits is when packets leave the output ports of the switch. Our scheme uses the switch's de-queue counters, which count the number of packets forwarded so far, to calculate when and how many free credits can be returned to the SDN controller. In our design, every 10 ms, if there are free credits, the switch will send a message to the SDN controller to tell it how many free credits can be returned to it in the message. If no free credits can be released in this 10 ms period, no message needs to be sent to the SDN controller to save network bandwidth.

A drawback of credit-based scheme is that the flow sending rate is limited by how fast the sending host can get credits (i.e., credit delay). If the credit delay is too long, the sending rate may be affected. One reason for long credit delays is the long link RTT from the sending host to the SDN controller (i.e., control delay). Later on, we will show the relationship between the control delay and the maximum rate a sending host can obtain in the credit-based scheme.

IV. NFS Traffic Patterns

To make our experimental results realistic, we evaluated the performance of our SDN-assisted rate-based and credit-based schemes using NFS traffic patterns. Our end hosts are either NFS server or NFS clients. We obtained the NFS traffic patterns from real NFS traffic measured on the network of our department. There are two kinds of NFS traffic: the first one is transferring a large file (called "greedy NFS transfer" in the experiments) while the second one is transferring an entire directory with many small files (called "bursty NFS transfer" in the experiments). To derive the traffic patterns, when transferring files from a NFS client to the NFS server, we used tcpdump [15] to log packets. Then, we calculated the elapsed time between when the NFS server receives a NFS write command and when it sends back a NFS ACK packet. This elapsed time is the disk I/O time that the NFS server must wait before writing the received data into the disk.

The distributions of the disk I/O time of the two kinds of NFS traffic are different. If the NFS client transfers only one large file, the data of the file will be divided into many 256 KB large data blocks and the client will issue a write command for each of these blocks. In this case, most disk I/O time are very small and pretty much the same. This is because most data of a file are continuously placed on the disk. In contrast, when the NFS client transfers an entire directory with many small files to the NFS server, because the directory has many small files and sub-directories in it and they are not continuously placed on the disk, the disk I/O time varies a lot — some are up to 1,000 times larger than the others.

According to our NFS traffic measurements, the distribution of the disk I/O time when transferring a large file has the following characteristics: mean=0.13 ms, std=0.02 ms, max=11.7 ms, and min=0.000001 ms. On the other hand, the distribution of the disk I/O time when transferring a directory with many small files has the following characteristics: mean=30.5 ms, std=0.34 ms, max=134 ms, and min=0.003 ms. The distribution of the data block sizes when transferring a directory with many small files has the following characteristics: mean=8,818 bytes, std=13,047 bytes, max=263,065 bytes, and min=1 byte. In our experiments, we used these distributions to model the

disk I/O time of the NFS server and the transferred data block sizes when evaluating the performance of SDN-assisted rate-based and credit-based control schemes.

V. Experimental Settings

We used Mininet [16] as the SDN network emulator to emulate all hosts and OpenFlow switches. We ran Mininet emulations on a PC equipped with a 2.3 GHZ CPU and 4 GB RAM. Our SDN-assisted rate-based and credit-based controllers are implemented on the Floodlight [17] SDN controller. The network topology used in the experiments is shown in Figure 1. The bandwidth and delay of each link are set to 100 Mbps and 0.05 ms, respectively. At first, we set the bandwidth of each link to 1 Gbps. However, we found that at 1 Gbps link bandwidth Mininet generated incorrect performance results due to CPU overload. Therefore, we lowered the link bandwidth to 100 Mbps to enable Mininet to generate correct performance results.

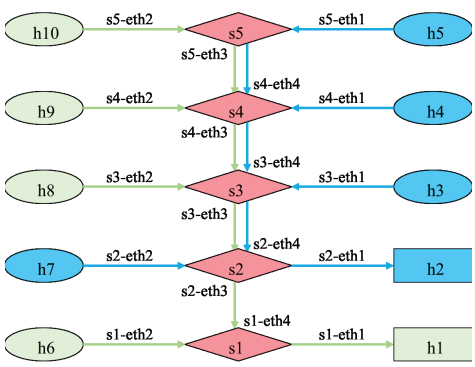


Fig. 1: The network topology used in experiments

In our experiments, there are two different cases: single-server and multi-server. In the single-server case, we used h1 as the server and h6, h8, h9 and h10 as the clients. In the multiple-server case, there are two groups of servers and clients and h1 and h2 are used as the server in these groups, respectively. The used traffic patterns in the experiments are of two kinds. The first kind is greedy NFS transfer and the second kind is bursty NFS transfer. Their characteristics have been presented in Section IV.

VI. Performance Evaluation

A. Single-Server Case

We considered the setup with h1 as the server and h6, h8, h9 and h10 as the clients. Under this setup, we measured the performance when multiple clients contended for the bandwidth of the bottleneck link, which is between s1 and h1. In all of our experiments, we found that no packet losses ever occurred due to buffer overflow in the switch. That is, both of our rate-based scheme and our credit-based scheme achieved the goal of zero packet loss in the network.

1) All Greedy NFS Traffic: In this experiment, each of the four clients launched a greedy NFS transfer to the server (without disk I/O time on the NFS server). Each client have different start time. From client 1 to client 4, every client starts transmitting data at a 5-second spacing and transfers 500 MB (greedy), 200 MB (greedy), 100 MB (greedy) and 50 MB (greedy) data, respectively.

We measured the achieved bandwidth of each client and their summation over time. Figure 2 shows the results when the rate-based controller was used. One can see that 1) later arriving flows gradually increased their sending rates until they reached their fair share with existing flows. The speed to reach their fair shares was low due to the used additive-increase design to strictly avoid buffer overflow. 2) when existing flows finished their transfers at around 50th and 60th second, the bandwidth used by them was released and used by other existing flows, and 3) the total achieved bandwidth remained high when flows entered or left the network.

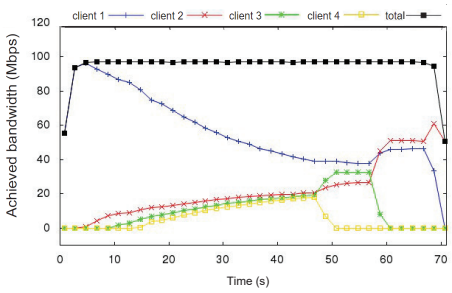


Fig. 2: Rate-based controller with greedy NFS traffic

Figure 3 shows the results when the credit-based controller was used. One can see that link bandwidth was allocated fairly between competing flows and the competing flows reached their fair shares more quickly than in the rate-based scheme. Besides, the total achieved bandwidth also remained high when flows entered or left the network. Comparing the two figures, one can see that under the greedy NFS traffic, the credit-based scheme performs better than the rate-based scheme in the speed of bandwidth re-allocation.

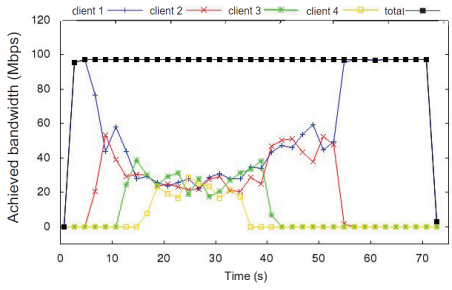


Fig. 3: Credit-based controller with greedy NFS traffic

2) Greedy and Bursty NFS Traffic: In this experiment, one client launched a greedy NFS transfer (without disk I/O time on the server) and each of the remaining three clients launched

a bursty NFS transfer (with disk I/O time on the server). Every client, from client 1 to client 4, also starts transmitting at a 5-second spacing and has a transfer size of 750 MB (greedy), 12 MB (bursty), 10 MB (bursty) and 5 MB (bursty), respectively.

Figure 4 shows the results for the rate-based controller. Initially, when there was only one greedy NFS transfer, the link utilization was high. However, when bursty transfers started to emerge, the link utilization dropped dramatically. The reason is that the rate-based scheme allocated bandwidth equally to the four competing flows. However, due to the disk I/O time on the NFS server, the clients launching bursty transfers cannot send their data fast enough to fully utilize their assigned bandwidth. As a result, they were idle most of the time and greatly wasted their assigned bandwidth.

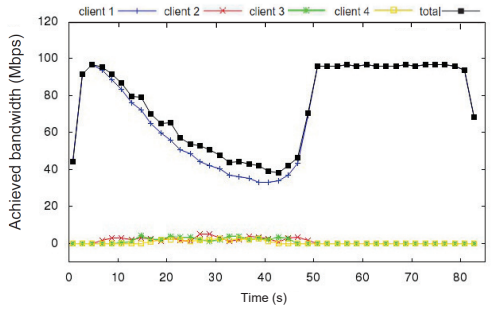


Fig. 4: Rate-based controller with greedy & bursty traffic

In contrast, Figure 5 shows the results for the credit-based controller. As can be seen, the most significant difference compared with the rate-based scheme is that the link utilization remained high when bursty NFS transfers were present in the network. The reason is that the credit-based controller allocated credits (chances to send packets, i.e., bandwidth) to a client only when the client has data to send. As a result, clients launching bursty transfers will not waste bandwidth. This explains why the credit-based scheme performed much better than the rate-based scheme in the link utilization.

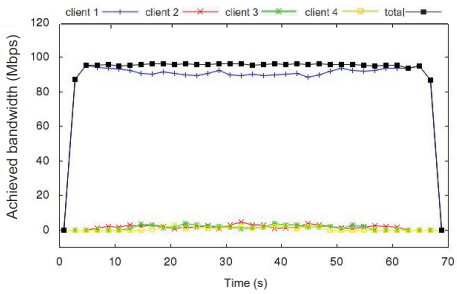


Fig. 5: Credit-based controller with greedy & bursty traffic

3) All Bursty NFS Traffic: In this experiment, each client launched 5 bursty NFS transfers (with disk I/O time on the NFS server). Figure 6 and Figure 7 show the achieved bandwidth of each client under the rate-based scheme and the credit-based scheme, respectively. Comparing the two figures, one can see that the credit-based scheme allocated bandwidth to NFS bursty clients very quickly. As a result, they finished their transfers more quickly. In contrast, NFS bursty clients obtained bandwidth more slowly in the rate-based scheme and the finish time of their transfers were postponed to a later time.

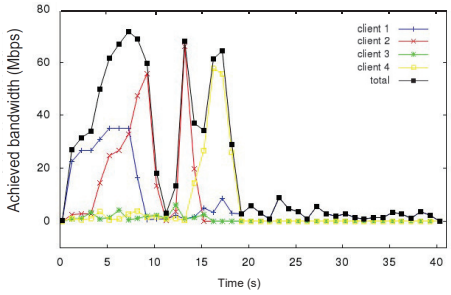


Fig. 6: Rate-based controller with bursty NFS traffic

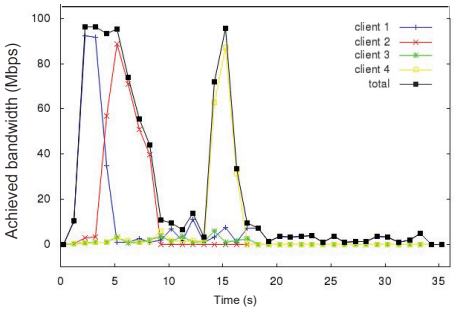
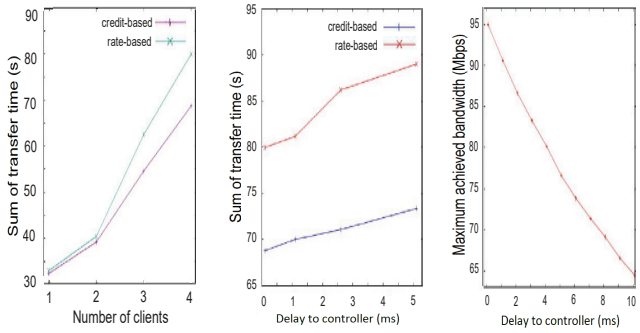


Fig. 7: Credit-based controller with bursty NFS traffic

In the experiment, we recorded the time that each client needed to finish its transfers and sum up their transfer time. We also varied the number of clients to compare their total transfer time between the rate-based and credit-based schemes. Figure 8(a) shows the total transfer time under different schemes. One can see that the total transfer time in the credit-based scheme was smaller than in the rate-based scheme. This shows that the credit-based scheme could more efficiently use the bandwidth than the rate-based scheme.

4) Effects of Different Control Delays: The control delay between a switch and the SDN controller is a key factor of performance in an SDN network. The control delay is the RTT between them, which depends on the link delay. In this experiment, we evaluated how different control delays would affect the performance of the two schemes. We let four clients launched bursty NFS transmission and set the extra one-way control delay to 1 ms, 2.5 ms and 5 ms, respectively. (Note: the default one-way link delay introduced by Mininet is tiny and is only 0.05 ms.) As shown in Figure 8(b), the total transfer time in both the rate-based scheme and the credit-based scheme increased as the control delay increased.

In the following experiment, we let a client launch a greedy NFS transmission to the server. Figure 8(c) shows the maximum bandwidth achieved by the client in the credit-based scheme, under different control delays. We can find that the maximum bandwidth the client can achieve decreases as the



(a) (b) (c)

Fig. 8: 8(a) Sum of transfer time under different number of clients 8(b) Relationship between control delay and sum of transfer time. 8(c) Relationship between control delay and maximum bandwidth in the credit-based scheme.

control delay increases. The reason is that the time to acquire credits for the client to send data is increased under higher RTT situations.

B. Multiple-Server Case

Here we evaluate the performance of the two groups in a multi-server environment. As stated before, there are two groups of clients and servers. The server of the first group is h1. The start time and transfer sizes of its clients are:

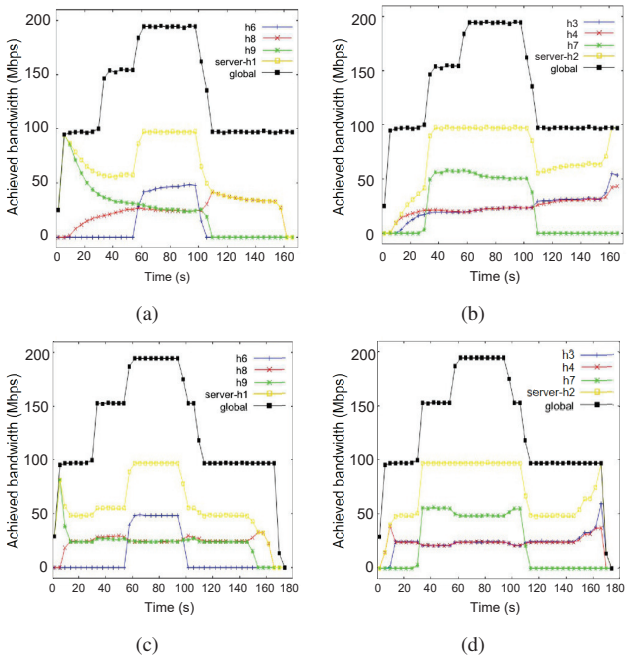
- h9 : starts at 1st sec, 500 MB, greedy
- h8 : starts at 8th sec, 500 MB, greedy
- h6 : starts at 55th sec, 250 MB, greedy

The second group uses h2 as the server and the start time and transfer sizes of its clients are:

- h4 : starts at 5th sec, 500 MB, greedy
- h3 : starts at 10th sec, 500 MB, greedy
- h7 : starts at 30th sec, 500 MB, greedy

Figure 9(a) and Figure 9(b) show each client's achieved bandwidth over time in the two groups respectively when using the rate-based controller. On the other hand, Figure 9(c) and Figure 9(d) show the results of the credit-based controller. Compared to the evaluations of the single-server case, a new performance metric, "global", is added into the graph. This metric is the summation of the achieved bandwidth of all clients from the two groups. With this metric, we can compare the overall bandwidth utilization of the rate-based controller to that of the credit-based controller. Comparing the four figures, we can see that both control schemes achieve almost the same overall bandwidth utilization.

In this test, the crucial bottleneck link is the link from switch s3 to switch s2, where four flows competed for the link bandwidth. The four flows were from h3 to h2, from h4 to h2, from h8 to h1, and from h9 to h1. From the four figures, we can see that each flow eventually obtained a fair share of the bottleneck link bandwidth. However, as can be seen, the rate-based controller adjusted the bandwidth allocation of these competing flows much more slowly than the credit-based controller.



(a) (b) (c) (d)

Fig. 9: 9(a) The first group with the rate-based controller 9(b) The second group with the rate-based controller 9(c) The first group with the credit-based controller 9(d) The second group with the credit-based controller

From the results above, one can see the relative strength and weakness of the two control schemes. The credit-based controller can handle bursty NFS traffic very well. It achieves a high bandwidth utilization and good fairness. However, the achieved bandwidth may slightly decrease when the control delay increases. On the other hand, although the rate-based controller cannot promptly achieve the maximum bandwidth utilization under bursty traffic load, the control delay does not affect its performance much. Despite these problems, both types control schemes can successfully meet the most important goal in SAN — zero packet loss in the network.

VII. Conclusion

In this paper, we proposed SDN-assisted rate-based and credit-based control schemes for storage area networks (SAN). An important requirement of SAN is ensuring zero packet loss in the network to provide high data access performance. To ensure zero packet loss, we implemented the two schemes in the Floodlight SDN controller and evaluated their performance with realistic NFS traffic patterns.

We found that the credit-based scheme outperforms the rate-based scheme under NFS traffic patterns in allocating bandwidth and maintaining a high link utilization. Although the performance of the credit-based scheme may suffer from a large RTT to the SDN controller, since a SAN is a local area network with small link delays, this problem is not serious.

REFERENCES

[1] Open Networking Foundation, "OpenFlow Switch Specification Version 1.5.0", https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf, Dec. 2014

[2] Claudio DeSanti and Joy Jiang, "FCoE in Perspective," ACM ICAIT (International Conference on Advanced Infocomm Technology), July 2008, Shenzhen, China

[3] Yueping Cai, Yao Yan, Z. Zhang and Yuanyuan Yang, "Survey on Converged Data Center Networks with DCB and FCoE: Standards and Protocols," IEEE Network Vol. 27, Issue 4, Page 27-32, July-August 2013.

[4] Joy Jiang and Claudio DeSanti, "The Role of FCoE in I/O Consolidation," ACM ICAIT (International Conference on Advanced Infocomm Technology), July 2008, Shenzhen, China

[5] Brent Stephens, Alan L. Cox, Ankit Singla, John Carter, Colin Dixon and Wesley Felter, "Practical DCB for Improved Data Center Networks," IEEE INFOCOM, April 2014, Toronto, Canada.

[6] "IEEE. 802.1 - Data Center Bridging Task Group," available: http://www.ieee802.org/1/pages/dcbridges.html.

[7] 'Internet Small Computer Systems Interface (iSCSI)," RFC 3720.

[8] "Software-Defined Networking: The New Norm for Networks," a white paper of Open Networking Foundation, April 13, 2012.

[9] "Extending Networking into the Virtualization Layer," B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker. Proc. of workshop on Hot Topics in Networks (HotNets-VIII), 2009

[10] "IEEE. 802.3x," available: https://standards.ieee.org/findstds/standard/802.3x-1997.html.

[11] Douglas E. Comer and Rajendra S. Yavatkar, "A Rate-Based Congestion Avoidance and Control Scheme for Packet Switched Networks," IEEE Distributed Computing Systems, May 1990, Paris, France

[12] S. R. Subramanya, Mingsheng Peng and J. Sarangapani, "Rate-Based End-to-End Congestion Control of Multimedia Traffic in Packet Switched Networks," IEEE ITCC (International Conference on Information Technology: Coding and Computing [Computers and Communications]), April 2003, Las Vegas, USA

[13] Shu-juan Wang and Man-gui Liang, "A Credibility-based Congestion Control Method," IEEE WiCom (Wireless Communications, Networking and Mobile Computing), September 2009, Beijing, China

[14] Masoumeh Gholami and Behzad Akbari, "Congestion Control in Software Defined Data Center Networks through Flow Rerouting," IEEE ICEE (International Conference on Electrical Engineering), May 2015, Boumerdes, Algeria

[15] "Tcpdump," available: http://www.tcpdump.org.

[16] "Mininet," available: http://mininet.org.

[17] "Floodlight," available: http://www.projectfloodlight.org/floodlight.