Per-Packet Load Balancing in Data Center Networks

Yagiz Kaymak and Roberto Rojas-Cessa

***Abstract*—In this paper, we evaluate the performance of per-packet load balancing in data center networks (DCNs). Throughput and flow completion time are considered among the main metrics to evaluate the performance of the transport of flows over the presence of long flows in a DCN. Load balancing in a DCN may benefit those performance metrics but also it may generate out-of-order packet delivery. We investigate the impact of out-of-order packet delivery on the throughput and flow completion time of long and short flows, respectively, in a DCN. Our simulations confirm the presence of out-of-order packet delivery in a DCN using per-packet load balancing. Simulation results also reveal that per-packet load balancing may yield smaller average flow completion time for short flows and larger average throughput for long flows than the single-path transport model used by TCP despite the presence of out-of-order packet delivery. As the delay difference between alternative paths decreases, the occurrence of out-of-order packet delivery in per-packet load balancing also decreases. Therefore, under the studied scenarios, the benefits of the per-packet load balancing prevail.**

***Index Terms*—data center network, load balancing, multipath forwarding, flow completion time, TCP.**

I. Introduction and Motivation

The interconnected servers of a data center run a wide variety of applications and services, which range from web search and social networking [1] to distributed file systems [2], [3]. Some of the underlying data distribution technologies are based on Hadoop [2] and Google File System [3]. While a web search entails a small amount of data to be transmitted, distributed file systems may require large data transfers between the servers of a data center [4]. Most applications/services in a data center require multiple servers working in parallel to assemble a response for a requested task. For instance a Hypertext Transfer Protocol (HTTP) request for a Facebook page requires a large number of servers, called workers, across the data center to be involved in the process of fetching partial results to generate the web page [1].

The performance of data centers is directly related with how fast the user-initiated requests are processed. Such a performance measure requires to generate the responses to requests as fast as possible [5]. This is where the intercommunication among the servers impacts the achieved performance of a data center. The communication between data-center workers takes place as flows, where a flow may be defined as a sequence of packets sent from a particular source to a particular destination [6]. Each worker is responsible for generating a small portion of the final result for a submitted task. The partial results are put together as a final result by an aggregator server. The distribution of the portions of a task to different workers and fetching the results requires a fast transmission of flows [7].

Y. Kaymak and R. Rojas-Cessa are with the Networking Research Laboratory, Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102. Email: yk79, rojas@njit.edu.

The time required to transmit a flow is referred to as the Flow Completion Time (FCT) [8], [9]. This parameter may be considered one of the most important metric for measuring the performance of the transport of short flows in data center networks (DCNs). Short flows have a size of a few kilobytes and they are associated with users' tasks that generate requests and responses between servers. On another hand, the performance of the transport of long flows is measured by their achieved throughput. Long flows, or background flows, have a size of several megabytes and they carry the data for the operation and maintenance of the data center. Therefore, long flows must be transferred with equal to or larger throughput than an acceptable minimum [4]. A high-performance DCN architecture in combination with the transport protocol used in it must simultaneously achieve small FCTs for short flows and high throughput for longs flows. In this paper, we focus on the performance metrics for both types of flows.

Congestion in the DCN is a factor that may impair achieving small FCT and high throughput. DCNs are generally provisioned with multiple paths between source-destination pairs to provide reliability and large bisection bandwidth. Therefore, link or path congestion may be circumvented by efficiently using the available multiple paths of a DCN. However, if a DCN with multiple paths is not accompanied with a multipath routing/forwarding mechanism, alternative paths may be underused and this phenomenon might lead to underutilization of the network. For instance, the use of the Transmission Control Protocol (TCP) and a shortest path routing algorithm may not be able to use the multiple paths in a DCN and the utilization of the DCN may be low.

This paper is motivated by the benefits of using a multipath routing/forwarding mechanism on a DCN and the effect it may have on the FCT for short flows and the throughput for long flows. Based upon the potential benefits of multipath forwarding schemes, we implemented a high-granularity (i.e., packet-based), low-complexity per-packet load balancing scheme based on Random Packet Spraying (RPS) [10], using NS-3 [11]. The load balancing scheme exploits the multipath feature of a DCN to balance the traffic among alternative paths, decrease the FCT of short flows, and increase the throughput of long flows. In this paper, we also unveil how the FCT and throughput of flows are affected by background traffic in the DCN.

TCP is a widely used transport protocol in the Internet and in DCNs [12], [13]. We adopt TCP as the transport mechanism in the load balancing scheme to keep the transport layer as simple as possible. However, because the use of packet-based multipath forwarding and the existence of possible delay differences between alternative paths, out-of-order packet delivery may occur at the receiver and TCP may react to these out-of-order packets as congestion indicators. Therefore, we

study the occurrence of out-of-order packets in short flows and show how FCT and throughput are affected by those out-of-order packets.

The remainder of this paper is organized as follows. In Section II, we describe the operation of per-packet load balancing and give our simulation results. In Section III, we present related works and in Section IV, we conclude the paper.

II. Mechanism and Evaluation

In this section, we describe the operation of per-packet load balancing. We then present the simulation parameters and show the simulation results from a data center using a fat-tree-based DCN [14].

A. Mechanism

Per-packet load balancing works at each switch by randomly selecting one of the output ports to independently forward every incoming packet. The selection of the output port for an incoming packet considers only the equal-cost paths to reach the destination of the packet. If multiple equal-cost paths to the destination are available, one of them is randomly selected with equal probability and the packet is forwarded to that specific port. The use of this technique requires calculation of equal-cost shortest paths in advance, so that the per-packet load balancing scheme is able to distribute the traffic among them.

It is expected that equal-cost paths between a source-destination pair have similar queue build-ups and latencies [10]. Having almost equal latencies between these paths may not always be the case because the traffic pattern on the network has an impact on the latency difference between paths and the number of out-of-order packets. This claim is supported by our findings, as presented in Figures 4 and 7 of this paper.

We adopt a fat-tree topology with four pods as DCN in our study. Each pod hosts four servers, two edge, and two aggregation switches. We use four core switches to connect these pods in the core layer of the DCN. The fat-tree network used in our simulations is shown in Figure 1. The letter in the label of each switch in this figure indicates the layer it belongs to. Edge, aggregation, and core switches are labeled by letters E, A, and C, respectively. The number following the capital letter indicates the switch number. Servers are labeled by the letter S and a number.

We selected the fat-tree topology as the architecture of the DCN because this architecture has multiple paths between any two servers, which makes multipath forwarding and load balancing suitable to implement in DCNs. In general, a fat-tree network with k pods has $(k/2)^2$ shortest paths between any two hosts [14]. Moreover, a fat-tree network may provide full bisection bandwidth to clusters with a large number of servers [14].

B. Evaluation

We base our study on computer simulation. Each simulated switch in our topology represents a commodity switch with shallow buffer. We selected a buffer size equal to 1000 packets, where each packet is 1500-bytes long. Therefore, the switch's buffer capacity is set to 1.5 Mbytes, which is a suitable value to represent the buffer size of shallow-buffered switches [7]. Drop-tail is employed as the packet dropping mechanism when the buffer of a switch is full. The DCN uses point-to-point (p2p) links with a symmetric capacity of 1 Gbps. This means that every link is full-duplex and both directions have the same capacity and delay. The delay of each link is set to 25μs to represent a Round Trip Time (RTT) value of $12 \times 25\mu\text{s} = 300\mu\text{s}$ between any two servers in the DCN when all the links carry no traffic. [7]. Any two interfaces located each side of a p2p link have IP addresses from the same subnet. In other words, we created a different subnet for each p2p link that interconnects two nodes. To keep the routing complexity low at each switch, we implemented per-packet load balancing by enabling packet-based random routing as the routing function in NS-3. Per-packet random routing allows a switch to randomly select one of its output ports among the possible equal-cost shortest paths for each incoming packet. Therefore, random routing may enable each packet to follow a distinct shortest path. Our simulation considers the hop count as the primary cost metric to calculate the shortest path, although different metrics, such as one-way delay or the residual capacity of the path may also be used. As an example of how per-packet load balancing works, let's assume that a flow is created from S0 to S15, as shown in Figure 1. In this example, each packet is first forwarded by switch E0, where there are two possible next-hop switches, A0 and A1, with equal-cost shortest paths to the destination node, S15. If A0 is selected as the next switch, C0 or C1; if A1 is selected as the next switch, C2 or C3 may be selected as the core switches, which in turn provide four different equal-cost shortest paths (i.e., $S0 \rightarrow E0 \rightarrow A0 \rightarrow C0 \rightarrow A6 \rightarrow E7 \rightarrow S15$, $S0 \rightarrow E0 \rightarrow A0 \rightarrow C1 \rightarrow A6 \rightarrow E7 \rightarrow S15$, $S0 \rightarrow E0 \rightarrow A1 \rightarrow C2 \rightarrow A7 \rightarrow E7 \rightarrow S15$, and $S0 \rightarrow E0 \rightarrow A1 \rightarrow C3 \rightarrow A7 \rightarrow E7 \rightarrow S15$) to the destination. Although the core switches have only one downlink to forward each packet to the destination pod, per-packet load balancing at edge and aggregation layers enable different packets to use almost completely disjoint paths to the destination node.

In our evaluations, we tested two different scenarios: static and randomized source-destination pairs for all flows. The

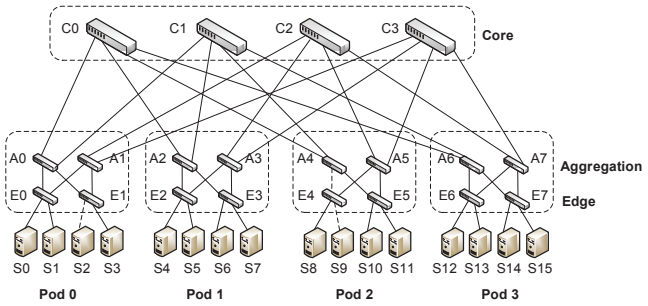

Fig. 1. A fat-tree DCN with four pods.

TABLE I
PARAMETERS USED IN SIMULATIONS

Parameter	Value
NS-3 version	ns-3.23
Line rate	1 Gbps
Sending rate for each interface	1 Gbps
Delay for each link	25μs
Long flow size	10 Mbytes
Short flow size	200 Kbytes
Packet size	1500 bytes
Queue size at each switch	1.5 Mbytes

static scenario consists of fixed source(s) and a destination server, and the randomized scenario consists of randomized source and multiple destination servers. Specifically, in the static scenario, sources generate from 0 to 5 flows, each with a size of 10 Mbytes, towards S15. Source and the destination nodes for long flows are fixed. This means that the destination node is always S15, and each source node is incrementally selected among S0, S1, S2, S3, and S4, where each server contributes with one long flow. For instance, if five long flows are generated, the traffic of these five flows would present a pattern resembling a TCP incast scenario [15]. In addition to long flows, we generate a 200-Kbytes short flow from S0 to S15 to show how the FCT and the percentage of out-of-order packets for short flow are affected. We also analyze the average throughput for long flows change under the presence of a different number of long flows. We stress out the downlink queue at switch E7 by increasing the number of long flows and we observe the change in the performance metrics. In the scenario with randomized source-destination pairs, we generated 20 short flows and a different number of long flows between randomly selected source-destination pairs. Except for the number of short flows and the randomized selection of source and destination servers, all other settings in the randomized source-destination pairs are similar to the static source-destination scenario. The reason why we use randomized source-destination pairs is to show how the per-packet load balancing mechanism affects the performance of flows under random background traffic. The transmission of all flows for both scenarios are initiated at the simulation time 0 and ended when all flows are completely transmitted. TCP is used as the transport protocol for all flows. The transmission speed of all generated flows is equal to the line capacity, 1 Gbps. Note that TCP is the only control mechanism on the sending rate and sender's TCP adjusts the transmission rate by reducing the size of the congestion window, if TCP detects any congestion on the network. Table I summarizes all the parameters used in the simulations. Each test is run 10 times and the average values are calculated.

Figure 2 shows the average throughput of the long flows for a different number of long flows is generated in the static scenario. As Figure 2 shows, per-packet load balancing provides a slightly larger average throughput for long flows than that achieved by the single-path forwarding of TCP, which we refer to as single-path forwarding. The achievable throughput of each flow is bounded by its fair share at the bottleneck link, E7-S15.

Figure 3 shows the average FCT of the short flow for a

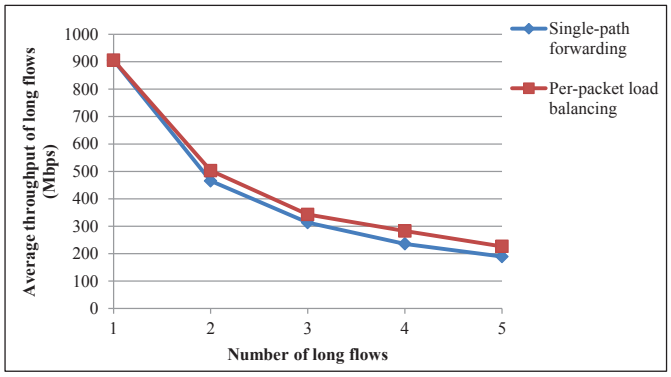

Fig. 2. Average throughput of the long flows for a different number of long flows in the static source-destination pair case.

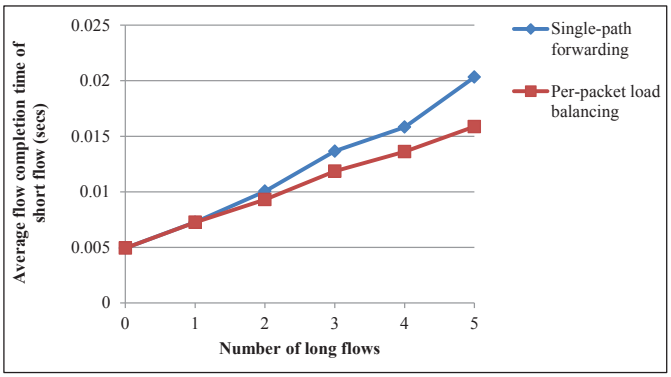

Fig. 3. Average FCT of the short flow for the static source-destination pairs.

different number of long flows for static source-destination pairs. The results in this figure show that the average FCT of the short flow is smaller than that of the single-path forwarding. The advantage becomes more noticeable as the FCT of the short flow is smaller. This phenomenon occurs because the short flow starts being affected by the increasing queuing delay caused by the long flows sharing the path with it. This effect is mitigated when per-packet load balancing is employed. The load distribution on multiple links decreases the queuing delays experienced by the packets of the short flow.

Figure 4 shows the percentage of out-of-order packets for the short flow over a different number of long flows when the source-destination pairs are static. The results in this figure exhibit almost the same percentage of out-of-order packets for two or more long flows. Despite generating out-of-order packets, per-packet load balancing is advantageous in terms of FCT and throughput.

We also investigate the average throughput, average FCT, and the percentage of out-of-order packets when 20 short flows are generated between random source-destination pairs for a different number of long flows. Figure 5 shows how the average throughput for the long flows changes when the number of long flows increases. Per-packet load balancing in Figure 5 exhibits a larger throughput as compared to that of the single-path forwarding approach. The reason of the larger throughput gap between the two different approaches in this

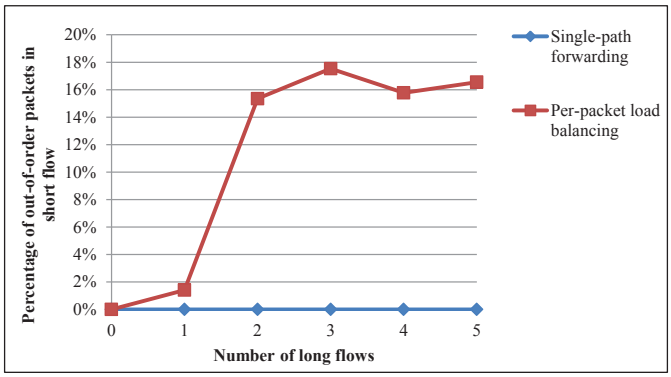

Fig. 4. Percentage of out-of-order packets in short flow for static source-destination pairs.

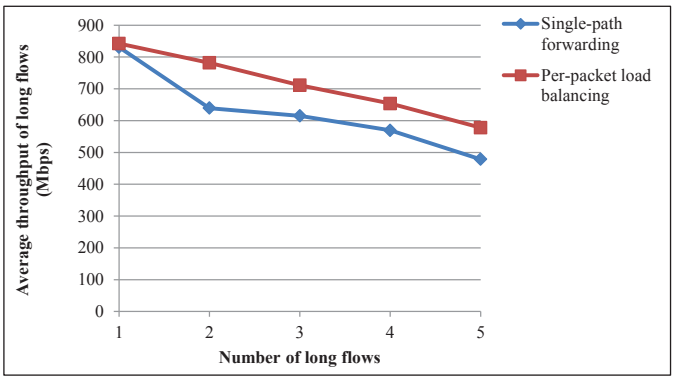

Fig. 5. Average throughput of the long flows for randomized source and destination nodes.

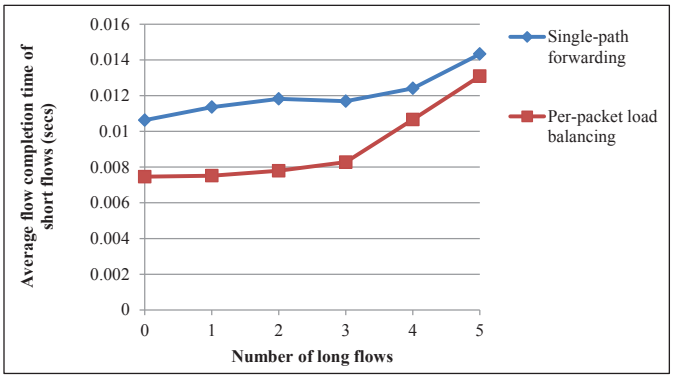

Fig. 6. Average flow completion time of the short flows for randomized source-destination pairs.

figure is that there is no bottleneck link that may limit the benefits of per-packet load balancing as in a TCP incast-like scenario.

Figure 6 shows the average FCT of the short flows under randomized background traffic for a different number of long flows. This figure shows a smaller FCT for per-packet load balancing, indicating that using the multipath feature of a DCN may benefit the FCT of short flows.

Figure 7 shows the percentage of out-of-order packets of short flows under randomized background traffic. This figure shows that although about 20% of packets are delivered in out-of-order, per-packet load balancing may still benefit both short

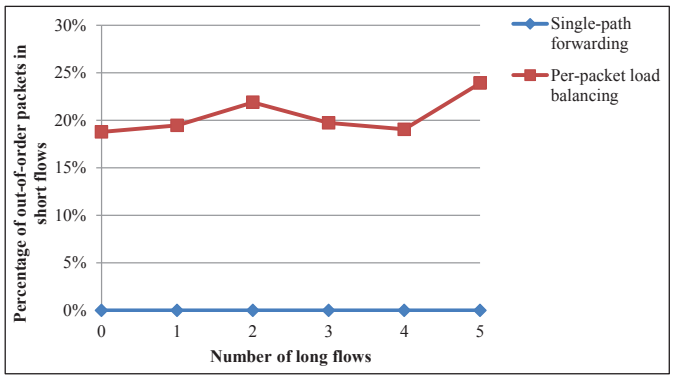

Fig. 7. Percentage of out-of-order packets in short flows for randomized source-destination pairs.

and long flows in terms of FCT and throughput, respectively.

III. Related Work

Equal-Cost Multiple-Path (ECMP) forwarding is a flow-based load balancing mechanism which is used to balance the load among equal-cost shortest paths to the destination. ECMP hashes the five-tuple (i.e., source and destination IP addresses and the port numbers, and the protocol number) of a packet to decide which output port at the switch should be selected to forward the packet. If the received packet is the first packet of a flow, a hash function is executed to determine the output port and the remaining packets of that flow are forwarded to the same output port. Because ECMP is a flow-based load balancing mechanism and it does not differentiate short from long flows, the same output port is selected to forward all these long flows. This means that large queuing delays or congestion may be experienced by the delay-sensitive short flows if they are coincidentally forwarded to the same output port the long flows use [4].

To improve the performance of DCN, some multipath forwarding schemes, such as DeTail [16], Hedera [17], Congestion-Aware Load Balancing (CONGA) [18], Multipath TCP (MPTCP) [19] and, Random Packet and Spraying (RPS) [10] have been recently proposed.

DeTail is a cross-layer scheme aimed at reducing long tail FCT of short flows in DCNs. DeTail proposes to use per-packet adaptive load balancing at network layer and a reorder-resistant transport to prevent TCP from reacting out-of-order packets as an indication of congestion. However, DeTail requires complex modifications at more than one layer at the network stack, hence it diverges from the objective to keep the implementation complexity low.

Hedera is a central flow scheduling scheme which aims to relocate long flows if they occupy at least 10% of the link capacity. Because Hedera uses a central scheduler, it is necessary to frequently run this central algorithm to relocate the long flows, which may make Hedera slow to react to dynamically changing traffic in a DCN compared to distributed multipath forwarding schemes.

CONGA is a distributed, congestion-aware, in-network load-balancing scheme. CONGA is specifically designed for a two-layer DCN architecture, which is called leaf-spine, to

balance the load among equal-cost shortest paths by splitting the flows into trains of packets, called flowlets. CONGA collects the congestion information using piggybacked feedback messages from a destination leaf (i.e., edge) switch, stores them, and selects the best output port in terms of congestion extent at the source leaf switch to forward the flowlets. Despite CONGA's distributed nature, it requires modifications at the switches and the necessary space requirements to store the congestion information at the switches makes it costly to implement.

MPTCP is a multipath forwarding scheme which splits a TCP flow into several sub-flows at the sender and transmits these sub-flows on different paths using ECMP. MPTCP shifts the complexity from switches to end hosts where each end host requires a more complex transport protocol than regular TCP.

RPS is a packet-based forwarding technique that forwards packets of flows through different equal-cost shortest paths to their destinations, where a path is randomly selected with uniform probability. We take RPS as our model since it is simple and it does not require any modification at switches or servers.

IV. Conclusions

In this paper, we analyzed the random packet spraying load balancing scheme and evaluated the performance of this scheme on short and long flows. We considered the FCT of short flows and the throughput of long flows as performance metrics. We conducted our tests in a simulated data center network using a fat-tree architecture with four pods and 16 servers. We also presented the percentage of out-of-order packets at the receiver node and showed that the out-of-order packet delivery directly impacts the performance metrics of all flows when TCP is employed as the transport protocol. Our results showed that implemented random packet spraying scheme achieves smaller average FCT for short flows and higher average throughput for long flows over the single path used by TCP. Our simulation results also reveal that the percentage of out-of-order packets of short flows remains almost constant as the number of long flows increases. However, we observed that out-of-order packet delivery does not have a significant impact on the average FCT of short flows and the average throughput of long flows under the tested conditions.

References

[1] N. Farrington and A. Andreyev, "Facebook's data center network architecture," in *Optical Interconnects Conference, 2013 IEEE*, May 2013, pp. 49–50.

[2] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*. IEEE, 2010, pp. 1–10.

[3] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," in *ACM SIGOPS operating systems review*, vol. 37, no. 5. ACM, 2003, pp. 29–43.

[4] R. Rojas-Cessa, Y. Kaymak, and Z. Dong, "Schemes for fast transmission of flows in data center networks."

[5] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible data center network," in *ACM SIGCOMM computer communication review*, vol. 39, no. 4. ACM, 2009, pp. 51–62.

[6] J. Rajahalme, S. Amante, S. Jiang, and B. Carpenter, "Ipv6 flow label specification," 2011.

[7] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center Tcp (DCTCP)," *ACM SIGCOMM computer communication review*, vol. 41, no. 4, pp. 63–74, 2011.

[8] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 59–62, 2006.

[9] C. Ding and R. Rojas-Cessa, "Daq: Deadline-aware queue scheme for scheduling service flows in data centers," in *Communications (ICC), 2014 IEEE International Conference on*, June 2014, pp. 2989–2994.

[10] A. Dixit, P. Prakash, Y. Hu, and R. Kompella, "On the impact of packet spraying in data center networks," in *INFOCOM, 2013 Proceedings IEEE*, April 2013, pp. 2130–2138.

[11] Network Simulator-3 (NS-3). [Online]. Available: https://www.nsnam.org/

[12] V. G. Cerf and R. E. Icahn, "A protocol for packet network intercommunication," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 2, pp. 71–82, 2005.

[13] P. Prakash, "Impact of network protocols on data center applications," Ph.D. dissertation, 2013.

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

[15] H. Wu, Z. Feng, C. Guo, and Y. Zhang, "ICTCP: Incast Congestion Control for TCP in data center networks," *IEEE/ACM Transactions on Networking*, vol. 21, no. 2, p. 345, 2013.

[16] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "DeTail: Reducing the flow completion time tail in datacenter networks," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 139–150, 2012.

[17] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *NSDI*, vol. 10, 2010.

[18] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. The Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proc. of SIGCOMM*. ACM, 2014.

[19] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 266–277.