Severe Loss is not Rare in High-speed Datacenter Networks
--- Response to "A Critique of Aeolus"

Recently, we notice that there are some arguments [1] about whether severe loss is possible in high-speed datacenter networks (DCNs). In this article, we would like to explain the theory how modern switch chip works and demonstrate with concrete numbers that severe packet loss is not rare under incast with high-speed DCNs.

1. High-speed DCN switch is becoming more shallow-buffered

	Broadcom 56538	Broadcom Trident+	Broadcom Trident II	Broadcom Tomahawk3
Capacity	48port × 1Gbps	48port × 10Gbps	32port × 40Gbps	32port × 100Gbps
Total Buffer	4MB	9MB	12MB	16MB (4 MMUs)
Buffer per port	85KB	192KB	384KB	512KB
Buffer per port per Gbps	85KB	19.2KB	9.6KB	5.12KB

Table-1. Buffer and capacity information of commodity datacenter switching chips [2]

In recent years, datacenter network (DCN) link speed grows rapidly from 1/10Gbps to 100Gbps. In contrast, the buffer size of commodity switches does not increase as expected. In Table-1, we have listed the buffer and capacity information of some widely used commodity switching chips. As we can see, the link capacity significantly outpaces the buffer size, resulting in decreasing buffer per port per Gbps (from 85KB to 5.12KB).
There are mainly two reasons that make high-speed switching chips even more shallow-buffered. First, The memory used in switch buffers is high-speed SRAM which is very expensive. Second, The chip area increases with the memory size, which means a larger buffer may incur a longer memory access latency and it would be really hard for the memory access speed to match the high link speed if the buffer size is too large. Due to above two reasons, we envision that such trend will also hold for future 200/400Gbps switching chips.

2. How shared-buffer switch works

Most of today's commodity switching chips use Dynamic Threshold (DT) algorithm [3] for dynamic buffer allocation. The shared buffer allocated to a queue is controlled by a parameter α. At time t, a threshold T(t) is used to limit the queue length. Let *B*, *N* and $Q_i(t)$ be the total switch buffer size, total number of switch egress queues and buffer occupancy of queue i at time t, respectively. T(t) is calculated as follows:

$$T(t) = \alpha \times (B - \sum_{i=1}^{N} Q_i(t)) \quad (1)$$

A packet arriving in queue i at time t will get dropped if $Q_i(t) >= T(t)$. As analyzed in [3], if there are M active queues, each queue can eventually get $\alpha \times B/(1 + M \times \alpha)$ buffer space. Obviously, the more active queues we have, the smaller buffer space each queue can get from the shared buffer pool. Moreover, a large α can help a queue to get more buffer space. But a too large α can cause short-term imbalanced buffer allocation. Typically, α values are set powers of 2 for implementation simplicity (e.g., 1/128 to 8 in Tomahawk).

# of active queues / α	1	4	8	16	32	64
4	12.80MB	3.76MB	1.94MB	0.98MB	0.50MB	0.25MB
1/4	3.20MB	2.00MB	1.33MB	0.80MB	0.44MB	0.24MB
1/16	0.94MB	0.80MB	0.67MB	0.50MB	0.33MB	0.20MB
1/64	0.25MB	0.24MB	0.22MB	0.20MB	0.17MB	0.13MB

Table-2. Switch buffer an active queue can get with varying # of active queues and typical α values (Taking 32port × 100Gbps Broadcom Tomahawk switching chip with 16MB buffer as an example.)

In Table-2, we have calculated the switch buffer an active queue can get with varying number of active queues and typical α values.

In practice, there are several other constraints that will further reduce the size of buffer an active queue can utilize for packet buffering. First, the architecture of high-speed switching chips may consist of several smaller chip-lets for the purpose of faster memory access (e.g., a 32-port 16MB-buffer Broadcom Tomahawk switching chip consists of 4 8-port 4MB-buffer switching chip-lets.). This architecture will limit the maximum buffer an active queue can get to be 1/K of total buffer, where K is the total number of switching chip-lets (e.g., k = 4 for Broadcom Tomahawk chip). Second, when processing incoming packets at both ingress and egress processing pipelines, part of switch buffer will be used to store some metadata of packets, which will consume a certain amount of switch buffer. Third, production DCNs often serve several traffic classes simultaneously. One widely adopted strategy is to support each traffic class with separate buffer space. For example, it is common to allocate half of the switch buffer for supporting RDMA traffic, and let the remaining half to support TCP traffic. This type of allocation strategy will also reduce the buffer size an active queue can utilize. In summary, the size of switch buffer an active queue can utilize is often much less than the values calculated in Table-2.

3. Severe packet loss under incast-like traffic is not rare
   Many-to-many incast-like traffic is common in production DCNs. With such traffic, it is easy for ToR switches to have multiple active queues that need to buffer incast traffic simultaneously. Assuming 100Gbps link speed and 8us base RTT, in Table-3, we calculate the buffer size needed by an active queue to buffer incast traffic in the first RTT with varying incast degress (i.e., the number of concurrent flows) and message size. We calculate the needed buffer size as *message_size * incast_degree – BDP*.

Message size / Incast degree	10KB	20KB	40KB	80KB
20	0.1MB	0.3MB	0.7MB	1.5MB
40	0.3MB	0.7MB	1.5MB	3.1MB
80	0.7MB	1.5MB	3.1MB	6.3MB
160	1.5MB	3.1MB	6.3MB	12.7MB
320	3.1MB	6.3MB	12.7MB	25.5MB

Table-3. Switch buffer size needed by an active queue to buffer incast traffic in the first RTT with varying incast degress and message size.

By comparing the values in both Table-2 and Table-3, it is easy to find **severe packet loss under incast-like traffic is not rare in high-speed DCNs.** For example, with 80 incast degree and 20KB message size, more than 75% of the conditions presented in table-2 will suffer from severe packet loss.

4. Is lossless property the right assumption for high-speed DCNs?
   Some recent work, such as Homa[4], establishes its design by assuming DCN switches have enough buffer for buffering all the bursty traffic in the first RTT. However, as we have demonstrated with concrete numbers in previous sections, switches are becoming more shallow-buffered at high-speed, and packet loss is easier to occur in the first RTT under incast.
   Furthermore, even if major cloud providers are willing to pay the expensive cost to deploy switches with adequate buffer to absorb all the first-RTT traffic burst, it may not be a right choice. The reason is that, once a large queue is built in the network in the first RTT, the end-host transports can do nothing but wait until the queue is slowly drained and incur a large delay to all flows passing through the port which hurts even more for modern Internet services as latency is a more critical metric than throughput nowadays. In contrast, if we drop the overwhelming traffic in the first RTT, there will be no large queue buildups and persistent end-to-end low latency is well preserved. Packet loss is not a problem as long as senders are notified about the loss quickly. This is the exact philosophy behind Aeolus [5] --- dropping overwhelming first-RTT traffic in the network earlier to preserve low end-to-end latency, and quickly notifying the sender to retransmit lost data since second RTT according to the available bandwidth it shares.

References

1. https://stanford.edu/~ouster/cgi-bin/aeolus.php
2. Wei Bai, Shuihai Hu, Kai Chen, Kun Tan, and Yongqiang Xiong. 2021. One More Config is Enough: Saving (DC)TCP for High-Speed Extremely Shallow-Buffered Datacenters. IEEE/ACM Trans. Netw. 29, 2 (April 2021), 489–502.
3. K. Choudhury and E. L. Hahne, “Dynamic Queue Length Thresholds for Shared-memory Packet Switches,” IEEE/ACM Trans. Netw., vol. 6, no. 2, pp. 130–140, Apr. 1998.
4. Behnam Montazeri, Yilong Li, Mohammad Alizadeh, and John K. Ousterhout. Homa: A receiver-driven low-latency transport protocol using network priorities. SIGCOMM’18
5. Shuihai Hu, Wei Bai, Gaoxiong Zeng, Zilong Wang, Baochen Qiao, Kai Chen, Kun Tan, and Yi Wang. 2020. Aeolus: A Building Block for Proactive Transport in Datacenters. In Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (SIGCOMM ’20).