Last time: Packet Switching

S — r1 — X — r2 — R

Propagation Delay:

$$\left(\frac{l_1 = distance\ of\ the\ link\ from\ S\ to\ X}{c_1 = speed\ of\ light\ (in\ this\ medium)}\right)$$

+ Serialization Delay:

$$\left(\frac{p = size\ of\ the\ packet}{r_1 = link\ rate\ (bits/second)}\right)$$

—---------- How long does it take for a packet to arrive from sender S to router X—----------

+ Queueing Delay (waiting at the out-going queue):

$$\left(\sum_{i\ \in\ out-going\ queue} serializationdelay(i) = \frac{\sum_{i\ \in out-goting\ queue} size_i}{r_2 = link\ rate\ of\ the\ out-going\ link}\right)$$

—---------- After it arrives at X, how long does it take for it to leave X —----------

$$\frac{l_2}{c_2} + \frac{p}{r_2}$$

—---------- After it leaves X, how long does it take to arrive R —----------

Q: Multiplexing?

A: The underlying links are multiplexed across different flows. Each packet is an atomic unit.

Q: How does a router know a flow ends?

A: The router knows nothing about the flow. It only cares about each individual packet.

Next: Congestion Control

As you see in the Gradescope problem, the performance is bottlenecked at leaving A, because L2 is much slower than L1. What happens if more and more packets are waiting for the same out-going queue? That could potentially cause a buffer overflow at A. The throughput of sending from S to R is bottlenecked by the slowest link on the path. Is there anything smart we could do to prevent more and more packets waiting at A?