Buffer Sizing Experiments at Facebook

Neda Beheshti
Facebook
nedab@fb.com

Petr Lapukhov
Facebook
petr@fb.com

Yashar Ganjali
University of Toronto
yganjali@cs.toronto.edu

ABSTRACT

In this paper, we present the results of several rounds of buffer sizing experiments at Facebook. We describe how we change buffer sizes in data center and backbone routers and study the impact of buffer sizes on several performance metrics, including flow completion time, latency, link utilization, and packet drop rate. Our observations suggest reducing buffers from millions of packets to a few thousand, and even a few hundred packets, does not lead to a general breakdown in network performance. In some cases we observed mostly tolerable degradation in some metrics (e.g., packet drop rates) and significant enhancements in others (e.g., latency).

1 INTRODUCTION

Router buffer sizes have a significant impact on network performance and system design and are a key factor in selecting ASIC/devices for production networks. Yet the question, "How big should a buffer be in a given network?" is still not well-understood.

For years, the common belief was that buffers as large as the network's bandwidth-delay-product are required for high performance [10]. Then, more than a decade ago, a number of studies questioned this belief and showed that high performance could be achieved with buffers that are orders of magnitude smaller than what was previously assumed, or even with only a handful of buffers, if certain conditions hold [4, 5, 7, 8]. These theories were validated in limited scenarios, in testbed and real networks [6].

To date, there is still not a considerable amount of data from production networks with real traffic to support any of these theories. This is mainly because buffer sizing experiments can be extremely challenging in production networks due to: i) practical difficulties associated with changing buffer sizes in commercial routers (e.g., when buffer is distributed over multiple linecards); ii) difficulty of characterizing and measuring network traffic conditions in order to model buffer size requirements; and iii) lack of control over network traffic and therefore, difficulty of enforcing any assumptions.

In this paper, we present a set of buffer sizing experiments conducted on Facebook's backbone and data center networks. We describe these experiments and our observations on how changing buffer sizes affect some of the network's performance metrics.

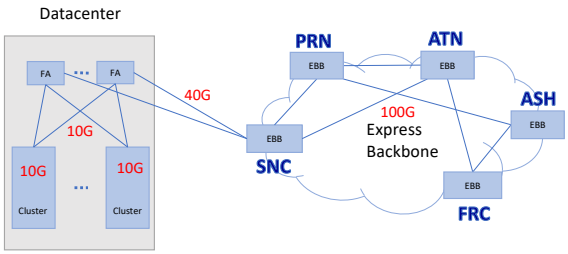

Figure 1: Data center and backbone topology.

2 EXPERIMENTS

Figure 1 shows a simplified view of Facebook's data center and backbone networks at the time of the experiments (2015-2016) [1]. Within a data center, Fabric Aggregation (FA) is the highest aggregation layer. FA routers handle both east-west traffic (which flows between buildings and clusters in the data center), and north-south traffic (which exits or enters the data center). Express Backbone [1] (EBB) is Facebook's backbone network that connects data centers in different regions and carries internal Facebook traffic, similar to other large-scale backbone designs [9, 11].

EBB had just become operational when we started our experiments and was mainly used for bulk data transfer and data replication. EBB was lightly loaded in general and had no bandwidth control or traffic engineering mechanisms implemented at the time of the experiments. For our experiments, we deliberately rerouted traffic towards a certain part of the network to ensure we have high utilization (even congestion) to get a better sense of the impact of buffer sizes.

Our experiments were performed independently on FA and EBB routers, as will be explained in the following sections. At the time of the experiments, most of the links from servers to top of rack switches were 10Gbps links. Data center to backbone links were running at 40Gbps, and all backbone links were running at 100Gbps.

At the time of our experiments, we used TCP Reno on our servers — we didn't have ECN enabled, didn't have pacing at the source, and didn't limit TCP's congestion window size on the servers. Traffic shaping was enabled on some servers using the standard Linux traffic control. For these

[1] More up-to-date information on Facebook's network topology and design can be found in [3]

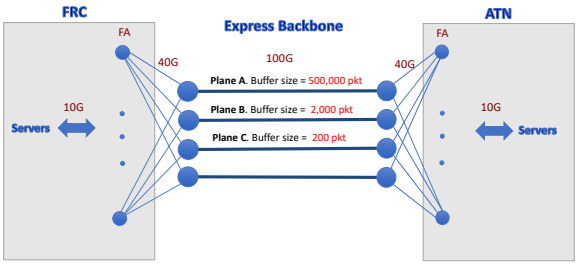

Figure 2: Parallel backbone links with different buffer sizes. Traffic is load-balanced (flow-based) across the links.

servers, the maximum allowed rate was determined by a central controller and updated every minute on the server.

Datasets collected during our experiments were mostly aggregated over a few minutes — one minute at best. We collected data either using SNMP or directly from the devices.

2.1 Backbone Experiments

Express Backbone is Facebook's cross-data-center backbone network that connects data centers in multiple regions and carries internal Facebook traffic. In 2016, EBB connected five regions with the topology shown in Figure 1. This topology was replicated on four independent and identical planes, which were running in parallel to each other, with backbone traffic being load-balanced across these planes.

EBB routers have large 3GB buffers, and buffering happens mostly on the ingress linecards of routers. There is one Virtual Output Queue (VOQ) for each pair of `<output port, traffic class>`. The 3GB buffer is shared among all input ports and all VOQs. To ensure that a single VOQ does not fill the whole buffer, a limit is set for the VOQ size. The default limit value is half a million packets, which was roughly the bandwidth-delay product in our network at the time of the experiments. We changed buffer sizes by changing this limit.

We did our experiments on the links between the FRC (Forest City, North Carolina) and ATN (Altoona, Iowa) data centers. These were some of the more congested backbone links at that time. We changed size of buffers on the backbone-facing interfaces and measured utilization and drop rate on those interfaces (Figure 2).

Since traffic was load balanced across backbone links, we were able to run our experiments simultaneously and use a different buffer size on each plane. We kept the default buffer size (half a million packets) on Plane A; reduced the buffer size to 2,000 packets on Plane B; and further reduced it to 200 packets on Plane C.

Congestion: During the few hours that we ran our experiments, we had intervals of very heavy congestion, with link utilization frequently reaching 100% (Figure 3). This congestion was deliberately generated by bringing some backbone

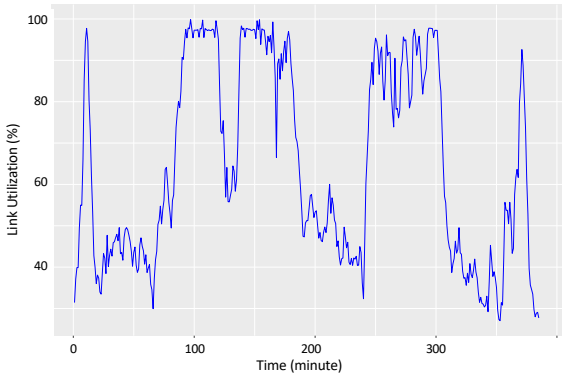

Figure 3: Backbone link utilization with default (half a million packet) buffer size.

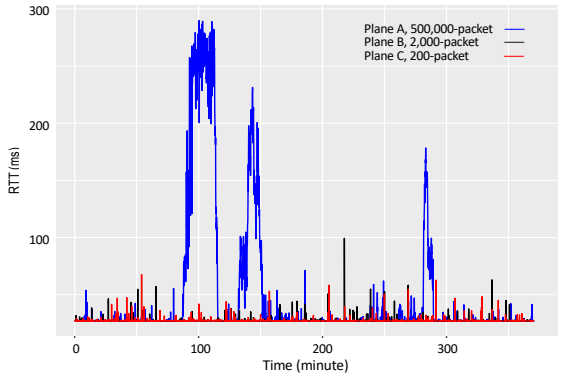

Figure 4: RTT with different buffer sizes.

interfaces down and forcing traffic being routed over the links under experiment.

As expected, this heavy congestion significantly increased latency on Plane A, which used the default large buffers (Figure 4). On this plane, the average RTT, measured using ping, was increased from the 30ms baseline to 300ms, with a small percentage of RTTs going above 700ms.

Packet Drop: Figure 5 shows how packet drop rate changes as traffic load increases on backbone links. Each point on this graph represents data collected over a few minute interval and shows the average packet drop rate and utilization during that interval. Data is collected directly from EBB routers and for each interface, it includes all traffic going over the interface.

On Plane A, we only see drops when link utilization is above 90%. On Plane B, drops start to show at 80% utilization. Finally, on Plane C, drops start at around 70% utilization. Based on this graph, at 90% utilization, packet drop rate is almost zero on Plane A, about 10^{-4} on Plane B, and about 10^{-3} on Plane C.

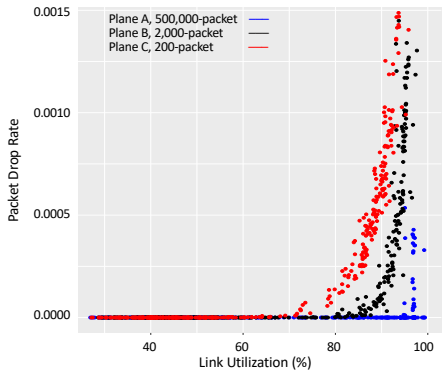

Figure 5: Packet drop rates with different buffer sizes. At 90% utilization, packet drop rate is almost zero on Plane A, about 10^{-4} on Plane B, and about 10^{-3} on Plane C.

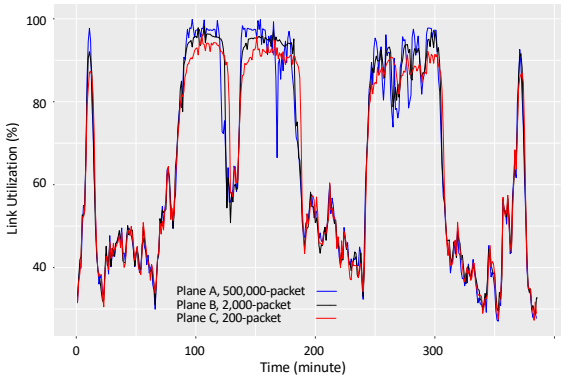

Figure 6: Backbone link utilization on links with different buffer sizes. We deliberately rerouted traffic towards target links to ensure we have high utilization, and congestion during experiments. Plane B shows lower utilization compared to Plane A during times of heavy congestion. Plane C shows about 10% reduction compared to Plane A.

Given that the measured utilization is averaged over a period of few minutes, the actual utilization during the time of packet drops might be much higher than the measured value. This explains why we see packet drops around 70% utilization. We expect the graphs to move to the right if instantaneous utilization data is used.

Link Utilization: Since TCP reduces the congestion window (and hence number of packets sent over each RTT) in response to packet drops, link utilization could be affected in presence of packet drops. Figure 6 shows link utilization on the three planes during the course of the experiment. Comparing planes A and B, we see slightly lower utilization on plane B during times of heavy congestion. The difference is about three to four percent when congestion is 100%. When load is below to 85-90%, however, utilization is the same on both planes.

On plane C, utilization is about 10% lower than plane A when congestion is 100%. Link utilization is the same on all three Planes when utilization is below 80%.

Flow Completion Time: To understand how individual flows were affected by different buffer sizes, we created synthetic traffic between a number of servers in FRC and ATN data centers. Since most services and applications at Facebook use Thrift [2] for RPC, we used a Thrift-based load-testing tool for traffic generation – in order to make the traffic more similar to that of real services. We configured the tool such that for each server-client pair, a Thrift query was made by the client. After the requested data was transferred, the associated TCP connection was closed, and the next query was sent over a new TCP connection. We set a small pause interval between consecutive queries. This was repeated for a few hours, and the generated traffic was distributed across all the four backbone links. We recorded the latency of all queries.

Figure 7 shows the CDF of flow completion times for 10MB flows. For any time value on the x-axis, the CDF shows the fraction of flows that were finished within that time.

This results show that Plane B, with 2000-packet buffers, consistently outperforms the other two planes. It also shows that, on average, Plane A and Plane C perform equally — meaning that they have almost the same average flow completion times (1142ms and 1135ms, respectively). An interesting observation here is that the default buffer size of Plane A results in more variations in flow completion times as the load varies. As shown in Figure 7, standard deviation of collected data is 1168ms on Plane A and 872ms on Plane C. This wider range of variations is caused by a small fraction of flows that take a long time to finish when congestion is heavy and buffers are very large.

Congestion Control: At the time of our experiments, there was no bandwidth control mechanism implemented in Express Backbone. Plain IGP was used for routing, without any auto-bandwidth or re-routing capabilities to control traffic volume on backbone links. This enabled us increase the load to 100% on the links under experiments and study extreme cases. But this does not happen when a proper traffic-engineering control mechanism is used in a network. Traffic throttling and bandwidth control keep link load below full utilization and reduce the required buffer size. In our experiments, there was no noticeable difference in packet drop rate or utilization of the three planes when load was below 70% – and as explained above, this number is likely an underestimate due to coarse granularity of measurements.

2.2 Data Center Experiments

All of our data center experiments were done in the FA layer, which is the highest aggregation layer within a data center

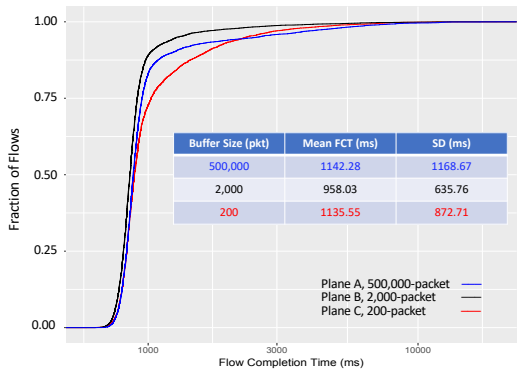

Figure 7: CDF of flow completion times. Plane B consistently outperforms the other two planes in terms of flow completion times.

(Figure 1). FA routers have small 12MB buffers, shared among all output ports and traffic classes. There are eight queues per egress port that serve different classes of traffic. These queues are served in a strict priority order.

Memory allocation to different queues in FA routers happens in a dynamic way, controlled by two parameters: the first parameter reserves a minimum amount of buffering for each queue (typically only a few packets) and the second parameter is a dynamic threshold that indicates relative priority of each queue compared to others. This threshold controls how the non-reserved portion of memory is assigned to incoming packets in real time. The default value of the reserved bytes is 1664 (almost one packet) in the FA routers.

In our experiments, we changed the buffer size by changing the dynamic threshold, making it half (small buffers) or double (large buffers) for a particular queue to decrease or increase the buffer size.

With the default dynamic threshold, we saw more drops in FA routers compared to the backbone experiments. This was expected because the shared buffer was very small. As we made the dynamic threshold smaller, we noticed more drops, similar to the backbone routers. However, we did not observe a significant impact on link utilization, probably due to the fact that traffic on these interfaces was mainly dominated by intra-data-center flows with extremely small RTTs.

To measure the impact of buffer size on flow completion times, we generated synthetic (iperf) traffic that was routed through three different FA routers: one with small buffers (as defined above), one with large buffers, and a control group consisting of routers with default settings. These flows were generated at a very slow rate (ranging from once in a few seconds to once in a few minutes) to ensure they will have minimum impact on network traffic, and ranged in size from 1MB to 1GB.

Figure 8 shows results for test traffic going through FA routers with small, large, and default buffer sizes. The graph on the left shows the case where source and destination of flows are in the same region, with an RTT of around 1ms. The graph on the right shows the case where source and destinations are in different regions, with an average RTT of 68ms. Interestingly, with short RTTs, there is very little difference in flow completion times of the three buffer sizes. However, as the RTT grows, flow completion times grow by 3-5% for 10-15% of flows.

This difference in small and large RTT cases can be justified by the fact that TCP can quickly recover from few packet drops when RTT is small. However, when RTT is large, it will take longer for TCP to learn about packet drops and accordingly adjust and recover from them; hence, packet drops have a more visible impact on flow completion times in this case.

3 CONCLUSION AND FUTURE WORK

Buffer sizing experiments are challenging, especially in an operational network. In this paper, we present a series of buffer sizing experiments we conducted in Facebook's backbone and data center networks using real network traffic, and occasionally, sample flows injected for measurement purposes. We studied the impact of reducing buffer sizes from millions of packets to a few hundred packets.

In our experiments, when buffer sizes are pushed to extremely small values, we observe slight reductions in link utilization and some increase in packet drop rate. But, when it comes to flow completion times, which is what many applications are concerned about, the difference is small, especially within data center networks where RTT is usually below 1ms. In terms of latency, small buffers can lead to significant improvements in certain scenarios.

Proposed Future Experiments. We consider this work a preliminary step in understanding buffer size requirements in our data center and backbone networks. The Express Backbone has grown in size, and its control mechanisms now (such as bandwidth control and traffic engineering) are much more extensive. Further, the amount of traffic and diversity in serviced applications have expanded as well. More comprehensive experiments are required to evaluate the impact of buffer sizes on this expanded EBB and more varied application set. Different production applications will have different performance metrics, and so we can explore conditions under which small (or large) buffers would be the preferred choice for a specific application.

Our data center experiments were limited to routers in the aggregation layer. Another over-subscription point to explore in data centers is top of rack switches, especially in

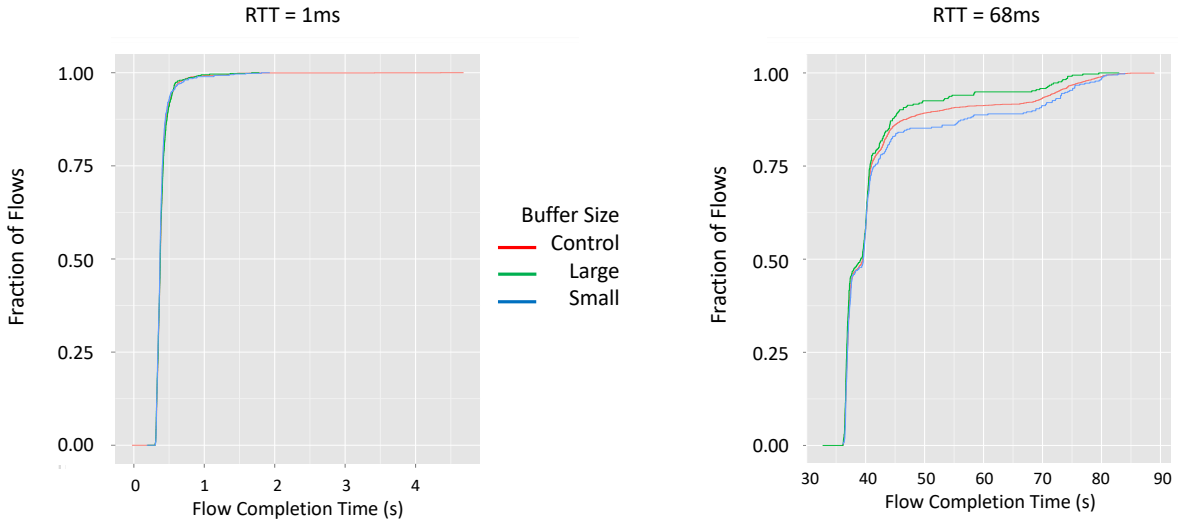

Figure 8: CDF of flow completion times in data center. There is very little difference in flow completion times of the three buffer sizes When flows have short RTTs (left), i.e. when source and destination nodes are in the same region. For larger RTTs (right), flow completion times grow by 3-5% for 10-15% of flows with small buffers.

the case of high-speed (e.g., 50G) senders, or when a shared network interface card is used by multiple servers.

An interesting phenomenon to study in this context is the TCP incast problem, which could become more apparent with the growth of server speeds. Here, few applications with large fan-outs may create very dense traffic bursts due to the precise synchronization of RPC-type requests, and might have a major impact on buffer size requirements.

Since traffic burstiness and packet drops can significantly be reduced through pacing (especially when there are large numbers of aggressive flows), another interesting direction for future work is to explore the impact of pacing on buffer sizing. TCP receiver window tuning is another technique that could manage burstiness and is worth a closer look.

A shortcoming of the results presented in this paper is the low granularity of measurements, which is in the order of minutes. More granular data collected through packet captures can provide more accurate results and better insight into how traffic characteristics and required buffer size are correlated.

4 ACKNOWLEDGEMENTS

We are indebted to Facebook's Backbone Engineering, Data Center Network Engineering, and Net Systems teams who supported these experiments on the Facebook production networks. We are especially grateful to Murat Mugan and Anshul Khandelwal for their help with running the experiments, and to Omar Baldonado for his continuous support of the work.

REFERENCES

[1] 2017. Building Express Backbone. Retrieved October 22, 2019 from https://engineering.fb.com/data-center-engineering/building-express-backbone-facebook-s-new-long-haul-network/

[2] 2019. Apache Thrift - Home. Retrieved October 22, 2019 from https://thrift.apache.org/

[3] 2019. Data Center Engineering. Retrieved October 22, 2019 from https://engineering.fb.com/category/data-center-engineering/

[4] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. 2004. Sizing Router Buffers. In *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '04)*. ACM, New York, NY, USA, 281–292. https://doi.org/10.1145/1015467.1015499

[5] Neda Beheshti, Emily Burmeister, Yashar Ganjali, John E. Bowers, Daniel J. Blumenthal, and Nick McKeown. 2010. Optical Packet Buffers for Backbone Internet Routers. *IEEE/ACM Transactions on Networking* 18, 5 (2010), 1599–1609. https://doi.org/10.1109/TNET.2010.2048924

[6] Neda Beheshti, Yashar Ganjali, Monia Ghobadi, Nick McKeown, and Geoff Salmon. 2008. Experimental Study of Router Buffer Sizing. In *Proceedings of Internet Measurement Conference (IMC)*. Vouliagmeni, Greece, 197–210 (Best paper).

[7] Neda Beheshti, Yashar Ganjali, Ramesh Rajaduray, Daniel Blumenthal, and Nick McKeown. 2006. Buffer Sizing in All-optical Packet Switches. In *Proceedings of Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC)*. Anaheim, CA.

[8] Mihaela Enachescu, Yashar Ganjali, Ashish Goel, Nick McKeown, and Tim Roughgarden. 2006. Routers with Very Small Buffers. In *Proceedings of the IEEE INFOCOM'06*. Barcelona, Spain.

[9] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Kondapa Naidu B., Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, Kirill Mendelev, Steve Padgett, Faro Rabe, Saikat Ray, Malveeka Tewari, Matt Tierney, Monika Zahn, Jonathan Zolla, Joon Ong, and Amin Vahdat. 2018. B4 and After: Managing Hierarchy, Partitioning, and Asymmetry for Availability and Scale in Google's Software-defined WAN. In *Proceedings of the 2018 Conference of the*

ACM Special Interest Group on Data Communication (SIGCOMM '18). ACM, New York, NY, USA, 74–87. https://doi.org/10.1145/3230543.3230545 event-place: Budapest, Hungary.

[10] V. Jacobson. 1988. Congestion Avoidance and Control. *ACM Computer Communications Review* (Aug. 1988), 314–329.

[11] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2013. B4: Experience with a Globally-deployed Software Defined Wan. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13)*. ACM, New York, NY, USA, 3–14. https://doi.org/10.1145/2486001.2486019 event-place: Hong Kong, China.