Race Cars vs. Trailer Trucks: Switch Buffers Sizing vs. Latency Trade-offs in Data Center Networks

Alexander Shpiner, Eitan Zahavi
Mellanox Technologies
{alexshp,eitan}@mellanox.com

***Abstract*— This paper raises the data center designers question of trade-off between high-buffer switches versus low-latency switches. Packet buffer hardware dictates this trade-off due to the constraints of DRAM and SRAM technologies. While the designers who prefer network robust solutions would typically prefer large-buffer switches with settling for high latency; the designers who can adapt applications to the network behavior would prefer the low-latency switches in order to gain better application performance.**

In this paper, we review the question of switch buffer sizing in data center networks, by considering the switch delay in light of common traffic patterns in data centers. To the best of our knowledge, this is the first paper that discusses the switch buffer sizing question by considering switch latency trade-off. We review previous works on switch buffer sizing given the typical parameters of data center networks, and survey the typical data center traffic patterns that challenge the switch buffer. Also, we provide simulation results that show the effect of switch latency on the effective bandwidth of acknowledgement-based congestion controlled flows. Finally, we discuss the gain that flow control provides to end-to-end network performance.

Keywords—switch; packet buffer sizing; data center network

I. INTRODUCTION

Data center network designers face the issue of handling congestion in the network. Every network is prone to congestion, and the most common solutions to address it can be grouped into two categories: solutions that allow applications to adapt to the network behavior and solutions that try to make the network as robust as possible.

The first solution category typically leads to the choice of low-latency[1] switches, typically with modest packet buffering. The second category favors switches with large buffering (ideally infinite), and are willing to settle for much higher network latency.

Large buffer switches can accumulate traffic when congestion occurs, but are subject to queuing delays and high switching latency. Small-buffer switches provide low switching latencies, but are potentially more vulnerable to incast scenarios, in which the arriving traffic to the switch is much larger than the switch can forward, due to the limited link capacities.

In this paper, we analyze buffers and latency trade-offs and suggest buffer size optimizations for typical data center traffic patterns. We show that network performance is not linear with buffer sizes, i.e. from a certain size, additional buffering only adds latency without significantly improving handling of incast scenarios. Moreover, latency sensitive applications, such as distributed computing, database query, web search, high performance computing, etc., demand a non-blocking and low-latency network. When networks are not overloaded, the higher switch latency has a negative effect on TCP flow throughput.

We review the previous work on buffer sizing, focusing on the data center network's typical characteristics. Next, we analyze the buffer sizing and latency for popular data center traffic patterns: web search, distributed storage, Map Reduce, and high performance computing (HPC). Next, we show that in typical high-performance networks, switch latency can actually have significant impact on the effective achieved throughput, when common window-based congestion control is used. Last, we show that network congestion is better managed by flow and congestion control, rather than by huge buffers.

Throughout the paper, we used an OMNet++-based Inet-enhanced network simulator[1][2] to evaluate our statements.

II. SWITCH BUFFER SIZING AND LATENCY: TRADE-OFF OF RACE CARS VS. TRAILER TRUCKS

Large-buffer switches are analogous to trailer trucks, as the packet buffers are large but slow. Their large packet buffers are normally implemented in slow DRAM memory. That leads to higher, varying switching latency, usually over-microseconds (e.g. Arista DCS-7280E latency is 3.8us with 9GB buffer [3], Arista DCS-7500E latency can go to 12us with 72GB-144GB buffer [4], Juniper QFX10000 latency is 5.5us with 8GB and 12GB buffers [5], Cisco Nexus 7000 latency is 9.5us with 5.6GB buffers [6]).

Small-buffer switches are analogous to race cars, as the packet buffers are implemented using fast SRAM memory, but with lower buffering size. Such switches usually have sub-microsecond latencies (for example, 220 ns in Mellanox SX1036 [3][7]), 550ns in Arista 7050QX [8]). Several enhancements, such as cut-through and advanced store and forward, are used to decrease the switch latency even further. Therefore, the trade-off of the network designer is essentially

[1] Unless stated differently, by switch latency we refer to the empty switch port-to-port packet latency, and not to the queueing latency.

between low latency and big buffers. We will evaluate this trade-off in the following sections.

III. Optimal Buffer Sizing

The question of optimal buffer sizing has been studied for more than two decades [9][10]. Insufficient buffers can cause a high packet drop rate and throughput collapse due to incast, while large buffers cause unnecessary latency and slow reaction to congestion.

The most popular work on buffer sizing is reference [11]. The suggested buffer sizing rule is to set the buffers to be equal to the product of the bandwidth times the flow round trip time divided by the square root of the number of flows ($BW * t_p / \sqrt{\#flows}$). This rule follows from the conventional TCP congestion control behavior with fast recovery function, according to which the congestion window is decreased by two upon a packet loss event. The buffer size is defined as a value that keeps the full bandwidth for a flow with minimal queueing latency. When the number of flows is large enough to avoid their inter-synchronization, the buffer is required to absorb only the accumulated rate fluctuation, which is reduced when the number of flows is large. Using the typical data center network parameters (link bandwidth of 100 Gigabits per second, round-trip propagation time of 200 microseconds and roughly 200 flows per link: $BW = 100Gbps, t_p = 200us, \#flows = 200$), it suggests very low buffers (~178KB) per switch. However, the drawback of that model is that it does not consider the fact that the fast recovery function of TCP does not enter into operation if the flow congestion window is under 4 segments (due to the 3 duplicate ACKs required to trigger the fast recovery mode). Therefore, it does not suit common incast scenarios in which the drops happen before the flows reach a congestion window of 4, as described in the literature [12][13][14][15].

Other works [16][17] extend the model to other cases, adding that when the number of flows is large, the buffering should be proportional to the number of flows. The suggested egress buffer size is equal to 9 packets multiplied by the number of flows ($9[packets] * \#flows$). With a rule-of-thumb of 200 concurrent flows in the switch, this would need a 2.7MB buffer per switch for 1.5KB packets. In other words, they suggest a buffer size that will avoid the TCP incast throughput collapse, without naming that specifically (References [16][17] were published before the TCP incast works).

The most recent IETF draft [18] strongly recommends using ECN and AQM schemes over implementation of large buffers. The Bufferbloat project [19][20] aims to solve the "laggy network performance" by actively limiting the queuing delay in the buffers. Subsequent works [21][22][23] recommend even smaller buffer switches than the original work.

The direct drawback of the unnecessarily large buffer on network performance is the extended end-to-end network queueing latency the packets may suffer. The impact of excessive queues reduces the performance of low-latency-sensitive transactions.

We evaluated the above statement of the buffer size impact on the switch queueing latency by running the following set of simulations. We simulated many-to-one scenario of 20 persistent (long) TCP incast flows sent from 20 hosts to a common destination over a single switch with a bottlenecked 100Gbps link. We varied switch buffer sizes and checked the end-to-end packet delay for each buffer size. The end-to-end packet delay was measured by periodically sending a UDP packet from each of the TCP flow injectors to a common destination and measuring its end-to-end latency. The UDP packets' injection rate is low enough so as not to produce a meaningful effect on the TCP traffic. Under the given network parameters, the system converges to the point where the TCP flows operate in the congestion-avoidance state without retransmission timeouts. We also calculated the theoretical bound of the queuing delay of the full buffer. The theoretical bound is calculated by dividing the switch buffer size by the link rate and adding the switch processing delay and links propagation time. We draw both the simulated end-to-end latency result and the theoretical bound on Fig. 1. The result shows that mean end-to-end latency is increased linearly with the increase in buffer size. In addition, since the end-to-end latency was relatively close to the theoretical bound hints that the buffers are kept almost full, even with increasing the buffer capacity.

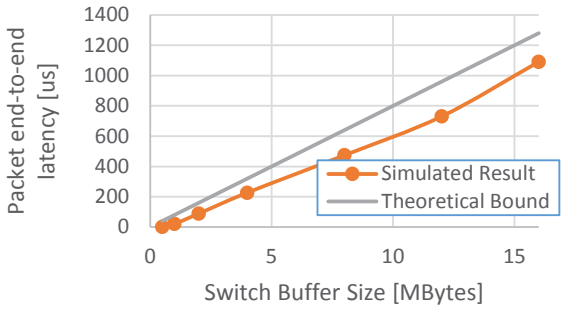


Fig. 1. End-to-end packet delay as a function of switch buffer size

Conclusion: TCP-based traffic keeps the buffers almost full for any buffer size and the mean queueing delay increases almost linearly with the buffer size.

In addition, as we will show in Section V, the end-to-end latency is also crucial for achieving high throughput for long flows driven by acknowledgement-based congestion control algorithms, such as TCP.

IV. When Do We Need Buffers in Data Centers?

In this section we analyze the large-buffer vs. low-latency trade-off by discussing the buffering requirements in the typical data center scenarios. A popular method for estimating the buffering requirements of data center switches is the incast scenario [12]. In this scenario, multiple TCP flows are transmitted over a single switch to a common destination through the same output port over a bottleneck link. The incast scenario is normally related to data center applications such as web search, distributed storage, Map Reduce, and HPC. We will take a deeper look into each of them in the following paragraphs.

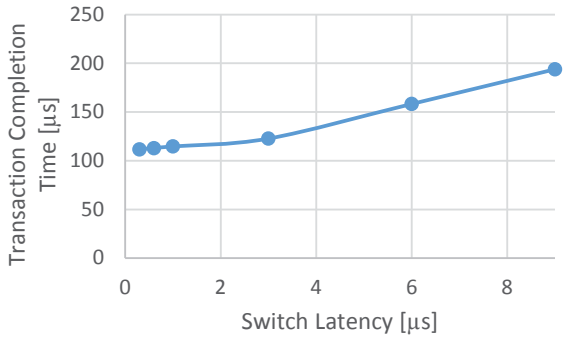


Fig. 2. Mean completion time of web search partition/aggregate transactions as a function of switch latency. Switches with smaller latency also have smaller buffer sizes, such that 300ns-latency switch is capable to 0.5MBytes of buffering, and 9µs-latency switch is capable to 16MBytes of buffering.

A. Web Search

Web search is a typical example of a partition/aggregate traffic pattern, explained in [24]. Applications set very tight deadline times for web queries, such that it is very hard for conventional TCP to meet them. Therefore, software designers use proprietary UDP-based transport layer protocols [25], and hence the conventional TCP throughput collapse incast problem is not applicable in these cases. Moreover, as stated in [24], data center network delays play a significant role in application design, causing the network designers to avoid network under-provisioning and to use the fastest possible low-latency network devices.

Due to the reasons stated above, the large buffering is unnecessary in web-search application networks. Hence, we analyze the application performance by switch latency parameter. We simulated partition/aggregate transactions over 64 hosts connected in a two-level fat tree network. The transaction is combined from 64B request messages into a random subset of the hosts and 20KB reply messages. The transaction is defined as completed after all the reply messages have been received. Transaction request arrivals were modeled as exponentially distributed with a mean of 1ms per host. We

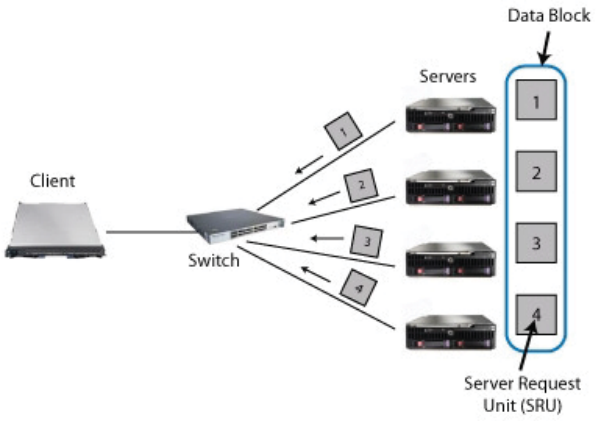


Fig. 3. Incast scenario [26]

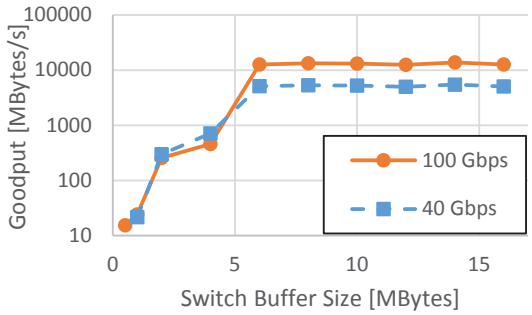


Fig. 4. Incast scenario goodput as a function of switch buffer size

varied the switch latencies from 300ns up to 9µs and checked the mean transaction completion time. Fig. 2 shows the mean completion time of transactions as a function of the switch latency. The results show that using fast switches of 300 ns latency reduces the transaction time by almost twice compared to 9us switches. The longer transaction times are caused both directly by the longer switch latency, and indirectly by additional queueing delay due to the larger amount of data in the network at each point of time.

Conclusion: Since web search applications are latency-sensitive and hence deployed on over-provisioned networks, the switch delay has more significant effect on the application performance than switch buffer size parameter.

B. Distributed Storage

Distributed storage applications are the first that were found to suffer from the TCP incast throughput collapse problem[13][26]. The problem is described as throughput collapse of a distributed storage request, where a bottleneck link is not utilized while the flows are in time-out mode, as depicted in Fig. 3. Synchronized flows create excessive packet drop rate, which enters the flows into a time-out state. During the time-out, the link bandwidth is not utilized. Since the distributed storage read transaction completes when all the flows involved are finished, the effective throughput of the transaction is drastically reduced.

There are various methods to mitigate and overcome the incast problem. Specifically, TCP incast throughput collapse can be solved in several different ways on the upper layers (application, transport, link, etc.) [14]. The straightforward solution in the switch architecture level is to increase the switch buffers. The incast papers refer to switches with buffer sizes of 32KB to 1MB to illustrate the problem [12] [15].

However, current switches have typically buffer size that is larger than the 1MB mentioned above. Hence, we aimed to evaluate the TCP throughput collapse problem with larger switch buffers of several Mbytes (Those buffer sizes are still small enough to be implemented using fast SRAM memories). In order to quantify the effect of buffer size in a distributed storage environment, we simulated an incast scenario with block size of 80MB, exactly as referred to as next generation data centers in [27], and 10 hosts with 20 sender servers on each, over 40Gbps and 100Gbps links to a common client server (Fig. 3). We varied the shared buffer size of the switch from 0.5MB to

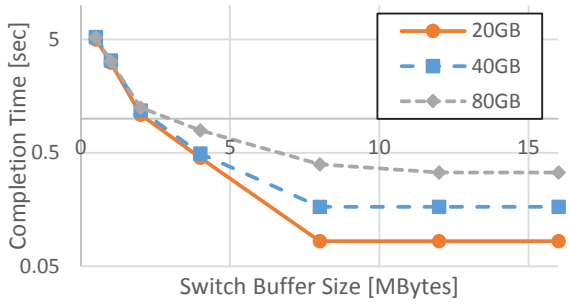


Fig. 5. Completion time of MapReduce shuffle phase as a function of buffer size

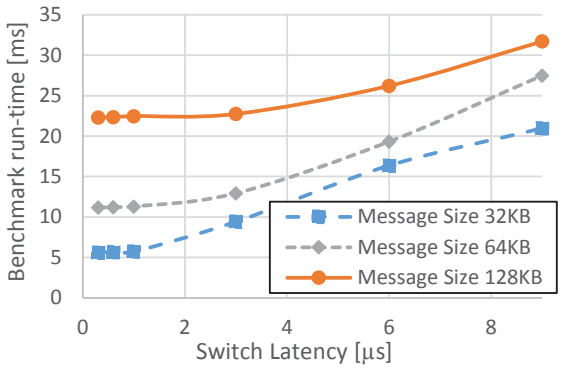


Fig. 6. Completion time of b_eff test under various switch parameters. Switch buffers are modeled with latencies of 0.3µs, 0.6µs, 1µs, 3µs, 6µs and 9µs and with buffer sizes of 16MB, 40MB, 80MB, 276MB, 666MB and 1GB respectively

16MB and checked the effective goodput under every buffer size value. We avoided to vary switch latency as well, since in current scenario the queueing latency dictates the latency of the packets inside the switch. As shown in Fig. 4, increasing the switching buffer in a shared memory model is efficient up to 6MB of buffer size. We show that shared buffer size of 6MB is enough for the typical incast scenarios that are presented in the known literature. Real environments combine background traffic, which was not considered in that analysis, hence real buffering requirements would be a bit larger.

Conclusion: Current SRAM based switch buffers are large enough to handle typical TCP incast scenarios.

C. *MapReduce*

MapReduce [28] is a popular operation in data centers that is used for Big Data computation. The common belief is that MapReduce requires large network bandwidth and large buffers. The demand for bandwidth is reasonable due to the large amount of data that must be transferred quickly in the Shuffle stage (when each mapper sends its results to each of the reducer processes). However, the desire for large buffers is questionable. Since the MapReduce communication stage shuffles traffic from many nodes to many nodes, its traffic pattern is similar to all-to-all with many short data transmission flows, usually of 100B to 10KB [29] [30]. In other words, each physical host participates in several parallel many-to-one traffic transactions. Since the traffic pattern is well spread [31], the network does not suffer from heavy congestion events, and therefore does not require huge buffers. Reference [32] analyzed popular data center network characteristics and could not find any direct evidence of incast problem. Moreover, the benefits of addressing incast are completely masked by overhead from other parts of the system. As reference [33] states: "Small jobs dominate several production Hadoop workloads. Non-network overhead in present Hadoop versions mask incast behavior for these jobs".

In the following simulations we evaluated the completion time of typical MapReduce pattern of Terasort. We simulated multiple incast flows with MapReduce flow parameters based on a 20GB, 40GB, and 80GB Terasort example [33]. 20 hosts of mappers and 20 hosts of reducers connected to a single switch in a star topology. Each host of reducers ran 10 reduce processes. 20, 40, and 80 GB of data was distributed on 20 mappers equally and was shuffled to the 200 reducer processes uniformly, such that each one of the 200 reduce processes received 5MB, 10MB, and 20MB of data respectively from each mapper (5MB * 200 reducers * 20 mappers = 20 GB). We varied the switch buffer size and collected the shuffle phase completion time. As in the previous simulations set, we avoided to vary switch latency as well, from the same reason. Fig. 5 shows the completion time of the communication phase as a function of switch shared buffer size. The results show that there is no observable benefit of increasing the buffers beyond 8MB in this case.

Conclusion: Current SRAM based switch buffers are large enough to handle typical MapReduce scenarios.

D. *HPC Benchmark*

High-performance computing (HPC) applications demand stiff high throughput and low-latency requirements from the network. HPC Challenge (HPCC) [34] is a popular benchmark suite to measure the performance of HPC applications. One of the benchmarks it includes is the 'effective bandwidth' (b_eff) test [35] that challenges the network performance. We simulated a b_eff test combined from 1500 nodes in a 3-level fat tree topology over a 56Gbps InfiniBand network, with switches of 16MB, 40MB, 80MB, 276MB, 666MB, and 1GB buffer sizes and 0.3us, 0.6us, 1us, 3us, 6us, and 9us latency respectively. Rank placement was random and the network topology was non-blocking. The simulation was run for three values of message size: 32KB, 64KB, and 128KB. The simulation results are shown in Fig. 6. It can be seen that using fast switches of hundreds of nanoseconds latency improves the benchmark run-times by 4 times in the 32KB messages case and by 45% in the 128KB messages case compared to the switches with 9us latency. This is despite the larger buffer sizes in the slower switches, which are supposed to better absorb the temporal incast load.

Conclusion: HPC application performance gains higher performance benefit from small-latency switches than from high-buffer switches.

V. Latency Impact on the Effective Throughput

In order to make latency sensitive transactions complete quickly, a low-latency end-to-end network is desired. One must also note that specifically the switch latency is an important

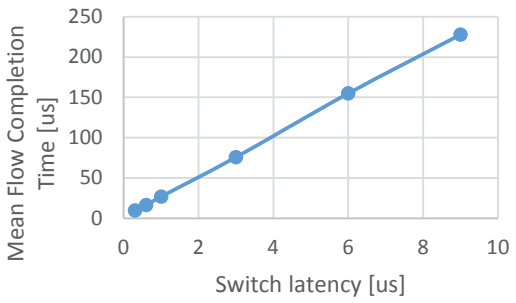
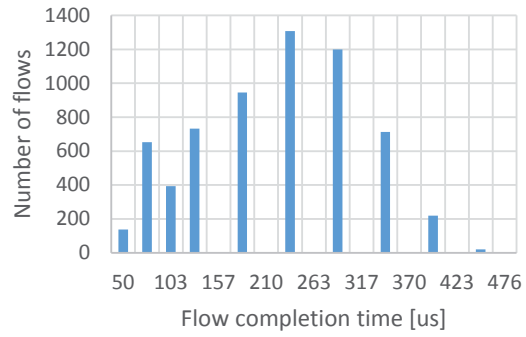
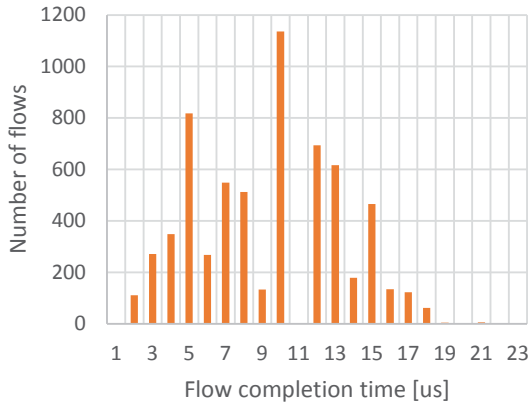


Fig. 7. Mean flow completion time vs switch latency under low load. Switches with smaller latency also have smaller buffer sizes, such that 300ns-latency switch is capable to 0.5MBytes of buffering, and 9µs-latency switch is capable to 16MBytes of buffering.



(a) 9us-latency switch: flow completion times are up to 450us



(b) 300 ns-latency switch: flow completion times are up to 22us

Fig. 8. Histogram of flow completion time vs switch latency under low load

factor on short congestion-controlled (for example TCP) transactions. For short TCP transactions, the flow completion time is inverse to RTT (round trip time), due to the acknowledgement-based congestion control, since the flow transmission rate is roughly equal to the congestion window divided by RTT. The transaction begins from connection establishment, which takes one RTT, then it sends one packet[2], waits an RTT for ACK, sends two more packets, waits another RTT for ACKs, and so on. Hence, for short transactions the link bandwidth is not fully utilized, and the transaction completion time is mostly defined by the end-to-end network latency.

We evaluated the switch latency impact on the effective bandwidth for short transactions using the following simulations. We simulated a two-level fat-tree network of 64 hosts with 100 Gbps bandwidth and 10 ns latency links. We injected random TCP flows of an average of 10KB length under low load and compared their completion times using several types of switches with latencies and buffer varying from 300 ns with 0.5MB buffer up to 9 us with 16MB buffer (since the injected traffic load was low, increasing the buffer size further was unnecessary).

Fig. 7 shows the mean completion time of the flows as a function of the switch latencies. It can be estimated that the effective bandwidth of the short flows linearly depends on the switch latency parameter.

In some cases, an application operation depends on completion of several transactions, for example file retrieval in distributed storage. Then the application performance depends not only on the mean metrics of completion time, but also on the completion time of 99% and more percentile of flows. Fig. 8 shows histograms of the completion time of flows in a network with switches of 300 ns latency and network with switches of 9us latency. It can be seen that low-latency switches reduce the 99%-th percentile flow completion time by 95.1%, from ~450us to 22us. The result implies that with 300ns latency switches in the network, the effective achieved throughput is equal to 3.6Gbps and with 9us latency switches it is only 170Mbps. Thus, latency is an important factor that also affects the throughput of the network.

Conclusion: Network latency in general, and switch latency specifically, dictate the effective throughput of short acknowledge-based congestion-controlled transactions.

VI. Mitigating Incast Throughput Collapse By Flow Control

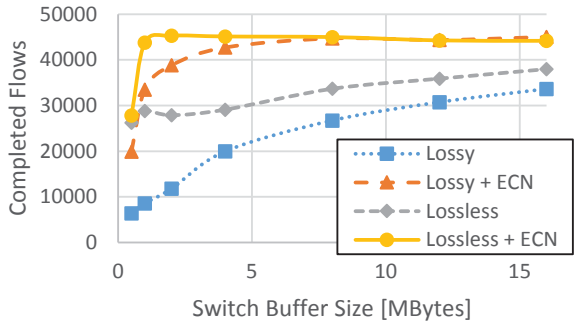
TCP incast throughput collapse can be solved in various ways other than using extremely large buffers. These solutions are based on adapting the application or transport layers [13][14].

From the network designer point of view, the recommended solution is to deploy small-buffer switches in the network with a congestion control algorithm that is based on congestion notification in the switches (for example, ECN) or round-trip delay measuring (for example, TCP Vegas) as notification of congestion prior to when the packets begin to be dropped. Using

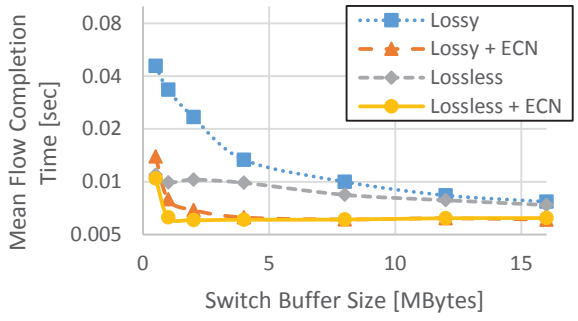
[2] We assume that the initial congestion window is equal to one MSS. Several operating systems configure slightly larger value.

small-buffer switches, the applications gain from the advantages shown for low latency, while the congestion control algorithm can mitigate the queuing latencies. The latest trend in data center networking is to use the Converged Enhanced Ethernet (CEE), whose key feature is the losslessness of the network [36]. Lossless networks do not drop packets, but use Priority Flow Control (PFC) [37] to pause the incoming packet transmission before the ingress buffers fill up. Lossless networks were evaluated in several prior works [38][39][40], and were shown to perform better than lossy networks in typical data center scenarios. Moreover, lossless networks possess the property that increased link capacity allows switch buffers to be reduced even more to achieve the same application performance [41].

We present simulation results that compare lossy and lossless networks performance under various switch buffer size. We simulated a two-level fat tree network of 40Gbps links with 64 hosts and varied the switch buffer sizes. We injected traffic of random-destination and random-length TCP flows and compared the performance under lossy and lossless networks with and without ECN. Fig. 9 shows the comparison between the lossy and lossless networks under various switch buffer sizes. Figure 9(a) shows the number of completed transactions during the simulated time of five seconds, and Figure 9(b) shows the average completion time of the transaction. The results show that lossless scenarios outperform lossy scenarios, while using ECN provides additional performance gain. The reason for this behavior is that pausing incoming traffic is less expensive than dropping traffic when the switch buffer overflows; while adding ECN provides congestion control for the lossless networks as well.



(a) Number of completed flows



(b) Average completion time of the flows

Fig. 9. Flow completion time vs buffer size in lossy and lossless network

Conclusion: Lossless network with ECN-based congestion control provides better network application performance than conventional lossy networks.

VII. Summary

This paper strives to determine the guidelines for switch buffer sizing for popular traffic patterns in data center networks. In addition, the switch latency parameter was introduced into consideration and was shown to have a non-negligible effect on network application performance.

As this paper's analysis shows, the preferred methods of achieving high application performance involve low-latency networks. MapReduce and other TCP applications perform better when the network round trip is short. Non-TCP applications, such as web search, prefer non-blocking networks and also benefit from low latency.

The network that delivers the lowest latency is built of small-buffering switches that implement flow control and congestion control mechanisms. While buffering is always required for transient events, excess buffering only has a negative impact on the data center. Our evaluation showed that there is no evidence for gain of switch buffer sizes larger than several MBytes. The right amount of buffering is correlated with the switch and its port speeds, such that bigger is not necessarily better.

References

[1] A. Varga, "Omnet++", 2004. http://www.omnetpp.org/

[2] "INET framework", 2006. http://inet.omnetpp.org/

[3] Arista 7280E Series Specification, http://www.arista.com/en/products/7280e-series

[4] Arista 7500E Switch Performance Test, Lippis Report, 2014, https://www.arista.com/assets/data/pdf/7500E-Lippis-Report.pdf

[5] Juniper QFX10000 Modular Ethernet Switches, https://www.juniper.net/assets/us/en/local/pdf/datasheets/1000529-en.pdf

[6] Cisco Nexus 7000 F1 and M1 Modules, http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/VMDC/2-6/vmdcmodulesaag.pdf

[7] Mellanox SwitchX-2 (SX1036) vs. Broadcom StrataXGS Trident II (Arista DCS-7050QX) Performance Evaluation Qualifying Data Center Ethernet Networks with RFC2544 at 40Gbps, Tolly Group Report, 2015, http://www.mellanox.com/related-docs/products/Tolly-215111-Mellanox-SwitchX-2_Performance.pdf .

[8] Arista 7050X Series Specification, http://www.arista.com/en/products/7050x-series

[9] "Ancient History of Buffer Sizing", http://people.ucsc.edu/~warner/Bufs/buffer-requirements

[10] V. Arun, V. Sivaraman, and M. Thottan. "Perspectives on router buffer sizing: recent results and open problems", *ACM SIGCOMM Computer Communication Review* 39.2: 34-39, 2009.

[11] "Buffer Sizing in the Internet", http://yuba.stanford.edu/buffersizing/

[12] Y. Chen, et al., "Understanding TCP incast throughput collapse in datacenter networks." *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009.

[13] E. Krevat, V. Vasudevan, A. Phanishayee, D. G. Andersen, G. R. Ganger, G. A. Gibson and S. Seshan, "On application-level approaches to avoiding

TCP throughput collapse in cluster-based storage systems," *Supercomputing*, 2007.

[14] Y. Ren, et al., "A survey on TCP Incast in data center networks." *International Journal of Communication Systems* 27.8: 1160-1172, 2014.

[15] V. Vasudevan, et al., "A (In) Cast of thousands: scaling datacenter TCP to kiloservers and gigabits. *No. CMU-PDL-09-101. Carnegie-Mellon Univ. Pittsburgh, PA, Parallel Data Laboratory*, 2009.

[16] A. Dhamdhere, H. Jiang, and C. Dovrolis. "Buffer sizing for congested internet links." *INFOCOM,* 2005.

[17] R. Morris, "TCP behavior with many flows," *IEEE International Conference on Network Protocols*, pp. 205–211, 1997.

[18] G. Fairhurst, "Advice on network buffering", TSVWG Working Group, 2013, https://tools.ietf.org/id/draft-fairhurst-tsvwg-buffers-00.txt

[19] Bufferbloat Project, http://www.bufferbloat.net/

[20] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet", 2011.

[21] N. Beheshti, et al., "Obtaining high throughput in networks with tiny buffers", *IWQoS, 2008.*

[22] M. Enachescu, et al., "Routers with Very Small Buffers." *INFOCOM*, 2006.

[23] N. Beheshti, et al., "Experimental study of router buffer sizing", *ACM SIGCOMM conference on Internet measurement*, 2008.

[24] M. Alizadeh, et al. "Data center tcp (DCTCP)," *ACM SIGCOMM computer communication review* 41.4: 63-74, 2011.

[25] J. Rothschild, "High performance at massive scale: Lessons learned at facebook",

mms://video-jsoe.ucsd.edu/calit2/JeffRothschildFacebook.wmv.

[26] Incast, Parallel Data Lab Project, http://www.pdl.cmu.edu/Incast/

[27] V. Vasudevan, et al., "Safe and Effective Fine-grained TCP Retransmissions for Datacenter Communication," *SIGCOMM*, 2009.

[28] J. Dean and S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters", *In USENIX OSDI*, pages 137–150, 2004.

[29] T. Benson, A. Akella, and D. A. Maltz. "Network traffic characteristics of data centers in the wild." *ACM SIGCOMM conference on Internet measurement*, 2010.

[30] N. Dukkipati, et al., "Proportional rate reduction for TCP", *ACM SIGCOMM conference on Internet measurement conference*, 2011.

[31] M. Chowdhury, et al. "Managing data transfers in computer clusters with orchestra." *ACM SIGCOMM Computer Communication Review* 41.4: 98-109, 2011.

[32] S. Kandula, et al., "The nature of data center traffic: measurements & analysis." *ACM SIGCOMM conference on Internet measurement conference,* 2009.

[33] Y. Chen, et al., "Understanding tcp incast and its implications for big data workloads", *No. UCB/EECS-2012-40. California Univ. Berkeley, Dept. of Electrical Engineering and Computer Science,* 2012.

[34] HPC challenge benchmark suite, http://icl.cs.utk.edu/hpcc/

[35] R. Rabenseifner, Effective Bandwidth (b_eff) Benchmark, www.hlrs.de/mpi/b eff/

[36] M. Ko, D. Eisenhauer, and R. Recio, "A case for convergence enhanced ethernet: Requirements and applications," *IEEE ICC*, 2008.

[37] P802.1Qbb/D1.3 Virtual bridged local area networks - amendment: Priority-based flow control," IEEE Draft Standard, 2010. http://www.ieee802.org/1/pages/802.1bb.html

[38] A. S. Anghel, R. Birke, D. Crisan, and M. Gusat, "Cross-layer flow and congestion control for datacenter networks," *DC-CaVES*, 2011.

[39] D. Crisan, et al. "Short and fat: TCP performance in CEE datacenter networks", *IEEE High Performance Interconnects (HOTI),* 2011.

[40] M. Gusat, Cyriel Minkenberg, and Gergely János Paljak. "Flow and congestion control for datacenter networks", *RZ3742,* 2009.

[41] A. Shpiner, E. Zahavi, and O. Rottenstreich. "The Buffer Size vs Link Bandwidth Tradeoff in Lossless Networks", *IEEE High-Performance Interconnects (HOTI),* 2014.