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

Lecture 1 Training
End-to-End Argument

Release date: 14th August 2021
Due: 19th August 2021, 23:59

In Lecture 1, we briefly discussed the end-to-end argument in system design and mentioned that there were trade offs.

In this question, we will investigate the impact of these trade offs on practical metrics that we care about. In particular, you are to determine the flow completion time (FCT, time between the first packet sent from the server to the last packet received at the receiver) and round-trip times (RTT) for different network scenarios.

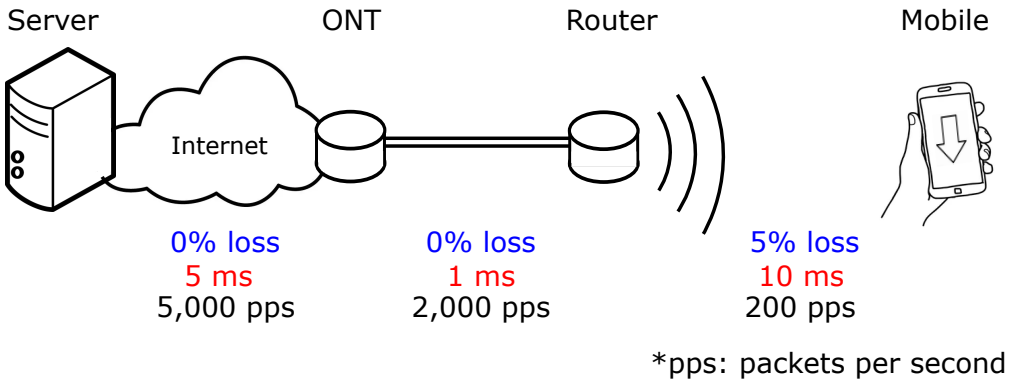


Figure 1: Simple wireless network.

Consider the network in Figure 1. Suppose the mobile end host wants to download a file consisting of 1,000 packets from the server. Each wired/wireless link in this topology has some loss rate, one-way delay, and bandwidth (in packets per second) associated with it (specified in Figure 1).

For simplicity, you can assume that there are no other competing flows in the network and that both retransmitted packets or acknowledgement packets will suffer no loss. All nodes in the network use the NACK-like retransmission policy described in Figure 2.

Basically, the packets will contain a sequence number and the receiver assumes that there is no reordering, so if it detects a skipped sequence number, it will immediately inform the sender of this missing packet via a NACK. The sender will retransmit the missing packet once it receives the NACK.

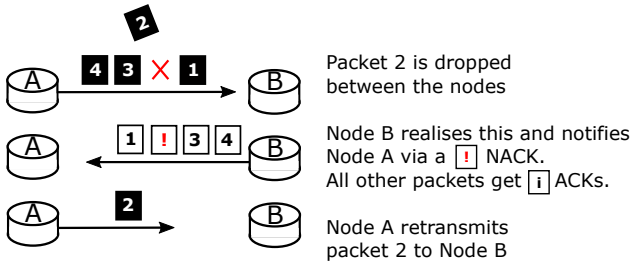


Figure 2: Retransmission policy

Given this information, answer the following questions on Coursemology:

1. What is the RTT and FCT if the loss rate at the wireless link is zero?
2. What is the expected ($E(x)$) RTT and FCT for downloading the 1,000 packet file when the loss rate at the wireless link 5%, for each of the following scenarios:
   (a) **Only** end-to-end retransmission are allowed
   (b) Link re-transmissions are allowed in the wireless link between the Router and the Mobile

You are not expected to do any complex modelling or mathematical computations to derive the above answers. In networking, we often do back-of-the-envelope calculations allow us to check that our experiments are yielding results that are within expectation.

Solutions

1. The FCT is the sum of the transmission time (time taken to put packet on the wire), plus the propogation time (time taken for the packets to travel to the receiver). Since the bottleneck bandwidth for this connection is 200 pps, we will use this bandwidth to calculate the transmission time. We will use the path's one-way delay to calculate the propogation delay.
   FCT = transmission time + propogation time=$\frac{1000}{200} + (0.005 + 0.001 + 0.010)$ = **5.016 s**

   The RTT, or the *round-trip time* of a connection is simply twice the one-way delay:
   RTT = $2 \times (5 + 1 + 10) =$ **32 ms**
2. With packet losses (5%)
   (a) *end-to-end retransmits:* 5% packet loss means the sender sends a total of 1,050 packets instead of 1,000 packets. Plugging in these values in the equation used earlier we get,
   FCT =$\frac{1050}{200} + (0.005 + 0.001 + 0.010)$ = **5.266 s**
   An end-to-end retransmission will have no effect on the RTT.
   Therefore, $E($**RTT**$) =$ **32 ms**
   (b) *link-level retransmits:* Following the same logic as before,
   FCT =$\frac{1050}{200} + (0.005 + 0.001 + 0.010)$ = **5.266 s**

   Note: You might be tempted to think that because the retransmits are happening at the link-level instead of end-to-end, the FCT is should be smaller because the RTT of path traversed by the retransmitted packets is smaller. But the length of this traversed path (which shows up as propogation delay in

the FCT equation) only determines how quickly your *first* packet can reach the receiver. After that the FCT is dominated by the transmission delay because the packets travel in a stream. This delay only depends on the bottleneck bandwidth and **not** the RTT of the path.

However, link-level retransmissions **will** have an effect on the RTT. Since the end host is blind to the link-level retransmission, the retransmission will show up as extra delay in the ACK for the lost packet. Since we know how many packets are lost and how much extra delay these retransmission will cause, we can calculate the RTT as follows:
$E(\textbf{RTT}) = 32 \times 0.95 + (32 + \textbf{20}) \times 0.05 = \textbf{33 ms}$
Here, the extra 20 ms is for the NACK to travel from the Mobile to the Router, and the retransmitted packet to travel from the Router to the Mobile.

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

Lecture 2 Training
Window-Based End-to-End Congestion Control

Release date: 20th August 2021
Due: 26th August 2021, 23:59

In Lecture 2, we discussed how TCP does a slow start in the beginning of the transmission. In this lecture training, we will investigate how a sender's congestion window (`cwnd`) evolves during the slow start (on a per-RTT basis) and how it impacts network congestion. We will also examine the effect of the initial congestion window size on the slow start and how it impacts user experience, especially during web browsing. Consider the following network scenario:

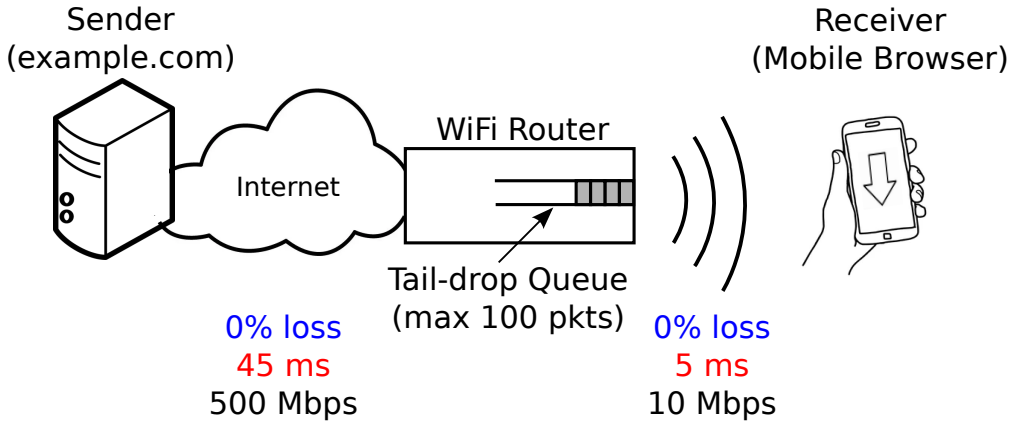


Figure 1: A simple network scenario

The mobile browser (TCP receiver) in Figure 1 wants to download a webpage from a server hosting `example.com` (TCP sender).

Unlike Lecture 1 training, assume that there are no transmission losses in the network for simplicity. Packet loss can only happen when the queue on the WiFi router overflows (> 100 packets). The link speeds are in Mega bits per second, where 1 Mbps = 1,000,000 bits per second. Also, assume that the WiFi router adds no hop delay or queueing delay (no matter the queue size). We consider the size of each data packet to be 1,500 bytes. Furthermore, assume[1] that each packet can carry 1,500 bytes of TCP payload. For example, the sender would need 2 packets to transmit a webpage of 3 KB (3,000 bytes) to the receiver.

The sender uses the following slow start mechanism which proceeds in time intervals of the RTT: (i) In the first RTT interval, the sender's initial `cwnd` is equal to 2 packets. The sender bursts these 2 packets and receives corresponding 2 ACKs from the receiver. (ii)

[1] In practice, a typical MTU-sized packet is 1,518 bytes long "on the wire" and can carry only 1,460 bytes of TCP payload: Ethernet header & checksum (14 + 4 = 18 bytes) + IP header (20 bytes) + TCP header (20 bytes) + TCP payload (1,460 bytes)

For each subsequent RTT interval, the sender increases the `cwnd` by 1 packet for every ACK it receives during the previous RTT interval. (iii) The receiver always sends 1 ACK for each packet received from the sender. This means that, the sender would effectively double its `cwnd` in each subsequent RTT interval. (iv) The sender would exit the slow start and switch to congestion avoidance if/when it detects the first packet loss.

Given this information, answer the following questions on Coursemology:

Q1 In which RTT[2] will the queue in the WiFi router start building up (queue size greater than 0)? Note that the RTT numbering starts with 1 i.e. in RTT #1, the sender sends 2 packets and receives 2 ACKs.

Q2 In which RTT will the first packet drop occur at the WiFi router?

Q3 Suppose the mobile browser needs to download a webpage of 90 KB (90,000 bytes) from `example.com`. How many RTTs would it take to complete the webpage download? A webpage download is considered to be "complete" when the sender receives the ACK for the last packet sent.

Next, suppose we upgrade our WiFi router and now our wireless link has a link speed of 100 Mbps.

Q4 In this new scenario, in which RTT will the queue in the WiFi router start building up?

Q5 In which RTT will the first packet drop occur at the WiFi router?

Q6 Suppose we want to download a webpage of 90 KB (90,000 bytes) from `example.com` again. How many RTTs would it take to complete the webpage download?

Instead of upgrading the WiFi router, suppose we increased the sender's initial `cwnd` to 10 packets and kept the wireless link's speed at 10 Mbps.

Q7 In this case, how many RTTs will it take to download the same 90 KB webpage as above?

Solutions

Q1 The queue will start building up only when queue's enqueue rate is greater than its dequeue rate. Now, the queue's enqueue rate is equal to the sender's sending rate while its dequeue rate is equal to the wireless link's speed i.e. 10 Mbps.

Let's compute the sender's sending rate first. The TCP sender sends `cwnd` worth of packets over a time period of 1 RTT. Therefore, the sending rate is cwnd/RTT. However, since the `cwnd` changes during each RTT, the effective sending rate in each RTT depends on the `cwnd` in that RTT. The sender starts with an initial `cwnd` of 2 packets in the first RTT and doubles its `cwnd` in each subsequent RTT. Therefore, the sender's `cwnd` in the n$^{\text{th}}$ RTT is 2^n. The enqueue rate in the n$^{\text{th}}$ RTT would then be:

$$enqueue\ rate = sending\ rate = 2^n\ pkts/RTT_{100ms} \quad (1)$$

The unit here is to be read as "packets per RTT". We use the subscript 100 ms to denote that the RTT is 100 ms in this case. To compare this enqueue rate with the

[2] we refer to a RTT time interval simply by "RTT". Note that the time is divided into intervals of the RTT.

dequeue rate, we need to first convert the dequeue rate from Mbps to "packets per $\text{RTT}_{100\text{ms}}$".

$$\begin{aligned} dequeue\ rate = 10\,Mbps &= \frac{10 \times 10^6}{8}\,bytes/s = \frac{125 \times 10^4}{1500}\,pkts/s \\ &= \frac{833.33 \times 100}{1000}\,pkts/RTT_{100ms} = 83.33\,pkts/RTT_{100ms} \end{aligned} \tag{2}$$

To answer the question, we need to find the first RTT in which the enqueue rate would be greater than the dequeue rate:

$$\begin{aligned} enqueue\ rate &> dequeue\ rate \\ 2^n &> 83.33 \end{aligned} \tag{3}$$

For $n = 7$, the enqueue rate is $128\,\text{pkts/RTT}_{100\text{ms}}$ which is greater than the dequeue rate. Therefore, the queue will start building up in the **7th RTT**.

Q2 To find the RTT in which the first packet drop will occur, we need to track how the queue will build over each RTT. As determined in Q1 above, for 10 Mbps wireless link speed, the queue will not start building until the 7th RTT. For example, in the 6th RTT, the sender will enqueue 64 packets while wireless link will dequeue 83.33 packets. The queue size at the end of an RTT is the queue size at the end of the previous RTT plus the extra packets added to the queue in the current RTT. Therefore, the queue size at the end of nth RTT can be written as:

$$queueSize_n = queueSize_{n-1} + \ extraPackets_n \tag{4}$$

Here, $extraPackets_n$ are the number of extra packets added to the queue in the nth RTT. The number of extra packets is the difference between the packets enqueued be the sender and the packets dequeued by the wireless link. Therefore,

$$extraPackets_n = 2^n - 83.33 \tag{5}$$

$$\therefore\ queueSize_n = queueSize_{n-1} + 2^n - 83.33 \tag{6}$$

Now, let's find the queue size at the end of the 7th RTT using equation 6. Note that the queue size at the end of the 6th RTT is zero:

$$queueSize_7 = queueSize_6 + 2^7 - 83.33 \tag{7}$$

$$= 0 + 128 - 83.33 = 44.67\,pkts \tag{8}$$

Let's continue this analysis to the further RTTs.

$$queueSize_8 = queueSize_7 + 2^8 - 83.33 \tag{9}$$

$$= 44.67 + 256 - 83.33 = 217.34\,pkts \tag{10}$$

Since the queue can only hold 100 packets, the first packet drop would occur in the **8th RTT**.

Q3 To transmit a 90 KB webpage, the sender would need to send 90,000/1500 = 60 packets. We know that in nth RTT, the sender sends 2^n packets. Therefore, we need to sum up the total packets sent by the sender in each RTT starting from the beginning:

$$2^1 + 2^2 + 2^3 + 2^4 + 2^{\mathbf{5}} = 62\,pkts > 60\,pkts \tag{11}$$

Therefore, it would take **5 RTTs** to complete the webpage download.

Q4 In the new scenario, due to the increase in the wireless link speed, the dequeue rate has increased 10 times compared to Q1 i.e. it has now become 833.33 pkts/$\text{RTT}_{100\text{ms}}$. Therefore, on the same lines as equation 3, we need to find the first n for which the following equation becomes true:

$$2^n > 833.33 \tag{12}$$

For $n = 10$, the enqueue rate is 1024 pkts/$\text{RTT}_{100\text{ms}}$ which is greater than the dequeue rate. Therefore, in this case, the queue will start building up in the **10^{th} RTT**.

Q5 For this question, we can do the same analysis as in Q2. The only thing different would be that the wireless link would dequeue 833.33 packets in every RTT. Also, from Q4, we know that the queue will start building up from the 10^{th}. Therefore, using equation 6 (modified: 83.33 $\rightarrow$ 833.33) for the 10^{th} RTT, we get a queue size of:

$$queueSize_{10} = queueSize_9 + 2^{10} - 833.33 \tag{13}$$

$$= 0 + 1024 - 833.33 = 190.67\, pkts \tag{14}$$

Since the queue can only hold 100 packets, the first packet drop would occur in the **10^{th} RTT**.

Q6 Same as Q3 i.e. **5 RTTs**. Upgrading the WiFi router for 100 Mbps wireless link speed does not change how the `cwnd` would evolve during the slow start. To affect the evolution of `cwnd` during the slow start, a packet loss needs to occur. There is no packet loss happening in the 10 Mbps case which could have been averted by increasing the wireless link speed to 100 Mbps and delaying the packet loss to a further RTT. In fact, at 10 Mbps, 60 packets (the 90 KB webpage) are transmitted in 5 RTTs, which is even before the queue would start building in the 7^{th} RTT. Therefore, with 100 Mbps wireless link, it would still take 5 RTTs to complete the webpage download since the `cwnd` will evolve exactly in the same way as it would with the 10 Mbps wireless link.

Q7 With 10 packets as the initial `cwnd`, the packets sent in successive RTTs would be 10, 20, 40 and so on. Since 10 + 20 + 40 > 60 pkts, it is clear that in this case it would take **3 RTTs** to complete the webpage download. This is a 40% improvement compared using an initial `cwnd` of 2 packets.

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

Lecture 3 Training
Buffer sizing and AQMs

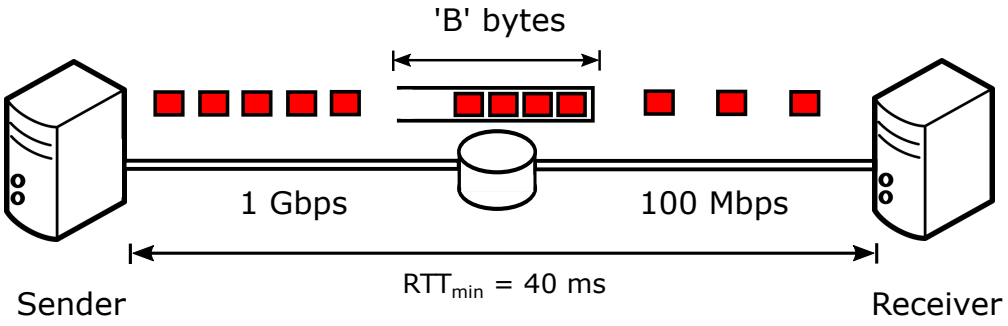


Figure 1: A simple network with a 100 Mbps bottleneck

In Lecture 3, you learnt how sizing the router buffer is crucial to ensuring link utilization with loss-based congestion control algorithms like Reno. In this training, we will explore the effects of buffer sizes on the throughput and RTT a little more.

Let's look at a simple network with 2 hosts with a 100 Mbps bottleneck link between them (Figure 1). Let the bottleneck buffer have a maximum capacity of B bytes and let RTT when the buffer is empty (RTT_{min}) be 40 ms. From the lecture, we know that if B is too small, we risk underutilizing the bottleneck. On the other hand if B is too large, there will be buffer bloat, which will increase the RTT of the connection (see Figure 2).

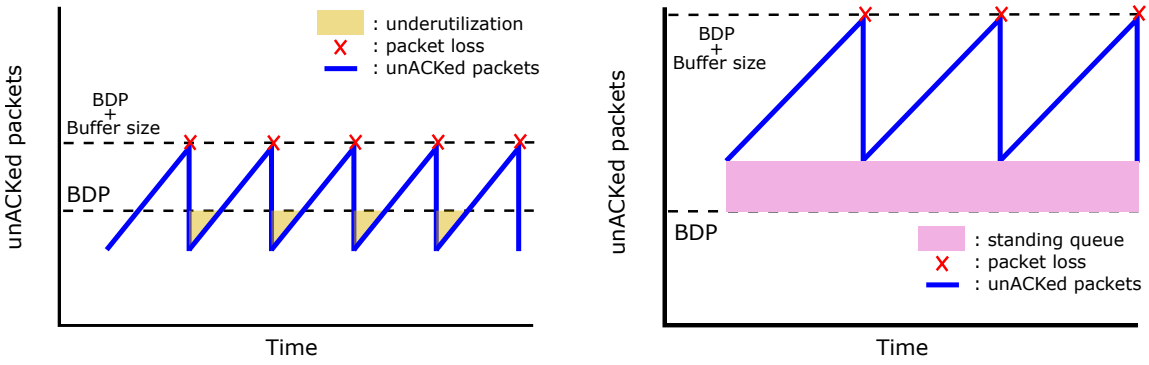


Figure 2: unACKed packets vs. time in different buffer sizes.

Assuming the sender starts a long flow (long enough for the slow start to be negligible) and uses Reno to do congestion control, answer the following questions. For simplicity sake, you can assume the time taken to drain the queue is negligible and the `cwnd` always matched the number of unACKed packets. Each packet is 1,000 bytes.

Questions

1. Calculate the average throughput and the minimum and maximum RTT when the bottleneck buffer is:
   (a) 100 Kilobytes
   (b) 500 Kilobytes
   (c) 1 Megabyte
2. In your opinion, which of the above mentioned buffer sizes is most suitable for this network, and why?

Solutions

Before answering the questions, we will first calculate the BDP of the network. This will make working easier later on:
$\text{BDP} = \text{Bandwidth} \times \text{RTT} = 100\,\text{Mbps} \times 40\,\text{ms}$
$\text{BDP} = 500,000\,\text{bytes}$

1. The average throughput whenever there is atleast 1 packet in the buffer is going to be 100 Mbps. However, when the buffer is empty, we will need to use the packets in the pipe to calculate the throughput. For calculalting the minimum and maximum RTTs, we will need to calculate the minimum and maximum queue size. If the queue size is zero, the minimum RTT will simply be 40 ms.
   (a) When the buffer size is 100 KB, the sender will be able to have a maximum `cwnd` of 600 KB and a minimum `cwnd` of $\frac{600}{2} = 300$ KB. Since Reno increases its `cwnd` by 1 packet every RTT, it will take $(500 - 300) = 200$ RTTs to fill the pipe again after a backoff. After this, the `cwnd` will continue to rise for 100 more RTTs after which there will be a packet loss in the 301^{st} RTT and the cycle will continue again.

   In this scenario since the buffer empties, the average throughput will not be the bottleneck bandwidth. We will need to use the packets in the pipe to calculate the throughput.

$$\text{Total data sent}(D) = \sum_{i=0}^{300} 300,000 + 1000i = 135,450,000 \text{ bytes}$$

   The amount of data calculated above is sent for 301 RTTs. In the first 201 of those RTTs, the buffer is empty and therefore the RTT = 40 ms. However, for the next 100 RTTs, there is a queue in the bottleneck buffer. This queue rises from 0 to 100 KB in those 100 RTTs. We can calculate the total time elapsed during the last 100 RTTs as follows:

$$= \sum_{i=1}^{100} 0.04 + i \times (0.00008) = 4.404 \text{ seconds}$$

   Therefore, the total time will be

$$\text{Total time}(t) = 201 \times 0.04 + 4.404 = 12.444 \text{ seconds}$$

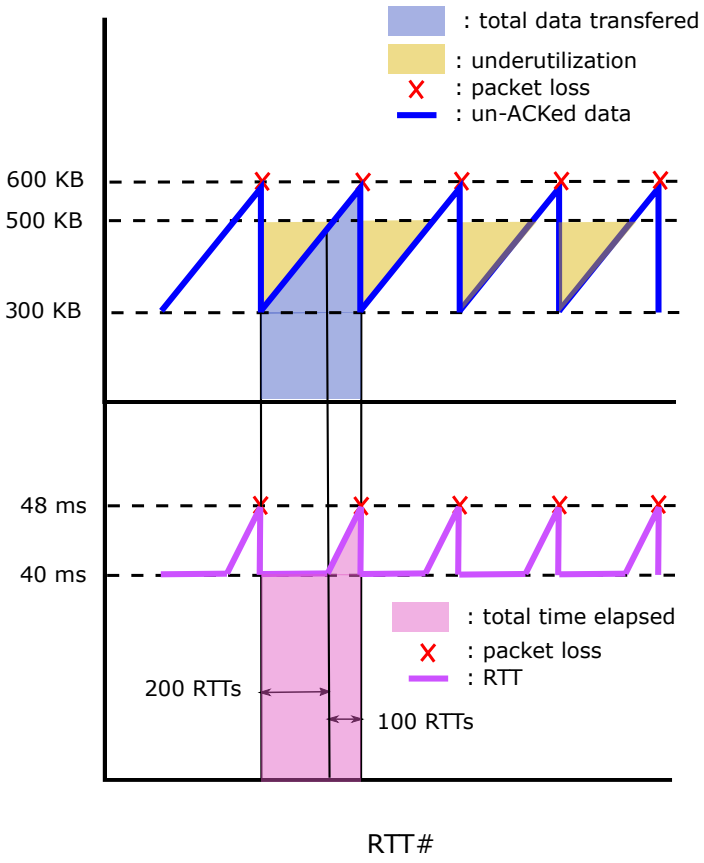


Figure 3: Variation in cwnd and RTT

$$\textbf{Avg. Throughput}(\frac{D}{t}) = \textbf{87.078 Mbps} \sim 87.08\text{Mbps}$$

Since the buffer empties in this scenario, **Minimum RTT = 40 ms**

The maximum queue size in this network will be 100KB, which will cause $\frac{100\times1000\times8}{100\times1,000,000} = 0.008$. Therefore, **Maximum RTT = 48 ms.**

(b) When the buffer size is 500 KB, the sender will be able to have a maximum cwnd of 1,000 KB and a minimum cwnd of $\frac{1,000}{2} = 500$ KB. Therefore, the pipe will always be full. Hence, **Avg. Throughput =100 Mbps**

Since the buffer also (just) empties in this scenario, **Minimum RTT = 40 ms**

The maximum queue size in this network will be 500KB, which will cause $\frac{500\times1000\times8}{100\times1,000,000} = 0.04$. Therefore, **Maximum RTT = 80 ms.**

(c) When the buffer size is 1 MB, the sender will be able to have a maximum cwnd of 1,500 KB and a minimum cwnd of $\frac{1,500}{2} = 750$ KB. Again, the pipe will always be full. Hence, **Avg. Throughput =100 Mbps**

The minimum queue size in this network will be 250KB, which will cause $\frac{250\times1000\times8}{100\times1,000,000} = 0.02$. Therefore, **Minimum RTT = 60 ms.**

The maximum queue size in this network will be 1000KB, which will cause $\frac{1000\times1000\times8}{100\times1,000,000} = 0.08$. Therefore, **Maximum RTT = 120 ms.**

2. A buffer size of 500 KB is clearly the best buffer size for this network. It ensures that the link is fully utilized and does not cause buffer bloat.

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

Lecture 4 Training
Rate-based end-to-end Congestion Control

In lecture 4 we studied BBR, a rate-based congestion control algorithm proposed by Google in 2016. BBR uses a simple network model to estimate the BDP and infer congestion. It does so by estimating the bottleneck bandwidth and the minimum RTT, with the later being extremely hard to measure in the wild. A BBR flow can only measure the minimum RTT when the buffer is empty. This becomes impossible when BBR competes with traditional buffer-filling loss-based congestion control algorithms and causes it to *over-estimate* the minimum RTT. This over estimation has been found to make BBR extra aggressive when competing with loss-based flows.

In this training, we will mathematically try to understand the basis of this extra aggression. Consider the network described in Figure 1 where a BBR flow shares a bottleneck with a flow running a toy congestion control algorithm called *Taco*. *Taco* is a very simple congestion control algorithm that **always** keeps 20 packets in the bottleneck buffer. You need to calculate how *Taco*'s 20 packets in the buffer will impact BBR's RTT estimates, and consequently its bandwidth.

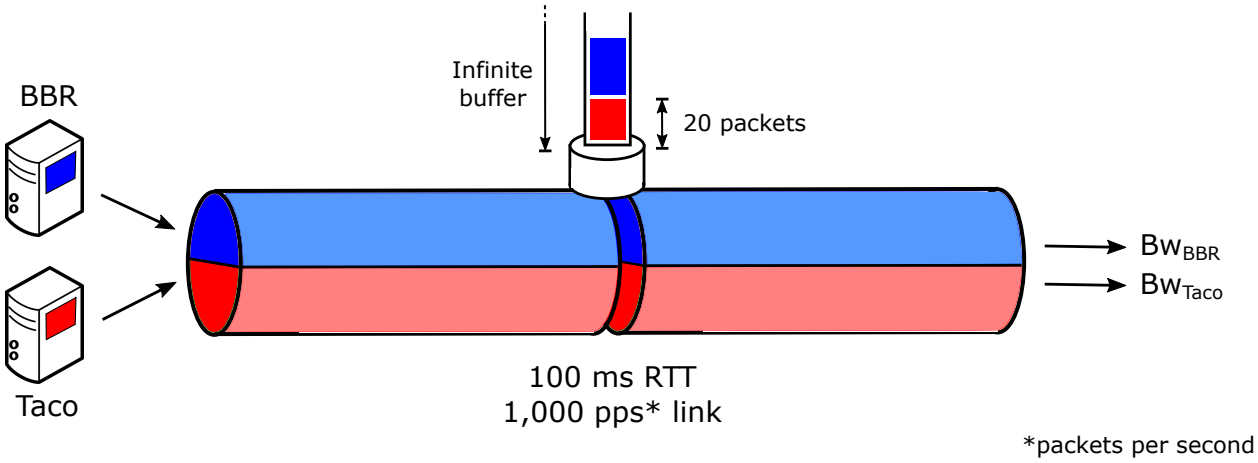


Figure 1: BBR competing with TCP Taco

To help you out, let's recap a key property associated with BBR. BBR's `cwnd` is equal to twice its estimated BDP. Therefore,

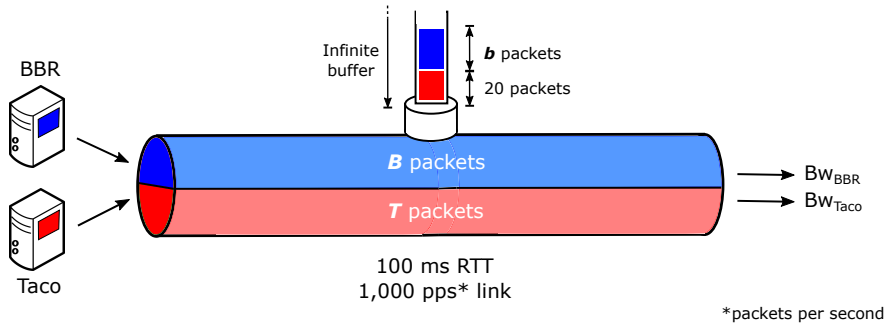
$$\texttt{cwnd} = 2 \times \bar{\mathbf{BDP}} = P_{pipe} + P_{buff} \tag{1}$$

Where $\bar{\mathbf{BDP}}$ is BBR's *estimated* BDP and P_{pipe} and P_{buff} are BBR's packets in the pipe and in the buffer respectively. Also, you can assume that the bottleneck buffer is FIFO and the bottleneck bandwidths received by the flows are directly proportional to their buffer occupancy.

Given these properties, answer the following questions in context of the network described in Figure 1:

1. Assuming the 20 *Taco* packets in the buffer contribute to BBR's RTT over-estimation, how many packets will BBR have in the bottleneck buffer?
2. When BBR does over-estimate the RTT, what is the bandwidth received by the two flows?
3. Let's say the BBR flow is run by an Oracle that knows the true minimum RTT. In other words there is no RTT over-estimation. In such a network, how many packets does the BBR flow place in the bottleneck buffer?
4. When there is no RTT over-estimation, what is the bandwidth received by the two flows?

Solutions



Let B and T be the total number of BBR and *Taco* packets in the pipe respectively, and let b be the total number of BBR packets in the bottleneck buffer. We already know that *Taco* has 20 packets in the buffer. From Equation 1 we know that,

$$B + b = 2 \times Bw_{BBR} \times RTT_{min} \tag{2}$$

With RTT over-estimation

With RTT over-estimation, BBR's estimate of the RTT will include the queueing delay caused by *Taco*'s 20 packets in the buffer. Also, $B = Bw_{BBR} \times (0.1)$. Therefore, Equation 2 can be re-written as:

$$(Bw_{BBR} \times (0.1)) + b = 2 \times Bw_{BBR} \times (0.12)$$

$$b = Bw_{BBR} \times (0.14) \tag{3}$$

In a FIFO buffer, the bandwidth the flows get is directly proportional to their buffer occupancy.

$$\frac{b}{b+20} \times 1,000 = Bw_{BBR} = \frac{b}{0.14} \tag{4}$$

Solving Equations 3 and 4, b=**120**, Bw_{BBR}=**857 pps**.

Without RTT over-estimation

Without RTT over-estimation, BBR will have an accurate estimate of the RTT_{min}. Also, $B = Bw_{BBR} \times (0.1)$. Therefore, Equation 2 can be re-written as:

$$(Bw_{BBR} \times (0.1)) + b = 2 \times Bw_{BBR} \times (0.1)$$

$$b = Bw_{BBR} \times (0.1) \tag{5}$$

In a FIFO buffer, the bandwidth the flows get is directly proportional to their buffer occupancy.

$$\frac{b}{b + 20} \times 1,000 = Bw_{BBR} = \frac{b}{0.1} \tag{6}$$

Solving Equations 5 and 6, b=**80**, Bw_{BBR}=**800 pps**.

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

Lecture 5 Training
Network-assisted Congestion Control

In Lecture 5, we discussed how congestion control can be driven entirely by the network i.e. the network can inform senders of the rate at which they should send. In this lecture training, we will investigate how rate allocations and network utilization can change depending on the notion of fairness adopted by the network.

Fair Rate vs. Max-Min Fair

Consider a network with 4 links with different capacities shown in Figure 1.

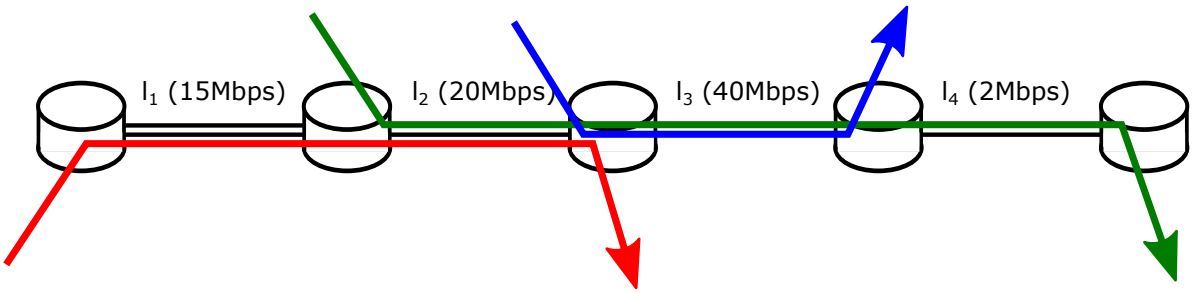


Figure 1: A simple network scenario

There are 3 flows (shown in red, blue and green) that share different bottleneck links in the network. Assume that the flows are all long running TCP flows and have reached a steady state in terms of the sending rate.

The switches in the network can choose to adopt one of the two types of rate allocations:

1. **Fair Rate:** With this rate allocation, each switch independently determines the number of flows sharing the bottleneck link and limits each flow to a rate no more than C/N, where C is the capacity of the bottleneck link and N is the number of flows sharing the link.
2. **Max-Min Fair:** With this rate allocation, the switches coordinate with each other and perform rate allocation so as to maximize the minimum rate achievable by each flow in the network.

Given this information, answer the following questions on Coursemology:

Q1 When the switches are doing fair rate allocation, what is the final (steady-state) throughput achieved by the Red, Blue and Green flows respectively?

Q2 Which links in the network are underutilized?

Next, suppose that the switches can coordinate with each other and do max-min fair allocation.

Q3 In this new scenario, what is the final (steady-state) throughput achieved by the Red, Blue and Green flows respectively?

Q4 In this new scenario, which links in the network are underutilized?

Fair Rate: Fixed vs. Instantaneous

In this section, we investigate the two ways of doing fair rate allocation and their implication on end-points with continuous and intermittent data flow. In order to do fair rate allocation for a link, the switch first needs to know the number of flows sharing the link. This estimation of the number of flows can be done in the following ways:

- **Fixed:** The switches count the number of flows based on the initial TCP handshake. So if a TCP flow establishes a connection through the 3-way handshake, the switch increases the flow count N by 1 and allocates a fair rate of C/N to all the flows, even if some of the TCP flows are not actively sending any data.
- **Instantaneous:** The switches count the number of flows based on which flows are actively sending data at the given moment. For example, if 3 flows have established a TCP connection, but at the current moment, only 2 flows are sending their data, then the allocated fair rate is $C/2$. After some time, if the third flow also starts sending data, then the rate allocation is changed to $C/3$ instantaneously.

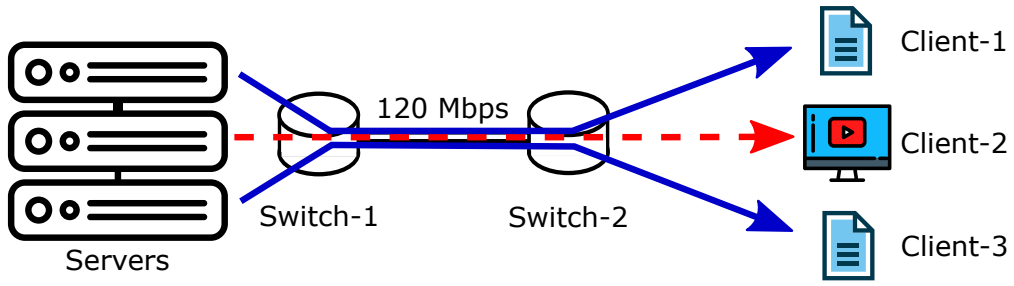


Figure 2: Network scenario with a DASH client

Now consider the scenario in Figure 2. Three clients are downloading data from servers while sharing a bottleneck of 120 Mbps. Switch-1 performs fair rate allocation – either fixed or instantaneous. Client-1 and client-3 are long running file download clients, each of them downloading a very huge file. Therefore, client-1 and client-3 have a continuous data flow. Client-2 is a browser-based video player which is streaming a long running live video stream using DASH (Dynamic Adaptive Streaming over HTTP). Client-2 has an intermittent data flow as explained below.

DASH Streaming: In DASH streaming, the video is divided into small segment files which the client downloads over HTTP. In this scenario, each segment is 5 secs worth of video content and the size of each segment is constant and fixed at 5 MB. After the TCP handshake is done and the HTTP connection is established over TCP, the video player (aka DASH client) downloads the first video segment (Seg-0). During this first download, the DASH client records the time[1] (say δ) it took to download the segment. After this download is complete, the DASH client will start playing the first segment. We denote this point of starting the video playback as $t = 0$ and the DASH client (client-2) would be

[1]Note that irrespective of fixed or instantaneous fair rate allocation, the time δ to download a video segment would remain the same for each segment.

in a steady state[2] here onward. Since each segment is 5 seconds long, at $t = 5$, the DASH client would need to play the next segment. Since it takes δ time to download a segment, the DASH client will start downloading the next segment at $t = 5 - \delta$. This behavior of the DASH client will create an intermittent data flow on the network as shown in Figure 3.

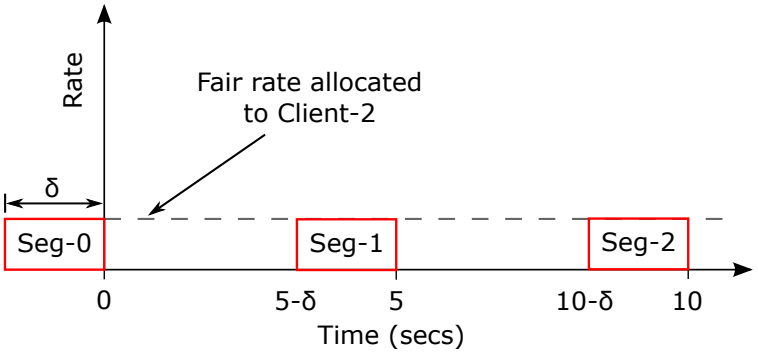


Figure 3: DASH client's data flow

Given this information, answer the following questions on Coursemology:

Q5 Suppose switch-1 is doing fixed fair rate allocation. In this case, what is the average utilization of the link between switch-1 and switch-2?

Q6 What is the average throughput observed by each of the three clients?

Now suppose that the switch-1 is doing instantaneous fair rate allocation.

Q7 For this new scenario, what is the average utilization of the link between switch-1 and switch-2?

Q8 What is the average throughput observed by each of the three clients?

Solutions

Q1 First we determine the maximum fair rates for the various links as following:

Link	**Capacity (Mbps)**	**Flows**	**Max fair rate (Mbps)**
l_1	15	1	15
l_2	20	2	10
l_3	40	2	20
l_4	2	1	2

Then the fair rate for each flow would be the minimum of all the fair rates provided by the links on the flow's path:

Flow	**Allocation (Mbps)**
Red	$\min(15, 10) = 10$
Green	$\min(10, 20, 2) = 2$
Blue	20

Therefore, the final throughput achieved by the Red, Blue, and Green and flows are **10 Mbps, 20 Mbps, and 2 Mbps**, respectively.

[2] steady state in terms of throughput/rate when averaged over a sufficient amount of time

Q2 Underutilized links are those that run below their capacity. Given the above rates for the Red, Blue and Green flows, the total traffic for each link is as following:

Link	**Capacity (Mbps)**	**Flows**	**Traffic (Mbps)**
l_1	15	1	10
l_2	20	2	10+2 = 12
l_3	40	2	2+20 = 22
l_4	2	1	2

Therefore, the links **l_1, l_2 and l_3** are underutilized.

Q3 Recall that for max-min fair allocation, we want to maximize the minimum rate for each flow. Therefore, it is easier if we start with a link that has the minimum capacity (l_4 in this case). This is because the minimum capacity link will basically limit the maximum possible rate for flows passing through it. Only the Green flow goes through l_4. Therefore, let's allocate 2 Mbps to the Green flow. Now this information is shared with other switches in the network. As a result, we know that on the links l_2 and l_3, the Green flow will consume only 2 Mbps. The remaining capacity on these links could then be allocated to the other flows. Therefore, on l_3, we can allocate the remaining 38 Mbps to the Blue flow. Similarly, on l_2, we can allocate the remaining 18 Mbps to the Red flow. Note that the Red flow also flows through l_1. However, since l_1 has a capacity of 15 Mbps, the Red flow would be bottlenecked at l_1, and it won't be able to utilize its entire share of 18 Mbps on l_2.

In summary, after the max-min rate allocations, the final throughput achieved by the Red, Blue and Green flows is **15 Mbps, 38 Mbps and 2 Mbps**, respectively.

Q4 Similar to Q2, we can now compute the link utilizations as following:

Link	**Capacity (Mbps)**	**Flows**	**Traffic (Mbps)**
l_1	15	1	15
l_2	20	2	15+2 = 17
l_3	40	2	38+2 = 40
l_4	2	1	2

Therefore, only **l_2** is underutilized.

Calculating δ

We first need to calculate δ in order to answer any of the remaining questions. In Figure 2, whether switch-1 does fixed or instantaneous rate allocation, the DASH client (client-2), will get its fair share of the bandwidth each time it downloads a segment. With 3 flows sharing a 120 Mbps bandwidth, the fair share is 120/3 = 40 Mbps. Now δ is the time it takes to download a 5 MB segment with 40 Mbps bandwidth:

$$\delta = \frac{5 \times 8 \ Megabits}{40 \ Megabits/sec} = 1 \ second \tag{1}$$

Q5 Since switch-1 is doing fixed rate allocation for the three flows, it will allocate fixed 40 Mbps for each flow. We can re-draw Figure 3 by using the value of δ and visualize the combined data flow on the bottleneck link in Figure 4. For analysis of the average link utilization, let's consider the first 10 seconds of the steady state, since the pattern would repeat thereafter.

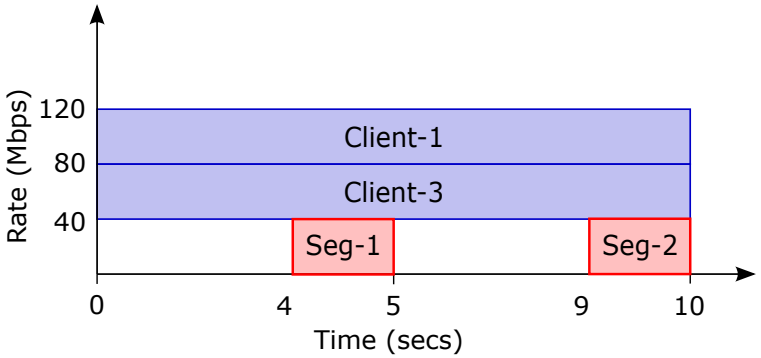


Figure 4: Data flow on the bottleneck with fixed fair rate allocation. Figure NOT drawn to scale.

As we see from the shaded areas in Figure 4, out of the 10 seconds, link utilization is 120 Mbps for 2 seconds and 80 Mbps for the remaining 8 seconds. Therefore, average utilization is

$$Average\ utilization = \frac{2}{10} \times 120 + \frac{8}{10} \times 80 = 24 + 64 = \textbf{88 Mbps} \tag{2}$$

Q6 From Figure 4, we see that the average throughput for client-1 and client-3 is constant at 40 Mbps. Client-2 it gets the average 40 Mbps bandwidth for 2 seconds and for rest of the 8 seconds gets no bandwidth. Therefore, the average throughput observed by client is given by

$$Average\ throughput\ (client\ 2) = \frac{2}{10} \times 40 + \frac{8}{10} \times 0 = 8\ Mbps \tag{3}$$

In summary, the average throughput observed by the client-1, client-2 and client-3 is **40 Mbps, 8 Mbps and 40 Mbps** respectively.

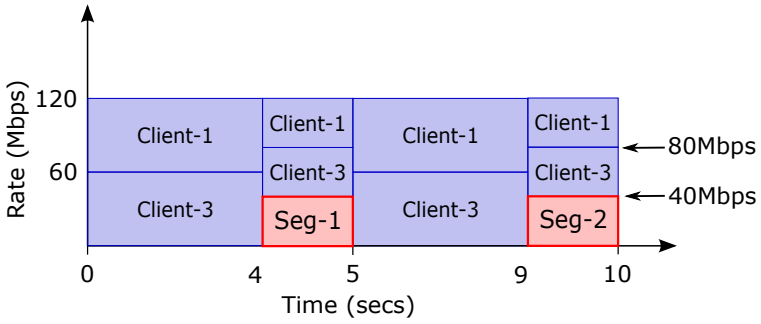


Figure 5: Data flow on the bottleneck with instantaneous fair rate allocation. Figure NOT drawn to scale.

Q7 When switch-1 is doing instantaneous fair rate allocation, client-1 and client-3 will share the 120 Mbps equally among themselves (60 Mbps each) when the DASH client (client-2) is not downloading any segment. Whenever the client-2 is downloading a DASH video segment, the allocation will change instanteneously such that all three flows get 40 Mbps each. We can visualize this combined data flow on the bottleneck link in Figure 5. We see that at all times the link is fully utilized. Therefore, the average link utilization is **120 Mbps**.

Q8 For average per-flow throughput, we can again perform an analysis for the first 10 seconds of the steady state. From Figure 5, we see that client-1 and client-3 get 60 Mbps each for 8 seconds and 40 Mbps each for 2 seconds. Therefore, the average throughput for client-1 and client-3 is (8/10) x 60 + (2/10) x 40 = 48 + 8 = 56 Mbps. For client-2, similar analysis as done in equation 3 would give us an average throughput of 8 Mbps. In summary, the average per-flow throughput observed by client-1, client-2 and client-3 is **56 Mbps, 8 Mbps and 56 Mbps** respectively.

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 for datacenter networks which helps keeps 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.

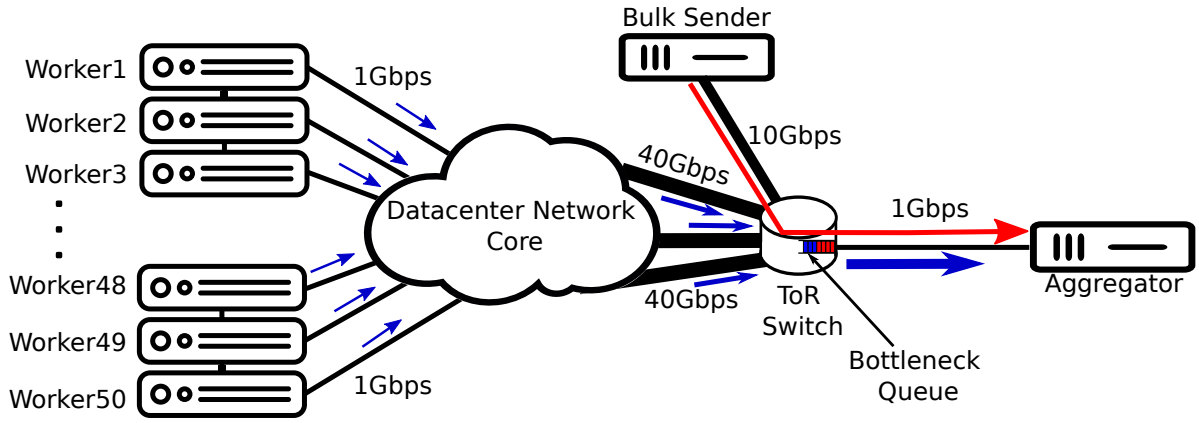
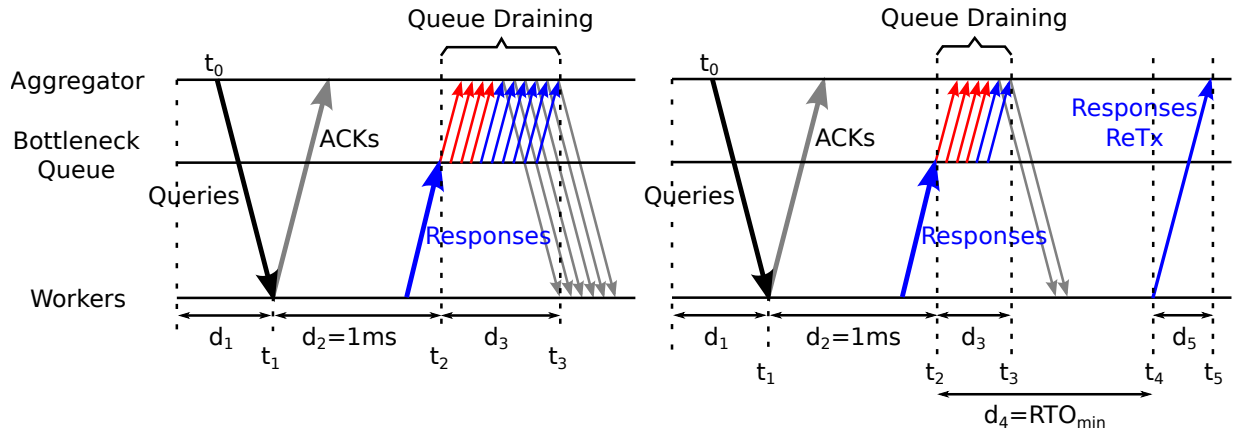


Figure 1: Incast in a datacenter network

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 moving on with a low quality 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). (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 linkspeed. 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. 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 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

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

worker who 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. Therefore, the worker waits for the retransmission timeout (RTO_{min}), before retransmitting the response packet(s) for which no ACK was received. Therefore, in this case, the query would complete only at t_5. In our setup, the value of RTO_{min} is **300 ms**[2]. 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). 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)?

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

[2]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 adjusted accordingly.

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.

Solutions

The first step in calculating QCT in any scenario is to determine whether or not any Incast loss will occur in that scenario. Then we can choose an appropriate subfigure from Figure 2 and calculate the QCT accordingly.

Incast loss or not

To determine if Incast loss will occur or not, we first need to know the total size of the response that would arrive in the bottleneck at t_2. Since response from each worker is 3 KB and there are 50 workers, the total response size is 3 x 50 = **150 KB**. Since the total response size is 150 KB, there would be Incast loss if the headroom available in the bottleneck queue is **less than 150 KB**.

Common delays

Some delays are common to all the scenarios. So let's compute them first.

$$d_1 = d_5 = \frac{RTT}{2} = \frac{120}{2} = 60\,\mu s \tag{7}$$

Scenarios

Q1 In scenario (A), the bottleneck queue size is 145 KB which is less than 150 KB required to absorb the Incast. Therefore, in this case Incast loss will occur. The QCT in this case is given by Equation 2:

$$\begin{aligned} QCT &= d_1 + d_2 + d_4 + d_5 \\ &= 60\,\mu s + 1\,ms + 300\,ms + 60\,\mu s \\ &= \mathbf{301.120\,ms} \end{aligned}$$

Q2 In scenario (B), the bottleneck queue size is 200 KB which is more than 150 KB required to absorb the Incast. Therefore, in this case, Incast loss will not occur. The QCT in this case is given by Equation 1. When Incast occurs, the queue will instantaneously fill to 150 KB. Therefore, d_3 is the time required for 150 KB to drain from the queue at 1 Gbps speed, **TODO: plus the latency between the bottleneck queue and the aggregator**:

$$\begin{aligned} d_3 &= \frac{150 \times 1000 \times 8\ bits}{1\ Gbps} \\ &= 1.2\,ms \end{aligned}$$

Now, we can use Equation 1 to calculate QCT:

$$\begin{aligned} QCT &= d_1 + d_2 + d_3 \\ &= 60\,\mu s + 1\,ms + 1.2\,ms \\ &= \mathbf{2.260\,ms} \end{aligned}$$

Q3 In scenario (C), it is not very straightforward to figure out whether any Incast loss will occur or not. This is because in steady state, TCP Reno's queue occupancy keeps changing in a saw tooth pattern as we saw in Lecture 3 training. So let us first figure out the two extremities of TCP Reno's queue occupancy.

$$\begin{aligned} cwnd_{max} &= BDP + queueSize \\ &= (1\,Gbps \times 120\,\mu s) + 200\,KB \\ &= 15\,KB + 200\,KB \\ &= 215\,KB \\ cwnd_{min} &= \frac{cwnd_{max}}{2} \\ &= \frac{215}{2} \\ &= 107.5\,KB \\ \text{Now, } q_{max} &= queueSize = 200\,KB \\ \text{and, } q_{min} &= cwnd_{min} - BDP \\ &= 107.5\,KB - 15\,KB \\ &= 92.5\,KB \end{aligned}$$

Clearly, when Reno is occupying the whole queue, there is no room to absorb the Incast. So there would be Incast losses. However, when Reno's queue occupancy is at a minimum of 92.5 KB, the remaining 107.5 KB (200 - 92.5) headroom in the

queue is still less than 150 KB required to absorb the Incast. Therefore, irrespective of Reno's steady-state queue size, Incast loss will always occur in scenario (C). The QCT is therefore given by Equation 2:

$$\begin{aligned} QCT &= d_1 + d_2 + d_4 + d_5 \\ &= 60\,\mu s + 1\,ms + 300\,ms + 60\,\mu s \\ &= \mathbf{301.120\,ms} \end{aligned}$$

Q4 Similar to scenario (C), we will first compute q_{max} and q_{min} for the DCTCP flow. Equation 10 from the DCTCP paper, gives the q_{max} as: $q_{max} = K + N$ where, N is the number of DCTCP flows. Therefore,

$$\begin{aligned} q_{max} &= K + N \\ &= 20 + 1 = 21\ packets = 31.5\,KB \end{aligned}$$

Also, using equations 8 and 11 from the DCTCP paper, we have

$$\begin{aligned} q_{min} &= q_{max} - A \\ &= 21 - \frac{1}{2}\sqrt{2N(C \times RTT + K)} \\ &= 21 - \frac{1}{2}\sqrt{2(10+20)} && \text{(using value of BDP in packets)} \\ &= 21 - 4 && \text{(3.87 rounded off to 4 packets)} \\ &= 17\ packets = 25.5\,KB \end{aligned}$$

Therefore, when DCTCP flow's queue occupancy varies between 25.5 KB and 31.5 KB in the steady state, the available headroom in the queue varies between 174.5 KB and 168.5 KB. This means that at all times, the queue can absorb an Incast of 150 KB and there would be no Incast losses. In this scenario, the QCT will therefore be given by equation 1. This QCT will be varying since the queue draining delay d_3 would be varying depending on the queue build up after Incast.

$$\begin{aligned} {d_3}^{min} &= \frac{q_{min} + IncastSize}{LinkSpeed} \\ &= \frac{(25.5 + 150)\,KB}{1\,Gbps} = 1.404\,ms \end{aligned}$$

Similarly,

$$\begin{aligned} {d_3}^{max} &= \frac{q_{max} + IncastSize}{LinkSpeed} \\ &= \frac{(31.5 + 150)\,KB}{1\,Gbps} = 1.452\,ms \end{aligned}$$

Therefore, using equation 1, we have

$$\begin{aligned} QCT^{min} &= d_1 + d_2 + {d_3}^{min} \\ &= 60\,\mu s + 1\,ms + 1.404\,ms \\ &= \mathbf{2.464\,ms} \end{aligned}$$

Similarly,

$$\begin{aligned} QCT^{max} &= d_1 + d_2 + {d_3}^{max} \\ &= 60\,\mu s + 1\,ms + 1.452\,ms \\ &= \mathbf{2.512\,ms} \end{aligned}$$

Therefore, in scenario (D), the QCT will vary between **2.464 ms** and **2.512 ms**. If we compare these values to scenario (B)'s QCT of 2.260 ms, we see that the addition of a long running DCTCP flow in scenario (D) adds less than 300 µs of additional QCT.

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

Lecture 8 Training
Mobile Ad Hoc Networks

In Lecture 8, we discussed various routing algorithms for mobile ad hoc networks where nodes communicate on wireless channels without centralized infrastructure and administration. We also discussed how location information can be exploited to achieve efficient routing using *geographic routing* algorithms. In this training, we will explore geographic routing in a modern context.

After taking CS5229 and graduating from NUS, you found yourself working for SpaceX under their Starlink project.

> "Starlink is a satellite internet constellation operated by SpaceX providing satellite Internet access to most of the Earth. The constellation consists of over 1600 satellites in mid-2021, and will eventually consist of many thousands of mass-produced small satellites in low Earth orbit (LEO), which communicate with designated ground transceivers. While the technical possibility of satellite internet service covers most of the global population, actual service can be delivered only in countries that have licensed SpaceX to provide service within any specific national jurisdiction. As of September 2021, the beta service offering is available in 17 countries." – Wikipedia

Understanding Local Minima

Your new boss asked you to evaluate the a proposal for a constellation of geo-stationary satellites[1] covering x longitudes and y latitudes. Basically, this constellation consists of xy satellites placed at the intersections of the longitudes and latitudes. You can assume that the intervals between the satellites are equal and uniform. You can assume that the satellites are equipped with radios that allow them to communicate with adjacent satellites along each latitude and also longitude (except the highest and lowest ones along each longitude, Figure 1). Basically, assume radio links do not extend across the north and south poles.

Given this information, which of the following statements is/are true:

Q1 Greedy forwarding between the satellites based on shortest distance along the circumference of the earth will be correct as long as there are no node failures.

Q2 Greedy forwarding between the satellites based on shortest distance along the circumference of the earth will be correct as long as there is at most one node failure.

Q3 Greedy forwarding between the satellites based on direct Cartesian distance will be correct as there are no node failures.

[1] Starlink uses Low-Earth Orbit (LEO) satellites instead of Geosynchronous Equatorial Orbit (GEO) satellites.

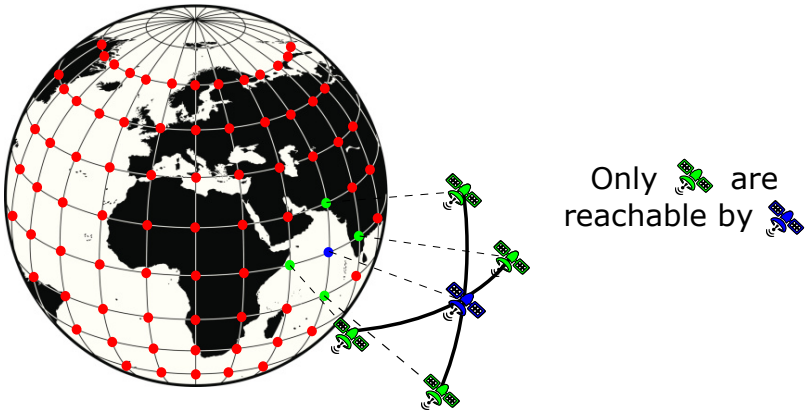


Figure 1: Organization of the geostationar satellites along the lattitudes and longitudes.

Recovering from Node Failures

Suppose greedy forwarding is adopted for the constellation of geo-stationary satellites and shortest distance along the circumference of the earth is used as the distance metric.

Q1 Propose what we should do if we experience one node failure, if we decide to stick with geographic routing.

Q2 Can we adopt the same strategy if there are more than one node failures? Explain. If not, proposal how we might be able to recover from the failure if we decide to stick with geographic routing.

GDSTR without Conflict Hulls

A key skill is to be able to analyze an algorithm and reason about how it works. We discussed GDSTR in lecture and mentioned briefly that *conflict hulls* will allow us to determine whether a packet is undeliverable more efficiently. It is quite difficult to explain this clearly in a lecture. In this question, your task is to reason about routing on a hull tree without conflict hulls. In particular, consider the scenario where you are running GDSTR with a single hull tree and the destination is invalid. In other words, none of the nodes in the network have a location that matches what is specified as the address. Explain how we would conclude that the packet is undeliverable.

Solutions

Understanding Local Minima

Q1 Greedy forwarding between the satellites based on shortest distance along the circumference of the earth will be correct as long as there are no node failures.

True. All faces are either rectangular or an x-gon. Since all faces are convex, greedy forwarding will always work.

Q2 Greedy forwarding between the satellites based on shortest distance along the circumference of the earth will be correct as long as there is at most one node failure.

False. The failure of a node at one of the top or bottom latitudes will cause one of the x-gon faces to become concave. A concave face will have a local minimum.

Q3 Greedy forwarding between the satellites based on direct Cartesian distance will be correct as there are no node failures.

True. To see that this statement is true, we construct bisecting planes for the various links. The volume containing each node cannot contain any other node, so it is not possible to have a local minimum when doing greedy forwarding based on this new metric.

Recovering from Node Failures

Q1 Propose what we should do if we experience one node failure, if we decide to stick with geographic routing.

Nothing needs to be done. All the faces will be convex, so greedy forwarding will continue to work.

Q2 Can we adopt the same strategy if there are more than one node failures? Explain. If not, proposal how we might be able to recover from the failure if we decide to stick with geographic routing.

It depends. There are only xy nodes and we know exactly where all the nodes are relative to each other. Unlike traditional geographic routing algorithms, we have no need to compute the state required to route around the local minima in a distributed way. The key idea is to compute the bisectors of the links adjacent to each node. A node is a local minimum if and only if some other nodes fall within the area nearer to it than it's neighbours. Given the layout around a global, it is quite unlikely. Once we compute all the nodes that can be possible local minimum, we also compute the required forwarding entries required to allow these local minimum nodes for the destination nodes that can cause local minima.

GDSTR without Conflict Hulls

Consider the scenario where you are running GDSTR with a single hull tree and the destination is invalid. In other words, none of the nodes in the network have a location that matches what is specified as the address. Explain how we would conclude that the packet is undeliverable.

Given that the destination is unreachable, greedy forwarding will definitely end up in a local minimum. This means that GDSTR will switch to *tree traversal*. Given that there is only one hull tree, there is no need to select a tree. There is no choice.

Now that we are at some (intermediate) node in the hull tree, there are 2 cases: (i) one or more of the children have convex hulls that contain the destination node; and (ii) none of the children have convex hulls containing the destination.

For case (i), the natural thing to do is to traverse the children in some fixed order. Since the destination is unreachable, the packet will eventually return to the current intermediate node.

Can we then conclude that the packet is unreachable? No, we cannot, because it is plausible that some other branch of the tree accessible higher up in the tree might have

a hull containing the destinations. Hence, we will eventually have to route the packet up to the parent. By this logic, we can only conclude that a packet is unreachable after the packet is routed to the root of the hull tree (and possibly completing the traversals of all child nodes with hulls containing the destination.

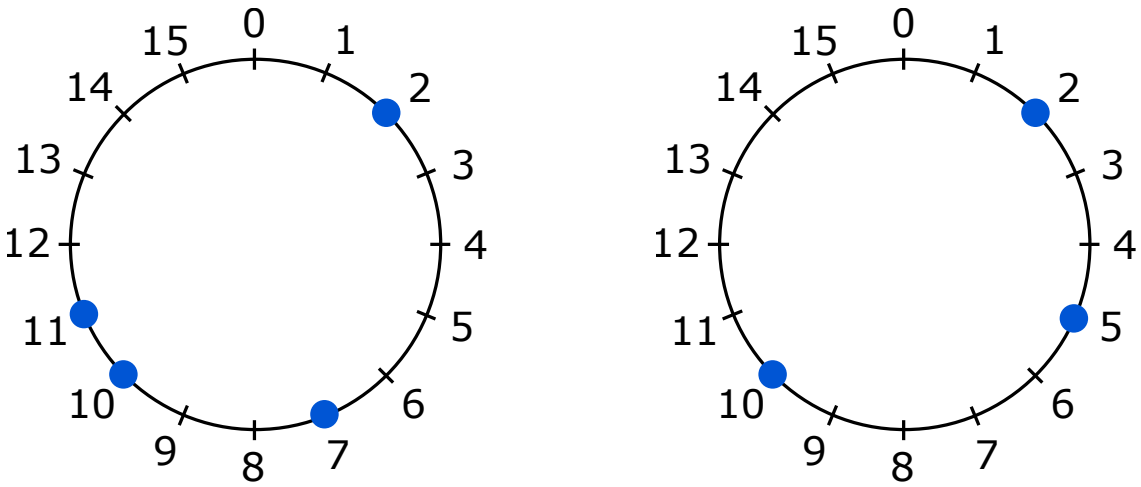
Once this is clear, one can reasonable about how conflict hulls work. Basically conflict hulls provide information on whether a destination could be contained in some part of the network that is reachable from the parent.

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

Lecture 9 Training
Peer-to-Peer Networks

In Lecture 9, we discussed how a distributed lookup service could be implemented using Distributed Hash Tables (DHTs) and the Chord routing protocol. In this lecture training, we will reinforce some of the concepts learnt in the lecture.

Finger Table Computation



(a) Identifier Circle for service A

(b) Identifier Circle for service B

Figure 1: Chord Identifier Circles for different distributed lookup services.

Figure 1 shows two identifier circles for two different distributed lookup services (A) and (B). Each identifier circle has 16 unique IDs each. Each distributed lookup service uses consistent hashing with identifier length $m = 4$ to represent the 16 IDs.

Different number of nodes (shown in blue) occupy different ID positions in the two services (Figure 1a and Figure 1b).

Given this information, answer the following question on Coursemology:

Q1 Compute the finger table for node 7 in service (A). Each option represents a row in the finger table in the format [start, interval, successor].

Performing Lookup

For the lookup service (B), we have already provided the finger tables. Figure 2 shows the finger tables for nodes 2, 5, and 10 in lookup service (B).

start	int	succ
3	[3,4)	5
4	[4,6)	5
6	[6,10)	10
10	[10,2)	10

(a) Finger Table for node 2

start	int	succ
6	[6,7)	10
7	[7,9)	10
9	[9,13)	10
13	[13,5)	2

(b) Finger Table for node 5

start	int	succ
11	[11,12)	2
12	[12,14)	2
14	[14,2)	2
2	[2,10)	2

(c) Finger Table for node 10

Figure 2: Finger Tables for nodes in lookup service B.

Now suppose that a client joins service B and wishes to perform lookup for objects with specific IDs. The client performs an iterative lookup i.e. it queries for a specific object ID to one of the nodes. If the node does not have the object with the requested ID, it replies back with the ID of the successor based on its finger table. The client will then query the successor and so on until one of the node returns the object with the requested ID.

Given this information, answer the following questions on Coursemology:

Q2 If the client queries node 10 for an object with ID 6, compute the sequence in which nodes in service B are queried.

Q3 If the client queries node 10 for an object with ID 4, compute the sequence in which nodes in service B are queried.

Iterative vs. Recursive routing

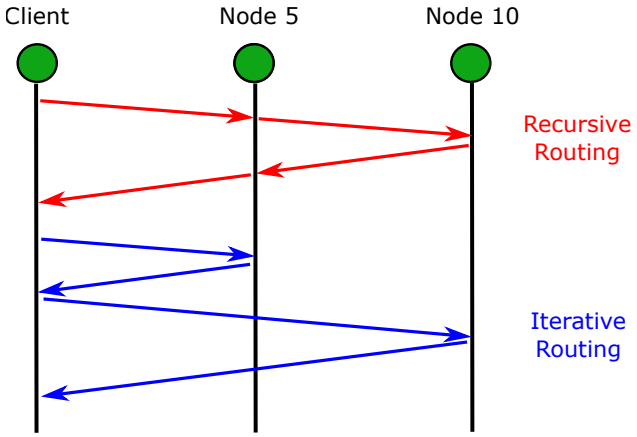


Figure 3: recursive vs. Iterative routing

Consider a client in service B (Figure 1b) who queries node 5 for the object with ID 9 and ID 12 respectively. Based on the finger tables already provided in Figure 2, we know that for retrieving the object with ID 9, the client will have to query **Node 5** and then **Node 10**, and for retrieving the object with ID 12, it will have to query **Node 5**, then **Node 10**, and then finally **Node 2**. It can perform both these queries either iteratively or recursively. Figure 3 shows how these two query types differ while searching for the object with ID 9, as an example. Your task is to figure out which routing scheme gives better response time for both of the queries. Note that the identifier circle is only a logical representation of the nodes. In practice, any two nodes could be arbitrarily apart on the

Internet. Therefore, to compute the actual response time for a lookup operation, we need to take into account the inter-node latencies. For this lecture training, you should use the latencies listed in Figure 4. Assume that the processing time at each node is negligible and therefore can be ignored. Also, note that the latencies shown in Figure 4 are one-way delays between the two nodes in either direction.

Pair of nodes	latency
Client $\longleftrightarrow$ Node 2	15 ms
Client $\longleftrightarrow$ Node 5	5 ms
Client $\longleftrightarrow$ Node 10	17 ms

Pair of nodes	latency
Node 2 $\longleftrightarrow$ Node 5	20 ms
Node 2 $\longleftrightarrow$ Node 10	50 ms
Node 5 $\longleftrightarrow$ Node 10	10 ms

Figure 4: Latencies between the various nodes for service B.

Now, answer the following questions on Coursemology:

Q4 What is the response time when iterative and recursive routing are used to retrieve object ID 9? Which is better?

Q5 What is the response time when iterative and recursive routing are used to retrieve object ID 12? Which is better?

Solutions

Q1 The first step in calculating a finger table for a node is to determine the start points. The start points are given by $nodeID + 2^n$, where n varies from 0 to $m - 1$. For this question, the node ID is 7 and $m = 4$. Therefore, the start points are (7+1), (7+2), (7+4), (7+8) i.e. 8, 9, 11, 15. The next step is to determine the intervals. The intervals are basically ranges between the start points such that the first start point is inclusive in the interval while the subsequent start point is not. The last interval ends with node ID (exclusive). Therefore, the intervals are [8,9), [9,11), [11,15), [15,7). The final step is to assign successors to each interval. A successor for an interval is the first node on the identifier circle that falls in that interval. If there is no node on the identifier circle within that interval, then successor for that interval is the first node that appears after the interval. So this way, the successors for the intervals are 10, 10, 11, and 2 respectively. Overall, the finger table for node 7 in lookup service A is shown in Table 1 below:

start	int	succ
8	[8,9)	10
9	[9,11)	10
11	[11,15)	11
15	[15,7)	2

Table 1: Finger Table for node 7 in service (A)

Q2 We first need to compute which nodes have which objects. Every node stores objects between the previous node and itself in the clockwise direction. Therefore, in lookup service (B) nodes store the following objects:

Node 2 $\leftarrow$ (10,2]
Node 5 $\leftarrow$ (2,5]
Node 10 $\leftarrow$ (5,10]
Since the client has queried node 10 for the object with ID 6, which node 10 already has, node 10 will respond back with object ID 6 to the client. Therefore, the sequence in which nodes in service B are queried is: **Node 10**.

Q3 When node 10 receives a query for object ID 4, it first checks if it posseses the required object ID. Based on the aforementioned analysis, node 10 only has (5,10]. Therefore, node 10 checks its finger table and finds that 4 falls within the interval [2,10) for which the successor is node 2. Therefore, it redirects the client to node 2. The client then queries node 2. Node 2 again does not have the object with ID 4 and so it checkes its finger table. For node 2, 4 falls within the interval [4,6) for which the successor is node 5. So node 2 redirects the client to node 5. The client then queries node 5. This time, node indeed has the object with ID 4 and returns it to the client as the query response. Therefore, the sequence in which nodes in service B are queried is: **Node 10, Node 2, Node 5**.

Q4 Recurive routing to retrieve ID 9:

$$c \rightarrow 5 \rightarrow 10 \rightarrow 5 \rightarrow c$$

Response time: **30 ms**
Iterative routing to retrieve ID 9:

$$c \rightarrow 5 \rightarrow c \rightarrow 10 \rightarrow c$$

Response time: **44 ms**
Recursive routing is better.

Q5 Recursive routing to retrieve ID 12:

$$c \rightarrow 5 \rightarrow 10 \rightarrow 2 \rightarrow 10 \rightarrow 5 \rightarrow c$$

Response time = **130 ms**
Iterative routing to retrieve ID 12:

$$c \rightarrow 5 \rightarrow c \rightarrow 10 \rightarrow c \rightarrow 2 \rightarrow c$$

Response time: **74 ms**
Iterative routing is better.

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

Lecture 10 Training
Introduction to Distributed Systems

In Lecture 10 we talked about Quorum-based consistency, a simple but effective way to maintain consistency between replicas in a distributed system. To recap, a quorum system maintains n replicas and waits for w successful writes before considering a write operation to be complete and r successful reads before considering a read operation to be complete. The Quorum consistency model applies the following constraints on these variables:

$$w + r > n \tag{1}$$

This ensures that during every round of read and write, there is **at least one** replica that has seen both the read and write operations, and is therefore consistent. Note that Equation 1 does not put any constraint on the individual values for w and r, just that their sum should be more than n. Now consider a network with 15 nodes in the Quorum group, with the nodes having different network latencies (Figure 1)

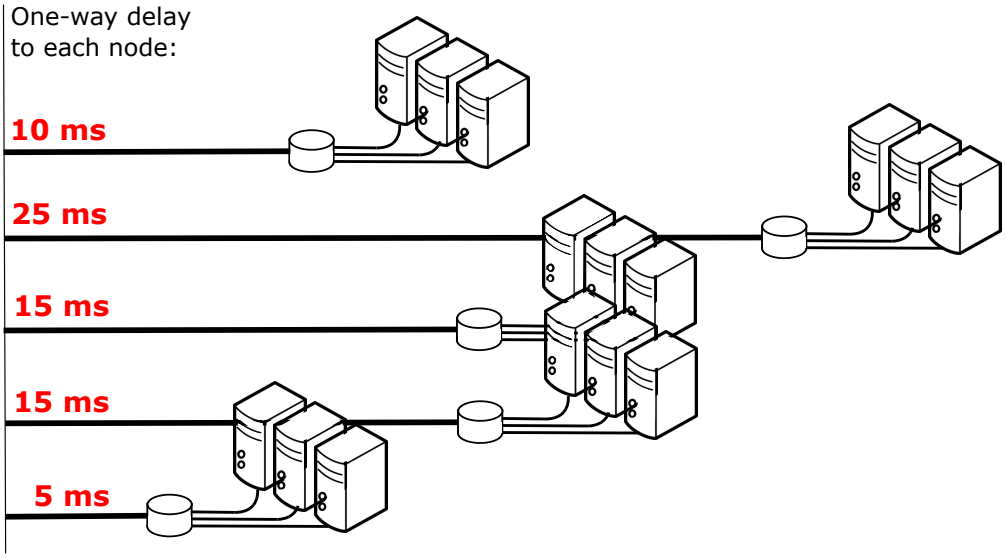


Figure 1: One-way delay to all the nodes in the Quorum group

Given this network (and assuming no failures), answer the following questions on Coursemology:

Q1 What is the response time for **write requests** when $w = 8$ and $r = 8$?

Q2 What is the response time for **read requests** when $w = 8$ and $r = 8$?

Q3 Let's say all the requests in this distributed system are packaged as *read-write* requests, i.e a read request followed by a write request. You can assume that

the write operation is only started when the quorum system considers the read operation to be complete. What will be the response time for a *read-write* request when:

(a) $w = 7$ and $r = 9$

(b) $w = 8$ and $r = 8$

(c) $w = 4$ and $r = 14$

Q4 **[EXTRA[1]]** Is there a quorum allocation that can yield a better response time for *read-write* requests? If yes, what is the improved response time under this allocation?

Q5 **[EXTRA]** Assume that this distributed system relies on the Quorum's pigeonhole principle-based consistency ONLY. Is the constraint described in Equation 1 enough to guarantee consistency? You can assume there are fixed quorum groups and no failures.

Solutions

A1 30 ms

A2 30 ms

A3 For each quorum size, the *read-write* response time is:

(a) 60 ms (30 + 30)

(b) 60 ms (30 + 30)

(c) 70 ms (20 + 50)

A4 **[EXTRA]** Yes, there is. $w = 4$ and $r = 12$, for example, gives a response time of 20+30 = 50 ms.

A5 **[EXTRA]** No, it is not sufficient. Consider $w = 1$ and $r = 15$. In this case, it is possible for subsequent writes to be written to two different servers while satisfying Equation 1.

[1] EXTRA questions carry no marks.