Proactive SDN-based Load Balancing for Datacenter Network

Minhyeok Kang, Hyunwoo Lee, Junghwan Song, Ted "Taekyoung" Kwon
Seoul National University
{mhkang,hwlee2014,jhsong}@mmlab.snu.ac.kr,tkkwon@snu.ac.kr

ABSTRACT

Most datacenter networks use ECMP that utilizes static hashing of flows to distribute network traffic to avoid congestion. However, it turned out that ECMP cannot achieve ideal distribution; thus, network congestion is still an important issue in datacenter networks. Recent proposals for load balancing reactively address the problem; thus, they incurs performance delay. Furthermore, they have deployment issues.

We propose a proactive SDN-based load balancing scheme that can proactively avoid congestion using the knowledge of the network state and that can be easily deployable. We evaluate our scheme on a Mininet emulator, which shows that our scheme achieves 2 times shorter flow completion time than ECMP in the emulation environment and shows more even traffic distribution. We are currently applying our scheme to the 100Gbps ethernet testbed to show its feasibility in the real world.

CCS CONCEPTS

• **Networks** → *Data center networks.*

KEYWORDS

load balancing, software defined networking

ACM Reference Format:
Minhyeok Kang, Hyunwoo Lee, Junghwan Song, Ted "Taekyoung" Kwon. 2019. Proactive SDN-based Load Balancing for Datacenter Network. In *Proceedings of CFI 2019.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/1234567890

1 INTRODUCTION

Datacenter networks must provide large bisection bandwidth to support an ever-increasing diverse set of workloads, ranging from latency-sensitive tiny flows to throughput-sensitive elephant flows. A Clos topology (a.k.a., a fat-tree topology) with the the Equal-Cost Multi-Path (ECMP) load balancing scheme is known to provide large bisection bandwidth [1, 2]; thus, it is widely deployed in datacenter networks. ECMP, however, provides uneven load distribution due to hash collisions leading to poor performance [2].

There have been several load balancing schemes [2, 3, 6] to address the limitation of ECMP. However, they require delay time due to their reactive behavior or have deployment challenges. For example, Hedra [2] re-routes flows after a congestion is discovered in the network. Although it reduces the impact of the congestion, it requires some delay time such as calculating a new path on congestion. On the other hand, MPTCP [6] is challenging to deploy since they require changes of kernel network stack. CONGA [3] requires replacing every network switch with specialized one that implements a new load-balancing algorithm.

We present a proactive and easy-to-deployable load balancing scheme based on Software Defined Networking (SDN). Our scheme proactively distributes traffic loads as evenly as possible to avoid congestion and, thus, maximizes overall link utilization to support large bisection bandwidth. Also, our scheme can be easily employed with any OpenFlow-enabled switches that are widely deployed in the current datacenter networks. We implement our solution with the Ryu SDN framework [4], emulate it with the Mininet emulator and conduct experiments in the 100Gbps ethernet testbed. Emulation result shows our load balancing scheme is feasible while showing better performance than ECMP and testbed result shows our scheme can achieve more even traffic distribution than ECMP.

2 PROACTIVE LOAD BALANCING WITH SDN

2.1 Design Goals

We consider the following properties to design our solution.

- **Proactivity** It is desirable to avoid a congestion proactively since the cost to address the congestion reactively is expensive.
- **Deployability** The solution should be easily deployed. It is irrelevant to require replacing any network commodities or modifying the kernel stack.

To achieve the above goals, we leverage a software defined networking (SDN) with our distribution algorithm. The SDN controller can get information about network traffic, e.g., TX/RX bytes on each switch port, from switches via OpenFlow channels. Based on the information, we can proactively detect possible congestions and decide good routes for flows. Note that use of SDN technology in data centers is growing [5], which makes our scheme immediately deployable in practice.

2.2 SDN Controller Behavior

Our SDN controller consists of the two basic components, namely *Monitor* and *Balancer*. Monitor keeps track of the status of the overall network by periodically querying information about incoming and outgoing traffic bytes to all the switches in the network. Balancer selects a good path for each new flow based on the network status. We describe the behavior of each component in the following.

Monitor. Monitor periodically, say 1 second, queries information about link loads including TX/RX bytes on switch ports to all the switches. Monitor updates the status of the network according to the reports from the switches and calculates link loads by:

CFI 2019, August 7 - 9, 2019, Phuket, Thailand

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/1234567890

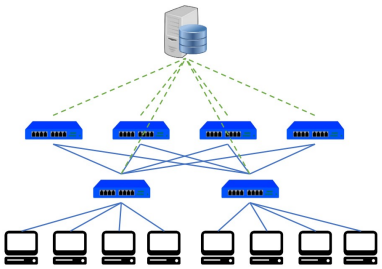

Figure 1: The network topology used for our evalution is shown. We build a Clos topology that consists of 4 spine switches and 2 leaf switches.

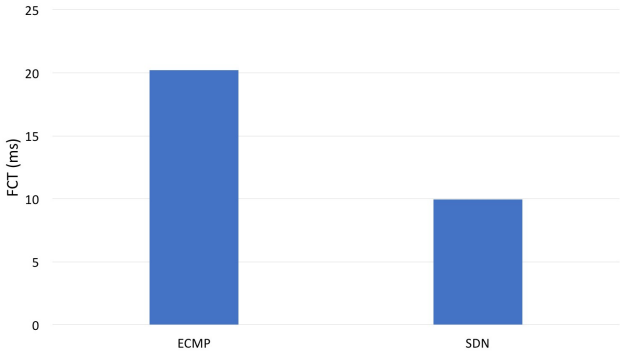

Figure 2: The emulation result shows our scheme only requires 49.20% of FCTs compared with ECMP.

$$\frac{\text{(TX bytes in the current period)}-\text{(TX bytes in the last period)}}{\text{(Capacity of the switchport)}}$$

Balancer. Balancer distributes the traffic by assigning the path per each flow. The procedure of Balancer is as follows:

(1) A switch sends a request to Balancer when it receives a packet of a new flow generated by an end-node.
(2) Then, Balancer checks the current link loads of possible paths for the requested flow to find the link with minimum load.
(3) Balancer responds with the egress port corresponding to the link found in the previous step to the switch.
(4) The switch adds an entry for the flow into the flow table.

3 EVALUATION

We evaluate our scheme by emulation. Figure 1 shows the network topology used in our evaluation. We use 2-tier Clos topology that consists of four spine switches and two leaf switches, while all the switches are attached to the SDN controller. Four machines are connected to each leaf switch, respectively; thus, each machine has four paths toward a machine in another rack. We use the Ryu SDN framework [4] to implement the SDN controller and the polling period is set to 1 second.

We use Mininet emulator in which all the links have 1Gbps bandwidth. We generate four flows from a machine in a leaf switch to a machine in another leaf switch and measure flow completion times (FCTs) for each flow. The total volume of traffic sent by each sender is 1GB.

Our emulation result is shown in Figure 2. Note that our scheme only requires 49.20% of FCTs, compared with ECMP. In ECMP, there happened lots of hash collisions; thus, two or three flows may share the same path, which incurs congestion. On the other hand, each flow utilizes its own path without sharing with others, which accounts for our result.

4 FUTURE WORK

We are applying our scheme to the 100Gbps-ethernet testbed. Our initial result is shown in Figure 3. In this experiment, we generate two 10GB flows from the source to the destination. We find that the

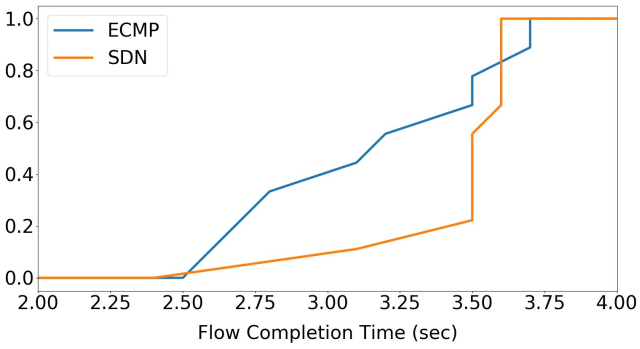

Figure 3: The testbed based result shows our scheme achieves more balanced distribution than ECMP.

distribution of FCT of our scheme is more evenly distributed than that of ECMP. Though the average FCT of our scheme (3.39s) is slightly slower than that of ECMP (3.13s), the standard deviation of FCTs of our scheme is 17.7% smaller than that of ECMP. We believe that the per-flow overhead of Balancer causes the performance degradation shown in the experiments, but we're still figuring out the exact cause and leave it as future work.

REFERENCES

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *SIGCOMM*. ACM.
[2] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, Amin Vahdat, et al. 2010. Hedera: dynamic flow scheduling for data center networks.. In *NSDI*.
[3] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Francis Matus, Rong Pan, Navindra Yadav, George Varghese, et al. 2014. CONGA: Distributed congestion-aware load balancing for datacenters. In *SIGCOMM*. ACM.
[4] Ryu SDN Framework Community. [n. d.]. Ryu SDN Framework. Retrieved "June 29, 2019 from "https://osrg.github.io/ryu/"
[5] Lightwave Staff. [n. d.]. IHS Markit: Data Center and enterprise LAN SDN market totaled $4.4 Billion in 2017. Retrieved "June 29, 2019" from "https://techblog.comsoc.org/2018/06/11/ihs-markit-data-center-and-enterprise-lan-sdn-market-totaled-4-4-billion-in-2017/"
[6] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. 2011. Design, Implementation and Evaluation of Congestion Control for Multipath TCP.. In *NSDI*.