National University of Singapore
School of Computing
CS5229: Advanced Computer Networks
Semester I, 2021/2022

Lecture 6 Training
Datacenter Networking

In Lecture 6, we discussed how the partition-aggregate traffic patterns can lead to the phenomenon of Incast leading to TCP timeouts. TCP timeouts cause applications to not meet their deadlines which hurts user experience and affects revenue. We also discussed DCTCP, a congestion control scheme for datacenter networks which helps keep queue occupancy low. Low queue occupancy provides sufficient headroom in the queue to absorb Incast traffic bursts. In this lecture training, we will investigate how buffer sizing and the choice of TCP congestion control can affect application performance in datacenter networks.

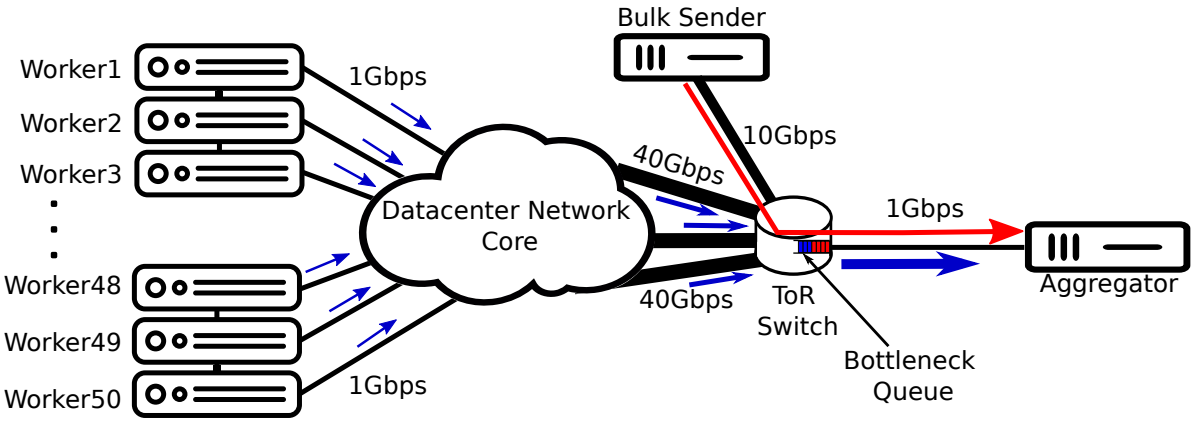
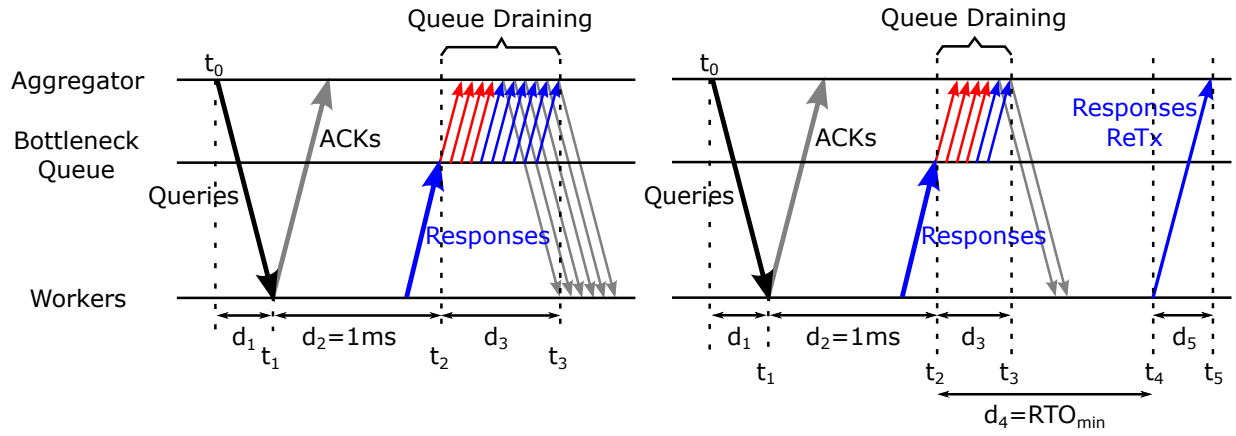


Figure 1: Incast in a datacenter network

Figure 1 shows an Incast scenario where an aggregator has queried 50 workers nodes and the worker nodes are sending back responses to the aggregator in a synchronized manner. The link between the top-of-rack (ToR) switch and the aggregator server becomes the bottleneck for such an Incast. Additionally, a bulk sender may also be sending a long running TCP data flow to the aggregator server. The link speeds are as shown in Figure 1. In particular, the bottleneck link's speed is 1 Gbps. The RTT between any worker and the aggregator is 120 µs. The RTT between the bulk sender and the aggregator is also 120 µs.

Query Completion Time

We are going to use query completion time (QCT) as the metric to evaluate the performance of the partition-aggregate application that is being run by the aggregator server. To make our analysis easier, we are going to consider a simplified partition-aggregate traffic pattern. In our partition-aggregate application, the size of each packet is 1500 bytes. The size of query response from each worker is exactly 3 KB (2 packets). Also, if the response from a worker gets delayed due to Incast packet loss, the aggregator chooses to wait for the worker to retransmit rather than skipping the worker's response.



(a) Incast without packet loss (b) Incast with packet loss

Figure 2: Partition-Aggregate traffic pattern without and with Incast packet loss. The red arrows denote that there could be existing packets in the bottleneck queue (from the bulk sender) when the Incast occurs.

Incast without packet loss

Figure 2a shows the case where no packet is lost during the Incast. The overall process unfolds as following: (i) At t_0, the aggregator first sends the queries to the workers simultaneously[1] i.e. the queries will reach all the workers at the same time (t_1). Assume that the queries do not face any congestion and therefore reach the workers after the network's one-way delay (RTT/2)[2]. (ii) The workers start processing the queries immediately without waiting for the corresponding ACKs to reach back. (iii) For each worker it takes exactly 1 ms to process the query and send the response back through the network till the bottleneck queue. (iv) At t_2, the responses from all the workers arrive synchronously in the bottleneck queue. This synchronous arrival of responses from all the workers leads to instantaneous increase in the bottleneck queue size i.e. if the initial queue size *just* before the Incast was q_0 bytes (due to a bulk sending flow) and the total response size from all workers is R bytes, then the queue size will increase instantaneously to $q_0 + R$ bytes. In case of Figure 2a, the queue has enough headroom to absorb the Incast. Therefore, no Incast packets will be lost. (v) The instantaneous queue build-up will then drain at the bottleneck link's speed. After the Incast, once (q_0+R) bytes have drained from the queue, the aggregator has received all the responses and the query is complete at t_3. Note that the link propagation delay between the bottleneck queue and the aggregator is negligible. Therefore, d_3 corresponds to the time it takes to dequeue (q_0+R) bytes at the link speed. The query completion time (QCT) in case of Figure 2a is the time between t_0 and t_3. In other words, the QCT is the sum of all the involved delays:

$$QCT = d_1 + d_2 + d_3 \tag{1}$$

Incast with packet loss

Figure 2b shows the Incast scenario with packet loss. In this scenario, everything works the same till the point that Incast occurs. When the Incast occurs at t_2, the bottleneck queue does not have enough headroom to absorb all the response packets and therefore

[1] In practice, the queries would be sent sequentially over 100s of µs. This time being small can be ignored for simplicity.

[2] For d_1, the link serialization delay is ignored since the queries are small.

at least some response packets from the workers are lost. The response size from each worker is only 2 packets. Therefore, we can assume that, there would be at least one worker who would either lose both the packets or lose just the second packet. In either case, there is no way for the worker to know if any response packet was lost, other than waiting for the ACKs from the aggregator. For the lost packets, there won't be any ACK from the aggregator and therefore the worker waits for the retransmission timeout (RTO_{min}) before retransmitting the response packet(s) for which no ACK was received. As a result, in this case, the query would complete only at t_5. In our setup, the value of RTO_{min} is **300 ms**[3]. For simplicity, assume that even if multiple workers lose packets in the Incast, those multiple workers would retransmit synchronously and the retransmitted packets won't face any Incast and won't be dropped. Also, assume that the retransmitted packets would not face any queuing delay at the bottleneck queue and would reach the aggregator after the network's one-way delay (RTT/2)[4]. Therefore, the QCT in this case would be the time between t_0 and t_5. In other words, the QCT would be the sum of the following delays:

$$QCT = d_1 + d_2 + d_4 + d_5 \tag{2}$$

Note that we are not considering the queue draining delay (d_3) here since RTO_{min} is typically much much larger than the maximum possible value of d_3.

Scenarios

Consider the following 4 scenarios:

(A) The bottleneck queue size is 145 KB and there is no traffic from the bulk sender. In other words, the bottleneck queue is empty and then a single Incast occurs from the 50 workers.

(B) Same as scenario (A) except that the bottleneck queue size is increased to 200 KB.

(C) The bottleneck queue size is 200 KB and a long running TCP Reno flow from the bulk sender is occupying the queue before the Incast occurs.

(D) The bottleneck queue size is 200 KB and a long running DCTCP flow from the bulk sender is occupying the queue before the Incast occurs. The DCTCP flow uses the following parameters:

- Marking threshold: $K = 20$ packets (30 KB)
- Estimation gain: $g = \frac{1}{16}$

Given this information, answer the following questions on Coursemology:

Q1 What will be the QCT in scenario (A)?

Q2 What will be the QCT in scenario (B)?

Q3 What will be the QCT in scenario (C)?

Q4 What will be the QCT in scenario (D)?

[3]Ideally RTO_{min} is counted from the time the original response is sent. In Figure 2b, since d_2 accounts for time slightly after the response is sent, the specified value of RTO_{min} is already adjusted accordingly.

[4]For d_5, the link serialization delay is ignored since the reponses are very small.

While answering any of the above questions, if your calculations shows that the QCT would vary, then please indicate the minimum and maximum QCT.

Hint: For scenarios (C) and (D), the bottleneck buffer's queue occupancy will vary between a minimum (q_{min}) and a maximum (q_{max}) value depending on Reno's and DCTCP's steady-state behavior. Therefore, in scenarios (C) and (D), you would need to calculate QCT for the two extreme cases: (i) the queue occupancy is q_{min} when the Incast occurs, and (ii) the queue occupancy is q_{max} when the Incast occurs. Please refer to the following references subsection for additional help on q_{min} and q_{max}.

References

Let q_{max} and q_{min} denote the maximum and minimum queue occupancy of a congestion control algorithm in the steady state.

TCP Reno

You have already done TCP Reno's steady-state analysis in Lecture 3 training. The following equations are for your ready reference to analyze TCP Reno's steady-state:

$$q_{max} = queueSize \tag{3}$$

$$cwnd_{max} = BDP + queueSize \tag{4}$$

$$cwnd_{min} = \frac{cwnd_{max}}{2} \tag{5}$$

$$q_{min} = \begin{cases} cwnd_{min} - BDP, & \text{if } cwnd_{min} > BDP \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

DCTCP

For DCTCP's q_{max} and q_{min} in the steady state, please refer to the equations 8, 10 and 11 in the DCTCP paper. While calculating q_{max} and q_{min} for DCTCP, please calculate in terms of number of packets (rounded off to nearest whole packet) first. Then convert to KB using packet size of 1.5 KB.