CS 144 Midterm Review

Feb 7, 2025

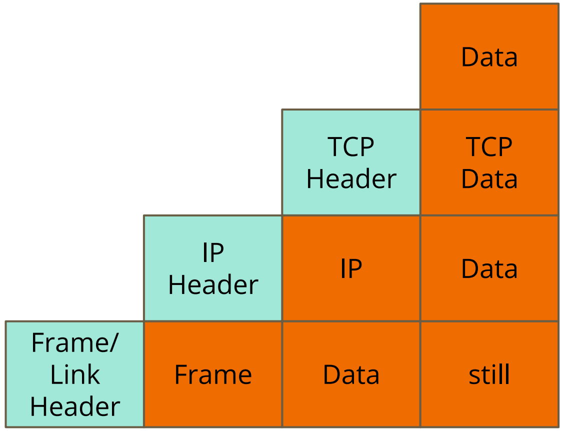
TCP - Overview

TCP is a "reliable" service on top of an "unreliable" service abstraction

Why isn't "best-effort" delivery enough?

- Delivered successfully
- Not delivered/packet loss
- Duplicate delivery
- Corruption/modified in transit
- Out of order delivery

TCP is built on top of IP which is built on top of the link layer (something like Ethernet)



TCP - Reliability Mechanisms

Sender: Sequence number (seqno) ensures order is maintained. Allows for commutative delivery - meaning the order of delivery doesn’t affect the final output. This also helps detect lost data (gaps in seqnos).

Receiver: Acknowledgement number (ackno) confirms receipt of all prior data and helps a sender decide when to retransmit messages

All messages are **idempotent**! multiple transmissions of the same data produce the same result! retransmissions do not create duplicate messages or inconsistencies.

TCP - Beginning and Ending Connections

To begin a connection, four things must happen.

- Peer A sends SYN, Peer B acknowledges this SYN, Peer B sends SYN, and Peer A acknowledges it.
- This is known as a three-way handshake, since it can be done in a minimum of three messages. (SYN, SYN-ACK, ACK)

To end a connection, four things must happen.

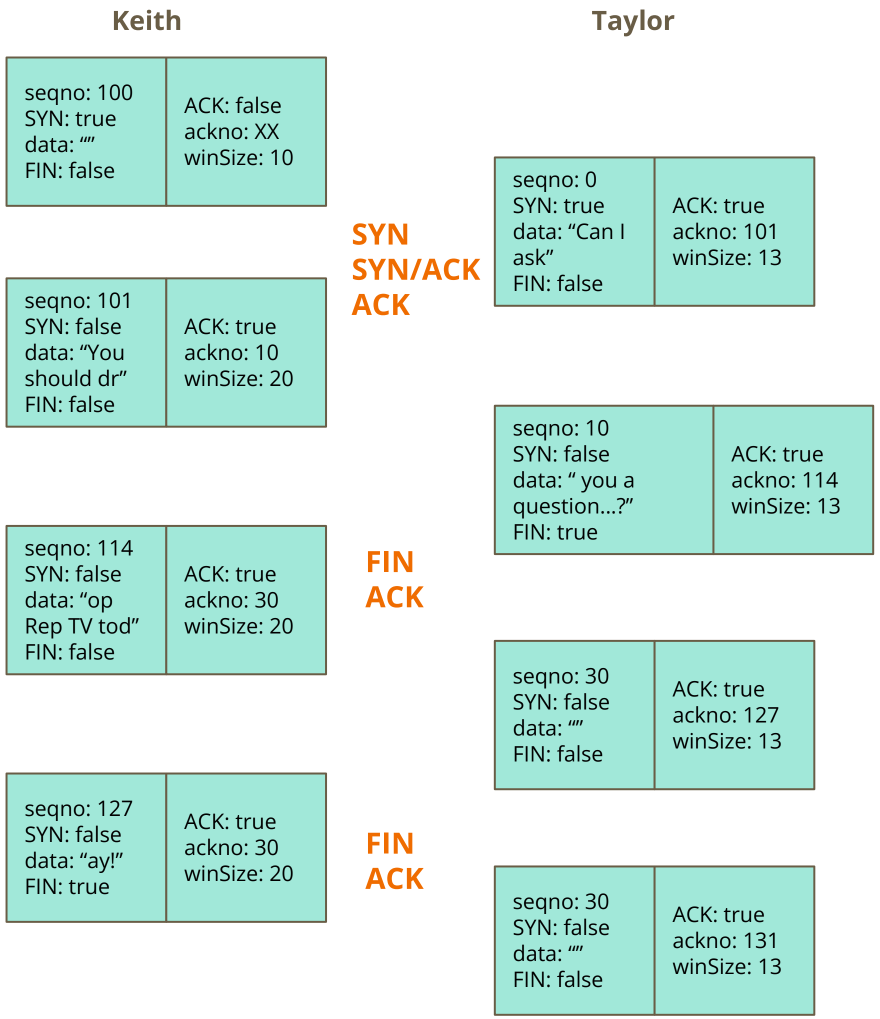
- Peer A decides to end connection and sends FIN. Peer B acknowledges this FIN. When Peer B is done, it sends a FIN. Peer A acknowledges this FIN.
- Peer A must stay awake longer in case Peer B didn’t get A’s acknowledgement.
- This handshake cannot always be completed in three messages, as it isn’t guaranteed that Peer B is done with Peer A sends a FIN.

TCP - Worked Example

Keith wants to send this message to Taylor Swift: "You should drop Rep TV today!"

Simultaneously, Taylor Swift happens to be curious about TCP and knows that Keith is the best professor to ask. She wants to send this message: "Can I ask you a question...?" and starts sending as soon as she receives a SYN from Keith.

Both of these should be sent in as few segments of possible. Keith will assume Taylor has an initial window size of 1.



Keith

seqno: 100 SYN: true data: "" FIN: false	ACK: false ackno: XX winSize: 10
seqno: 101 SYN: false data: "You should dr" FIN: false	ACK: true ackno: 10 winSize: 20
seqno: 114 SYN: false data: "op Rep TV tod" FIN: false	ACK: true ackno: 30 winSize: 20
seqno: 127 SYN: false data: "ay!" FIN: true	ACK: true ackno: 30 winSize: 20

Taylor

seqno: 0 SYN: true data: "Can I ask" FIN: false	ACK: true ackno: 101 winSize: 13
seqno: 10 SYN: false data: " you a question...?" FIN: true	ACK: true ackno: 114 winSize: 13
seqno: 30 SYN: false data: "" FIN: false	ACK: true ackno: 127 winSize: 13
seqno: 30 SYN: false data: "" FIN: false	ACK: true ackno: 131 winSize: 13

IP, Ports, and Sockets in Networking

Definitions:

IP Address: An identifier for a device on a network, specifically following the Internet Protocol (ex. IPv4: 192.168.1.1)

Port: An identifier that identifies the application on a device where network traffic originates from, or the application where network traffic should be directed - see UDP / TCP headers (ex. Port 443 is for HTTPS)

Socket: An abstraction of a communication endpoint using a specified protocol, where one may read / write to (ex. UDPSocket, TCPSocket)

IP, Ports, and Sockets in Networking

Worked Example:

1.Your device is assigned and identified by an IP address (e.g., 192.168.1.100)

2. The host serving a website (ex. Google) is also identified by an IP address (ex. 8.8.8.8)

3. Your web browser connects to Google's server on port 443 (HTTPS)

4. A connection via socket API is established:

    a. 192.168.1.100:random_port (your side)
    b. 8.8.8.8:443 (Google's side)

5. Data is exchanged via the specified protocol between these endpoints (HTTPS in this case)

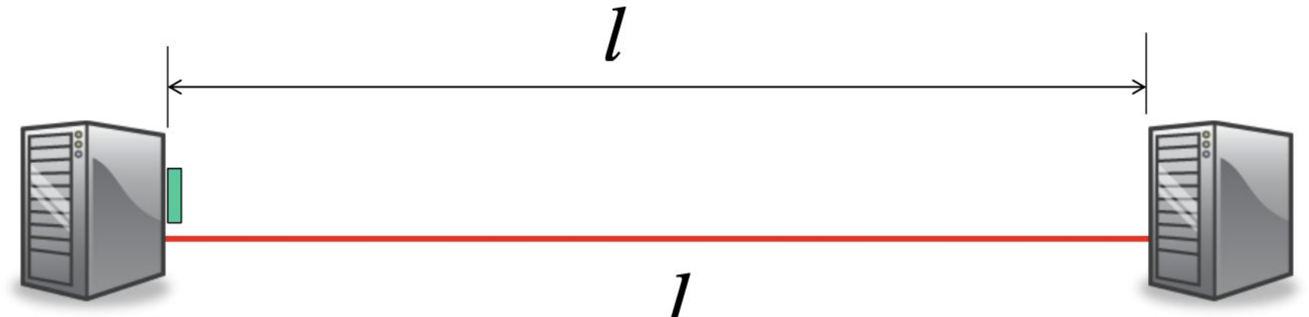
Delay Types

- Propagation Delay
- Serialization Delay
- Queueing Delay
- Store and forward Delay

Propagation Delay

The time it takes a single bit to travel over a link at propagation speed c

- L: length of the link between devices
- C: propagation speed over the fiber (close to the speed of light)

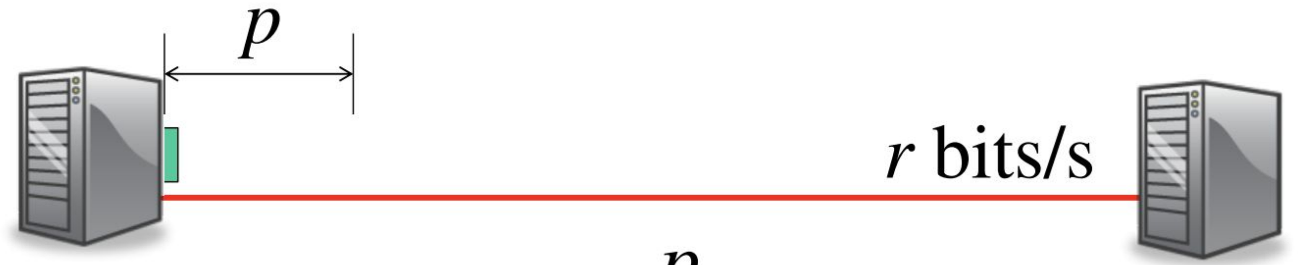


$$t_l = \frac{l}{c}$$

Serialization Delay

Time it takes for the whole packet to be received after the first bit is received

- P: size of the packet
- R: link rate (bits per second)



$$t_p = \frac{p}{r}$$

Queueing Delay

- Time a packet spends waiting in a queue before it could be sent
- **Cut-through:** Queuing starts when the first bit arrives
- **Store & Forward:** Queuing starts after the entire packet has arrived

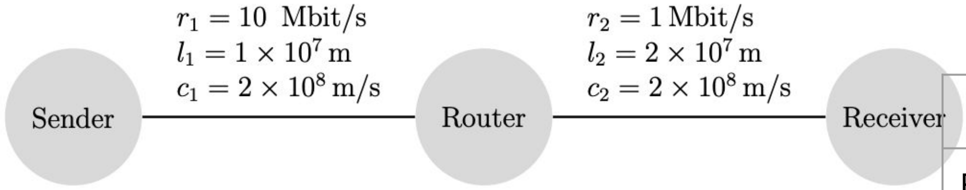
Store & Forward Delay

- The extra time a router takes to **fully receive** a packet before forwarding it to the next destination.
- This delay is absent in **cut-through** approach, since it starts forwarding immediately if received bit

Example Walkthrough(23 Spring Midterm Q2)

In this problem, we'll consider a single flow over a top-hop network path. The router is "normal" and processes each packet as a unit (the entire packet must arrive before it can begin being sent on an outgoing link).

Packets 1 and 2 are 10 kbit



Arithmetic helper: $\frac{10^7 \text{m}}{2 \times 10^8 \text{m/s}} = \frac{10^{-1}}{2}$ s $= 0.05$ s $= 50$ ms

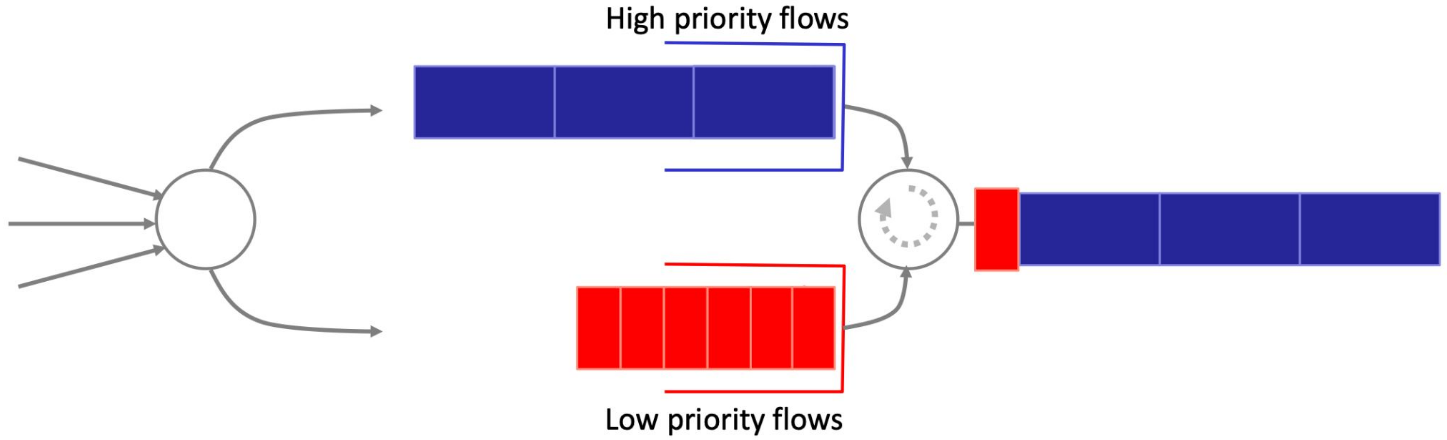
	Packet 1	Packet 2
Propagation Delay	150ms	150ms
Serialization Delay	10ms	10ms
Queueing Delay(host+router)	0ms	(1+9)ms
Store & Forward Delay	1ms	1ms
Total time for the full packet arrival	161ms	171ms

If overlap allowed between Serialization Delay & Store & Forward Delay

	Packet 1	Packet 2
Propagation Delay	150ms	150ms
Serialization Delay	11ms	11ms
Queueing Delay(host+router)	0ms	(1+9)ms
Store & Forward Delay	1ms	1ms
Total time for the full packet arrival	161ms	171ms

Queuing Policies

Strict Priorities



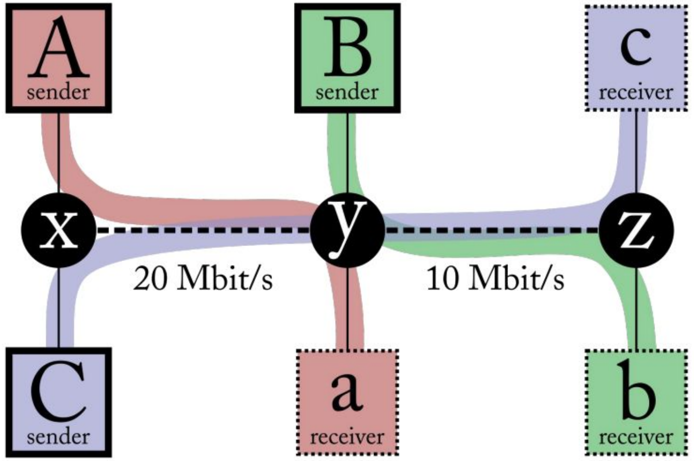
"Strict priorities" means a queue is only served when all the higher priority queues are empty

Congestion Control – Why?

- In addition to the receiver’s capacity, TCP senders should also respect the **network’s capacity** (limited by the data rates of the network’s links).
- If senders send too much data, the routers’ queues fill up and packets start to get dropped. This is called **congestion.**
- Congestion is bad because:
  - It can lead to **congestion collapse:** lots of demand for network resources, but it is not doing useful work.
  - It can cause **bad fairness:** some flows send too much and cause others to be starved.
  - Increased **queuing delay**.

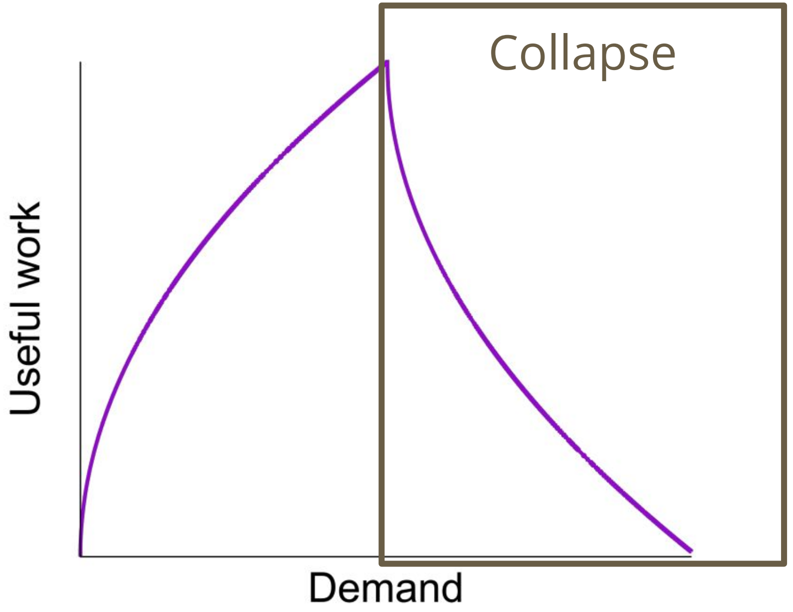
Congestion Control – Examples

A → a	B → b	C → c	Total
20	10	0	30 (max utilization)
10	0	10	20 (best for C)
0	0	20 attempted, 10 achieved	10 (collapse)
15	5	5	25 (max-min fair)
16	6	4	26 (proportionally fair)



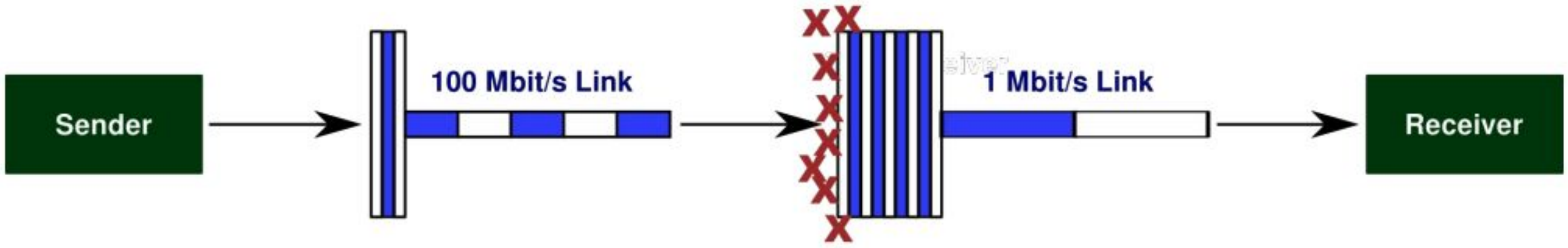
Congestion Control – Collapse

- Situation where there is high demand for resources, but lower goodput than the network could achieve.



- Some resource must be wasted.
- That resource should have been able to do useful work.

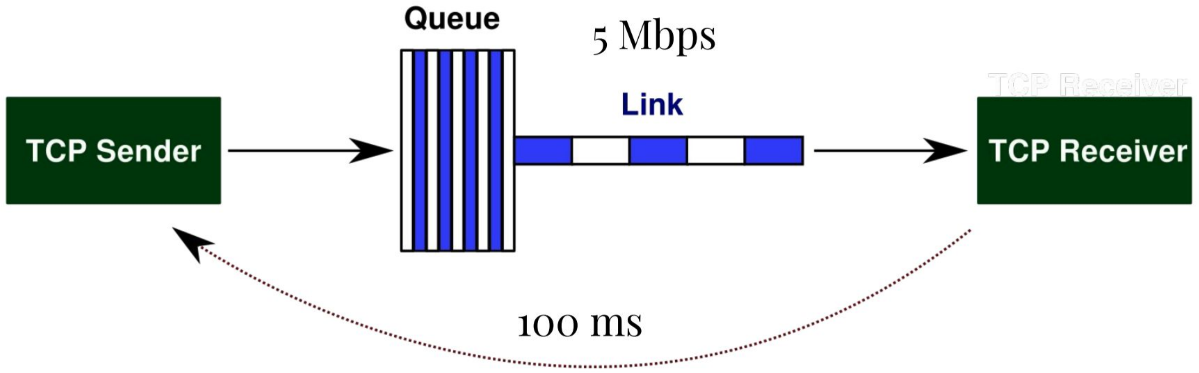
Congestion Control – Examples



- Sender might monopolize the 100 Mbit/s link between the two routers, but packets are dropped at the next link → **wasted work.**
- If another sender wanted to send data along this link, it couldn't even if that data might have improved overall goodput → **collapse occurs.**

Preventing Network Congestion

- Senders track a **‘congestion window’**, and send data until min(receiver window size, congestion window) are outstanding. When a byte is acked, one more byte can be sent → **self clocking**.
- What should cwnd be? If we were sending at the bottleneck data rate, then number of bytes outstanding is (bottleneck data rate) * (min round trip time i.e. RTT), also known as the **BDP (bandwidth delay product).**



5 Mbit/s * 100ms
= **500Kbit/s**