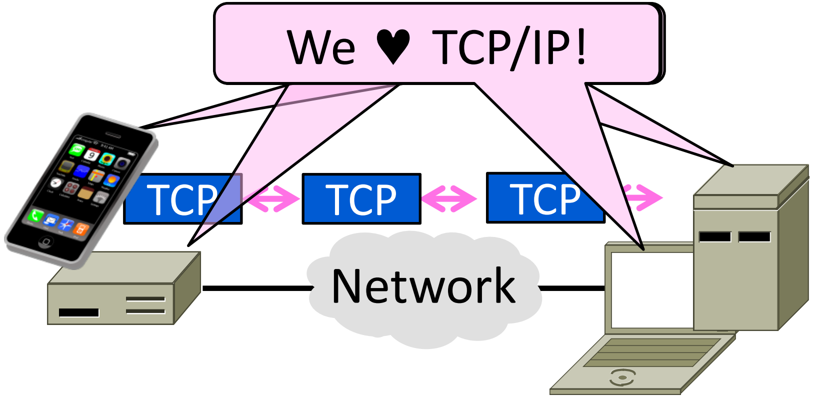
Topic

- How TCP works!
  - The transport protocol used for most content on the Internet



TCP Features

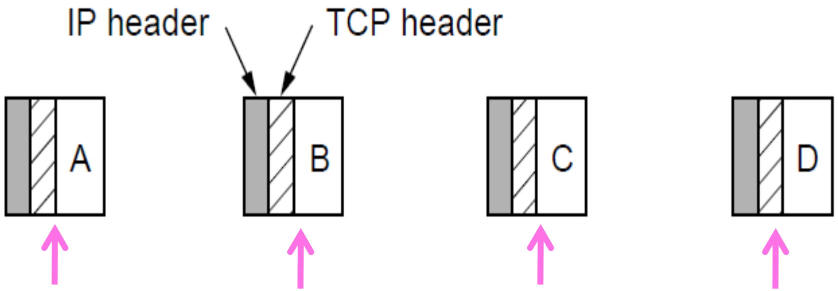
- A reliable bytestream service »
- Based on connections
- Sliding window for reliability »
  - With adaptive timeout
- Flow control for slow receivers

- Congestion control to allocate network bandwidth

Reliable Bytestream

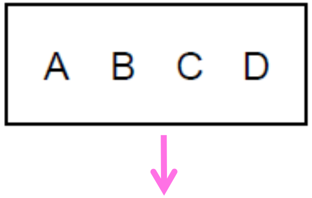
- Message boundaries not preserved from send() to recv()
  - But reliable and ordered (receive bytes in same order as sent)

Sender



Four segments, each with 512 bytes of data and carried in an IP packet

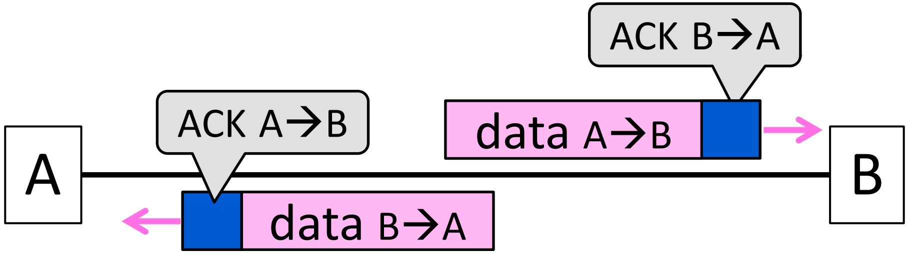
Receiver



2048 bytes of data delivered to app in a single recv() call

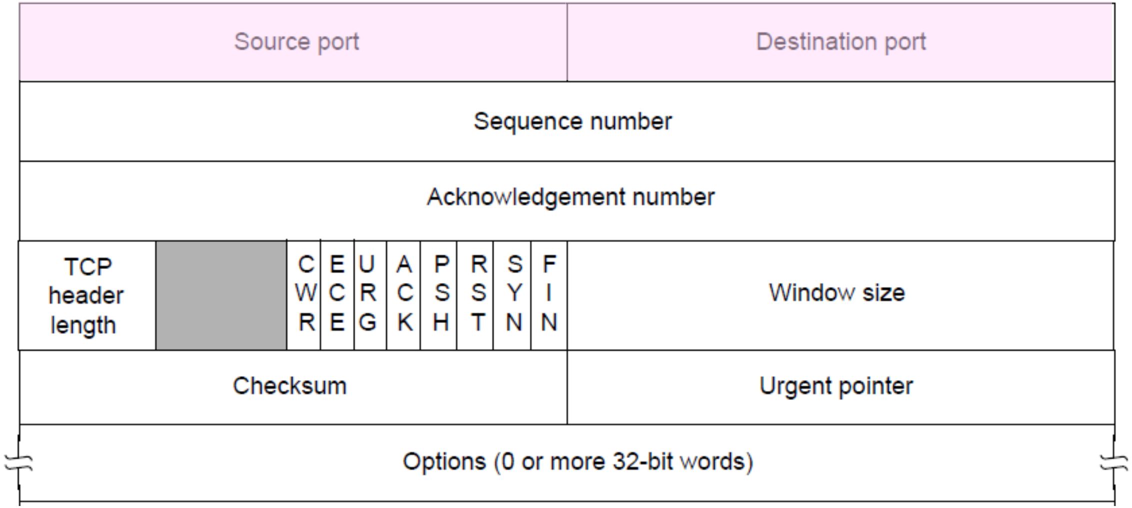
Reliable Bytestream (2)

- Bidirectional data transfer
  - Control information (e.g., ACK) piggybacks on data segments in reverse direction



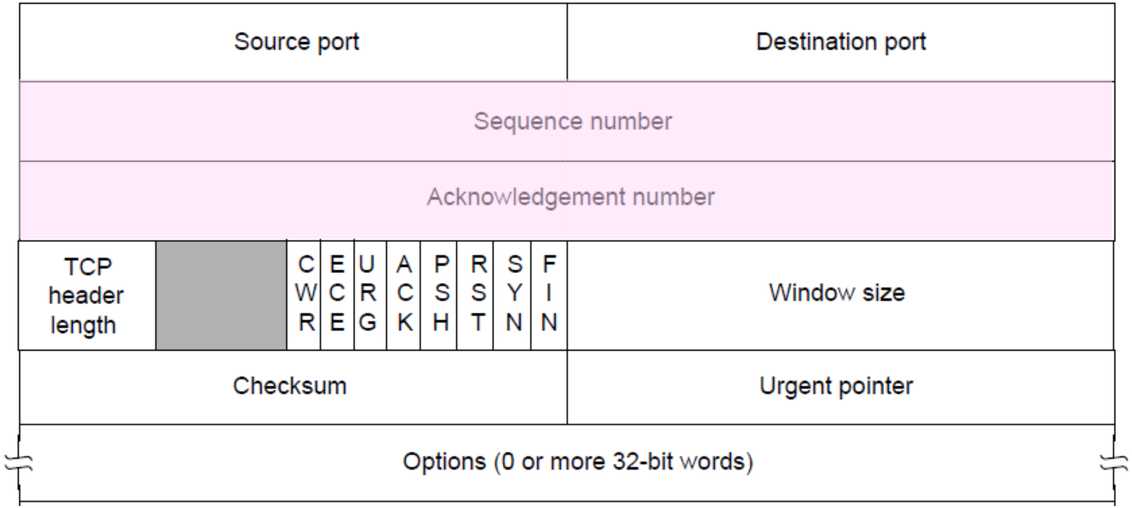
TCP Header (1)

- Ports identify apps (socket API)
  - 16-bit identifiers



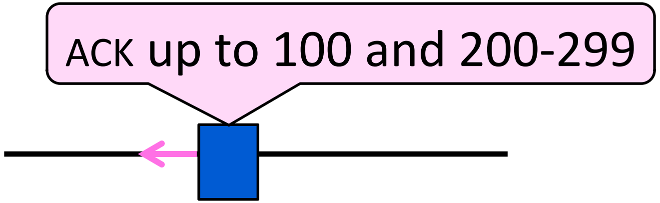
TCP Header (2)

- SEQ/ACK used for sliding window
  - Selective Repeat, with byte positions



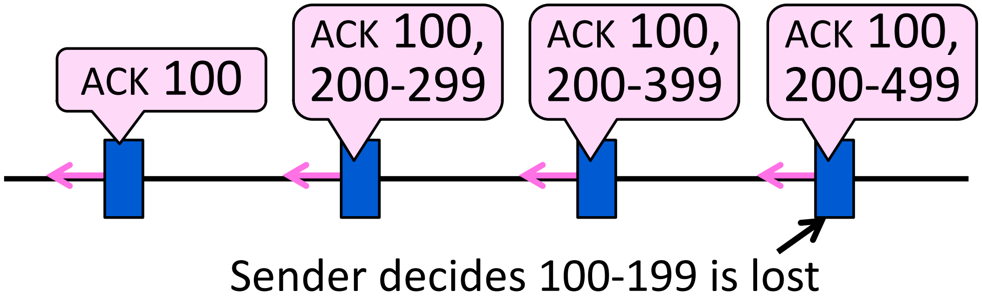
TCP Sliding Window – Receiver

- Cumulative ACK tells next expected byte sequence number ("LAS+1")
- Optionally, selective ACKs (SACK) give hints for receiver buffer state
  - List up to 3 ranges of received bytes



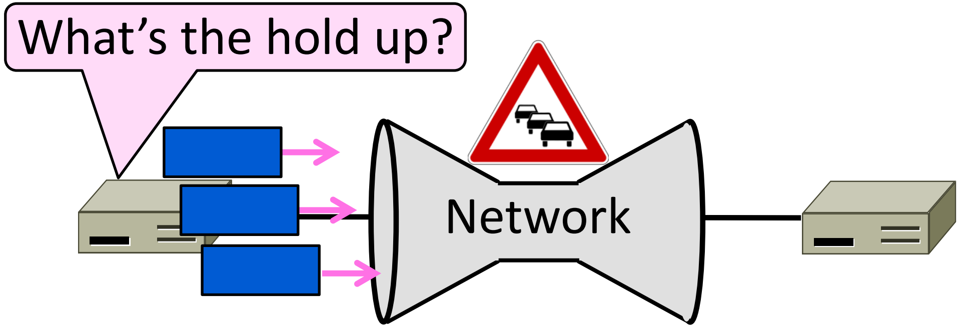
TCP Sliding Window – Sender

- Uses an adaptive retransmission timeout to resend data from LAS+1
- Uses heuristics to infer loss quickly and resend to avoid timeouts
  - "Three duplicate ACKs" treated as loss



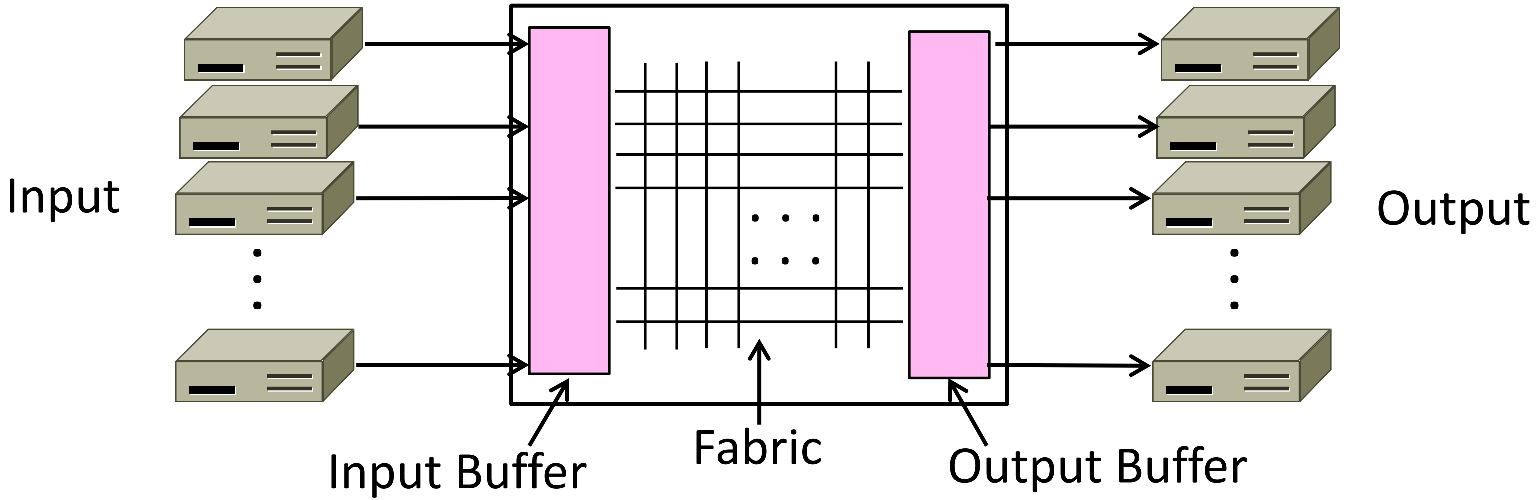
Topic

- Understanding congestion, a "traffic jam" in the network
  - Later we will learn how to control it



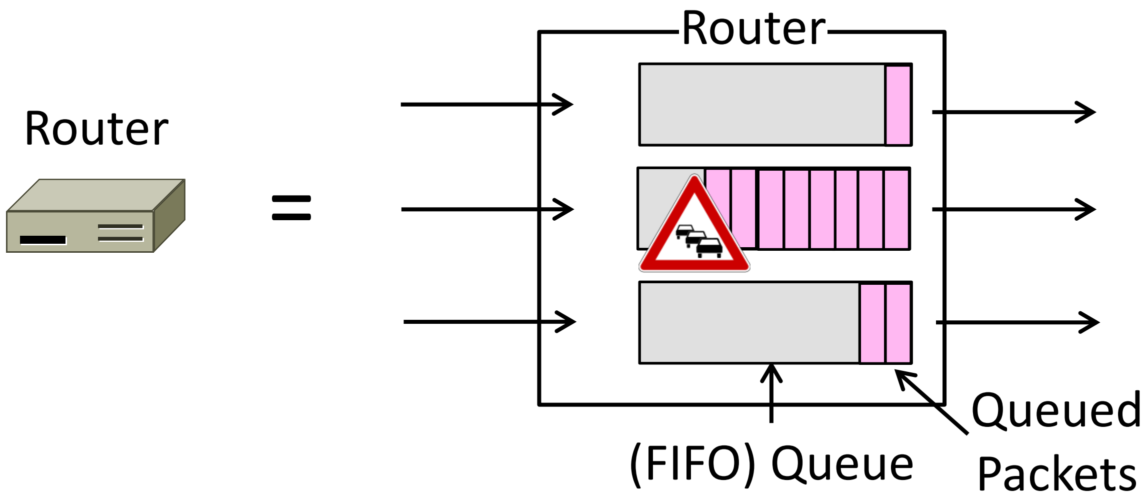
Nature of Congestion

- Routers/switches have internal buffering for contention



Nature of Congestion (2)

- Simplified view of per port output queues
  - Typically FIFO (First In First Out), discard when full

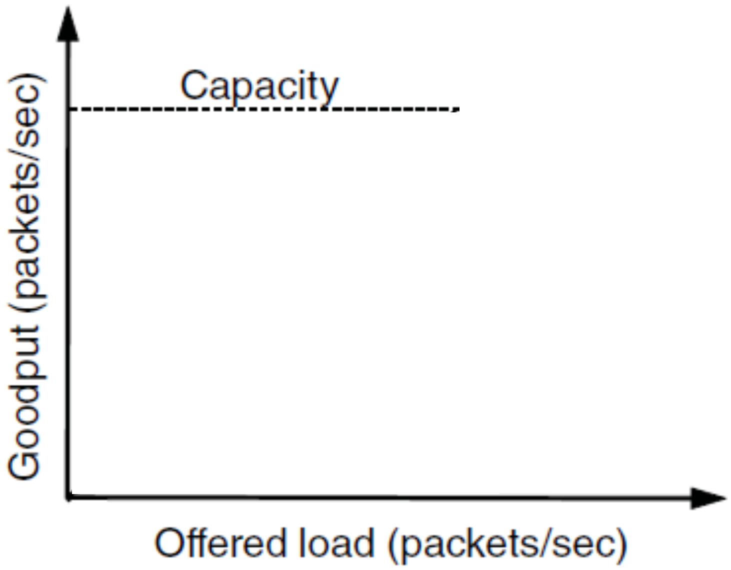


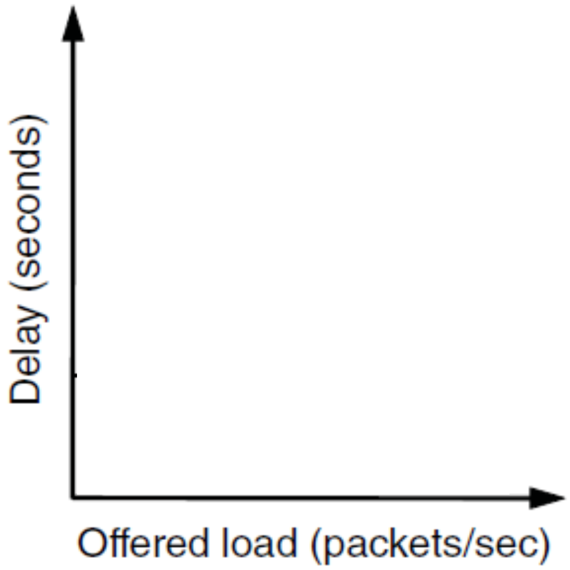
Nature of Congestion (3)

- Queues help by absorbing bursts when input > output rate
- But if input > output rate persistently, queue will overflow
  - This is congestion
- Congestion is a function of the traffic patterns – can occur even if every link have the same capacity

Effects of Congestion

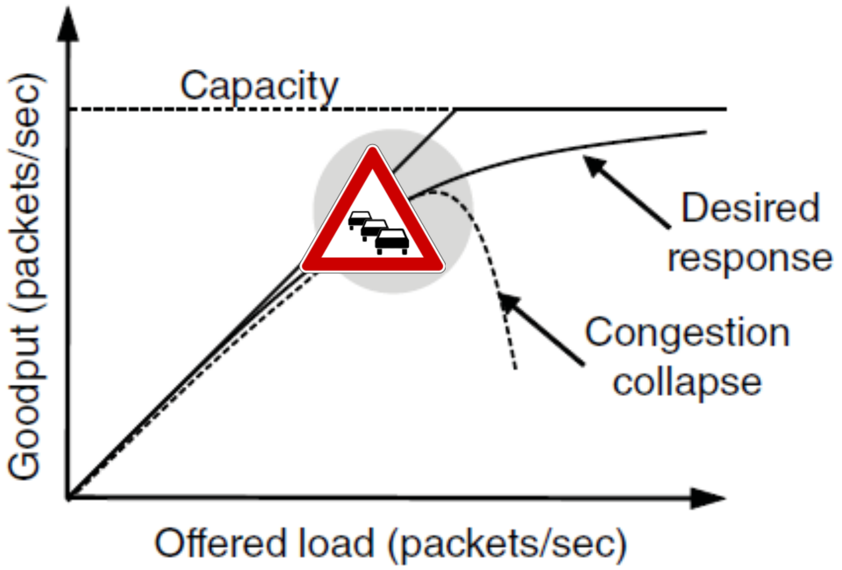
- What happens to performance as we increase the load?

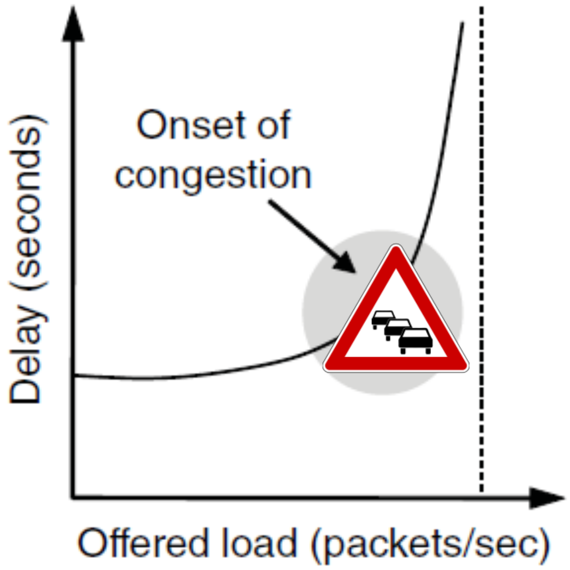




Effects of Congestion (2)

- What happens to performance as we increase the load?





Effects of Congestion (3)

- As offered load rises, congestion occurs as queues begin to fill:
  - Delay and loss rise sharply with more load
  - Throughput falls below load (due to loss)
  - Goodput may fall below throughput (due to spurious retransmissions)

- None of the above is good!
  - Want to operate network just before the onset of congestion

Bandwidth Allocation

- Important task for network is to allocate its capacity to senders
  - Good allocation is efficient and fair
- Efficient means most capacity is used but there is no congestion
- Fair means every sender gets a reasonable share the network

Bandwidth Allocation (2)

- Key observation:
  - In an effective solution, Transport and Network layers must work together
- Network layer witnesses congestion
  - Only it can provide direct feedback
- Transport layer causes congestion
  - Only it can reduce offered load

Bandwidth Allocation (3)

- Why is it hard? (Just split equally!)
  - Number of senders and their offered load is constantly changing
  - Senders may lack capacity in different parts of the network
  - Network is distributed; no single party has an overall picture of its state

Bandwidth Allocation (4)

- Solution context:
  - Senders adapt concurrently based on their own view of the network
  - Design this adaption so the network usage as a whole is efficient and fair
  - Adaption is continuous since offered loads continue to change over time

Topics

- Nature of congestion
- Fair allocations
- AIMD control law
- TCP Congestion Control history
- ACK clocking
- TCP Slow-start
- TCP Fast Retransmit/Recovery
- Congestion Avoidance (ECN)

Topic

- What's a "fair" bandwidth allocation?
  - The max-min fair allocation

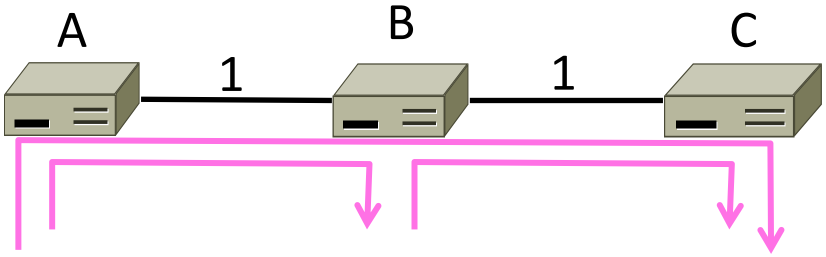
Recall

- We want a good bandwidth allocation to be fair and efficient
  - Now we learn what fair means

- Caveat: in practice, efficiency is more important than fairness

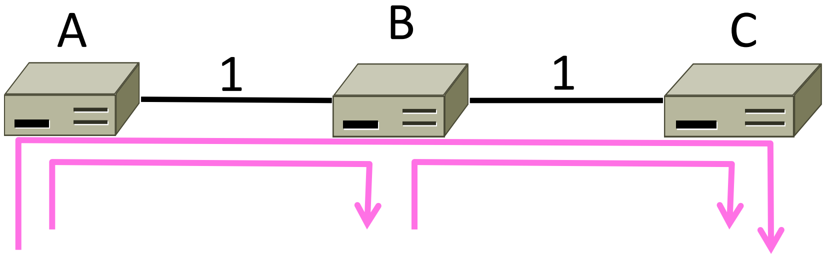
Efficiency vs. Fairness

- Cannot always have both!
  - Example network with traffic A→B, B→C and A→C
  - How much traffic can we carry?



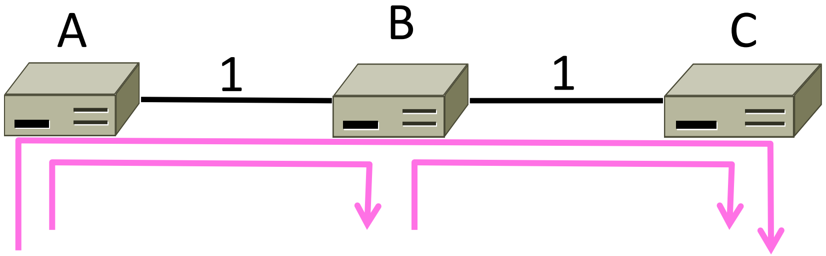
Efficiency vs. Fairness (2)

- If we care about fairness:
  - Give equal bandwidth to each flow
  - A→B: ½ unit, B→C: ½, and A→C, ½
  - Total traffic carried is 1 ½ units



Efficiency vs. Fairness (3)

- If we care about efficiency:
  - Maximize total traffic in network
  - A→B: 1 unit, B→C: 1, and A→C, 0
  - Total traffic rises to 2 units!

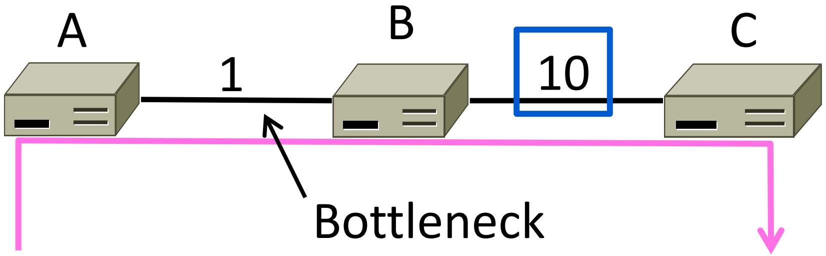


The Slippery Notion of Fairness

- Why is “equal per flow” fair anyway?
  - A→C uses more network resources (two links) than A→B or B→C
  - Host A sends two flows, B sends one
- Not productive to seek exact fairness
  - More important to avoid starvation
  - “Equal per flow” is good enough

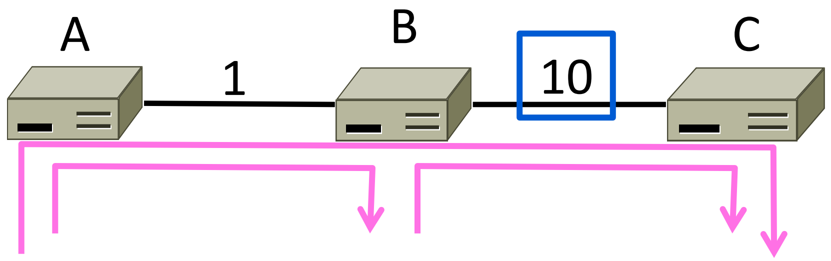
Generalizing "Equal per Flow"

- Bottleneck for a flow of traffic is the link that limits its bandwidth
  - Where congestion occurs for the flow
  - For A→C, link A–B is the bottleneck



Generalizing "Equal per Flow" (2)

- Flows may have different bottlenecks
  - For A→C, link A–B is the bottleneck
  - For B→C, link B–C is the bottleneck
  - Can no longer divide links equally …



Max-Min Fairness

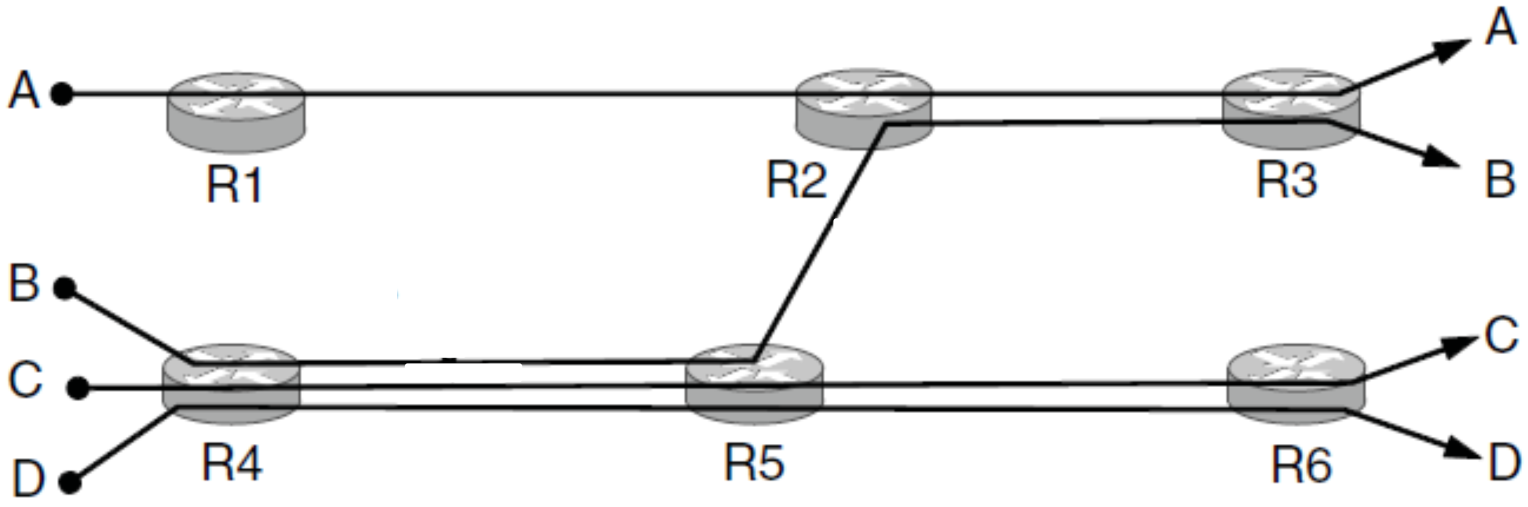
- Intuitively, flows bottlenecked on a link get an equal share of that link
- Max-min fair allocation is one that:
  - Increasing the rate of one flow will decrease the rate of a smaller flow
  - This "maximizes the minimum" flow

Max-Min Fairness (2)

- To find it given a network, imagine "pouring water into the network"
  1. Start with all flows at rate 0
  2. Increase the flows until there is a new bottleneck in the network
  3. Hold fixed the rate of the flows that are bottlenecked
  4. Go to step 2 for any remaining flows

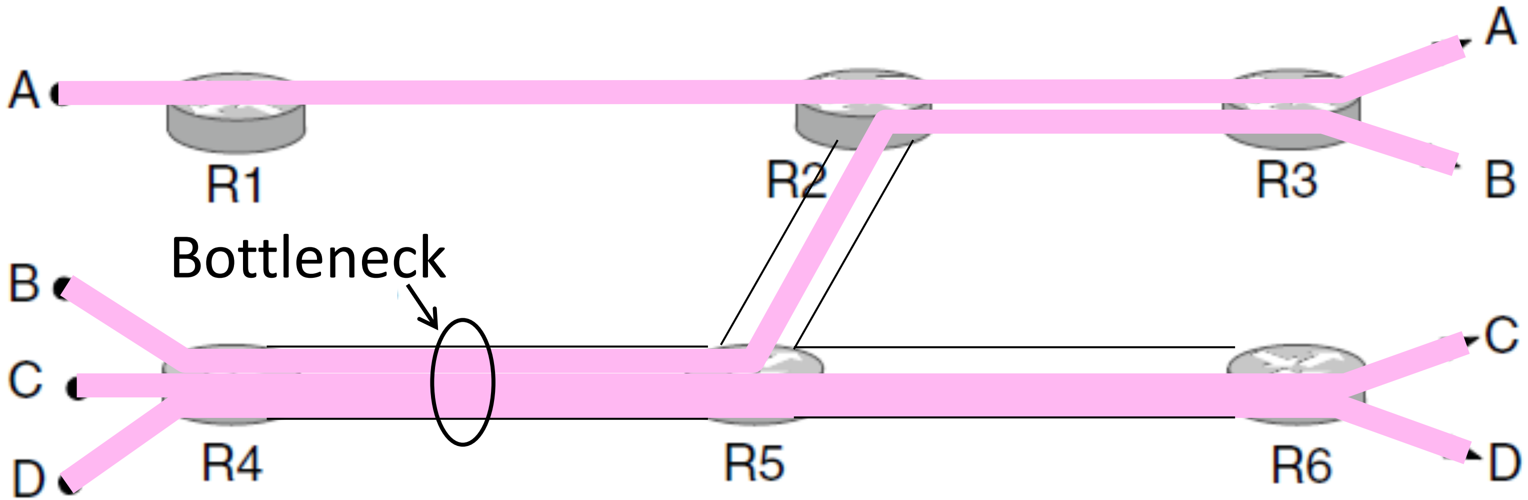
Max-Min Example

- Example: network with 4 flows, links equal bandwidth
  - What is the max-min fair allocation?



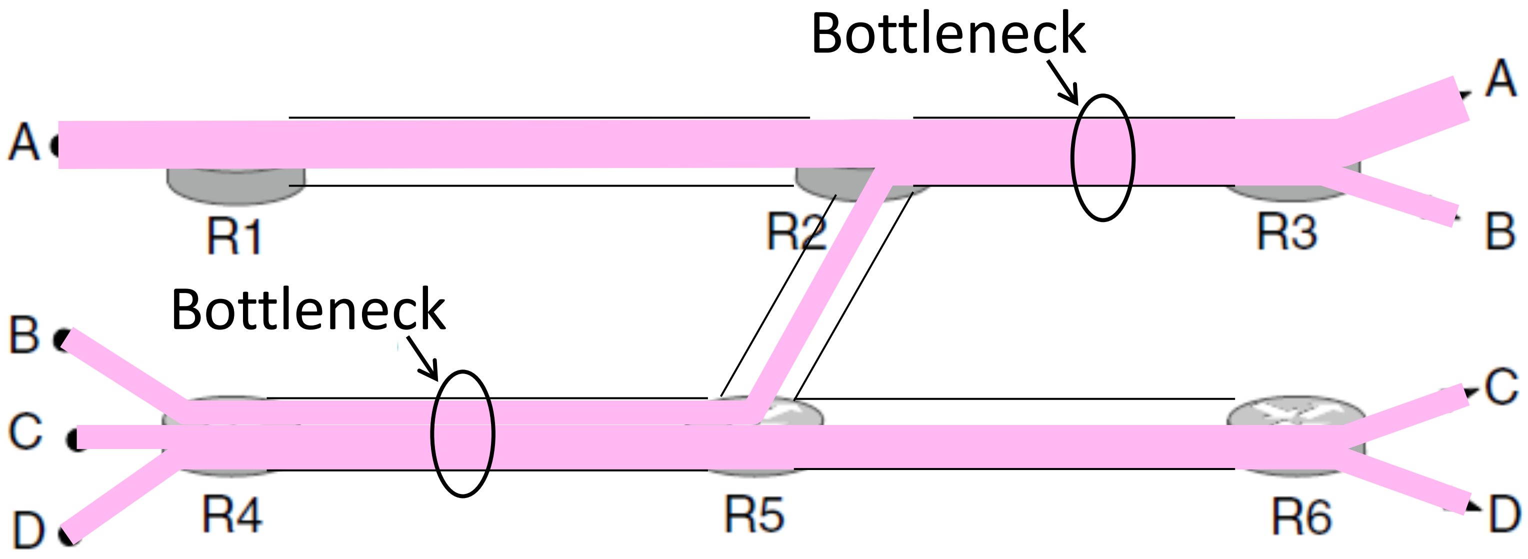
Max-Min Example (2)

- When rate=1/3, flows B, C, and D bottleneck R4—R5
  - Fix B, C, and D, continue to increase A

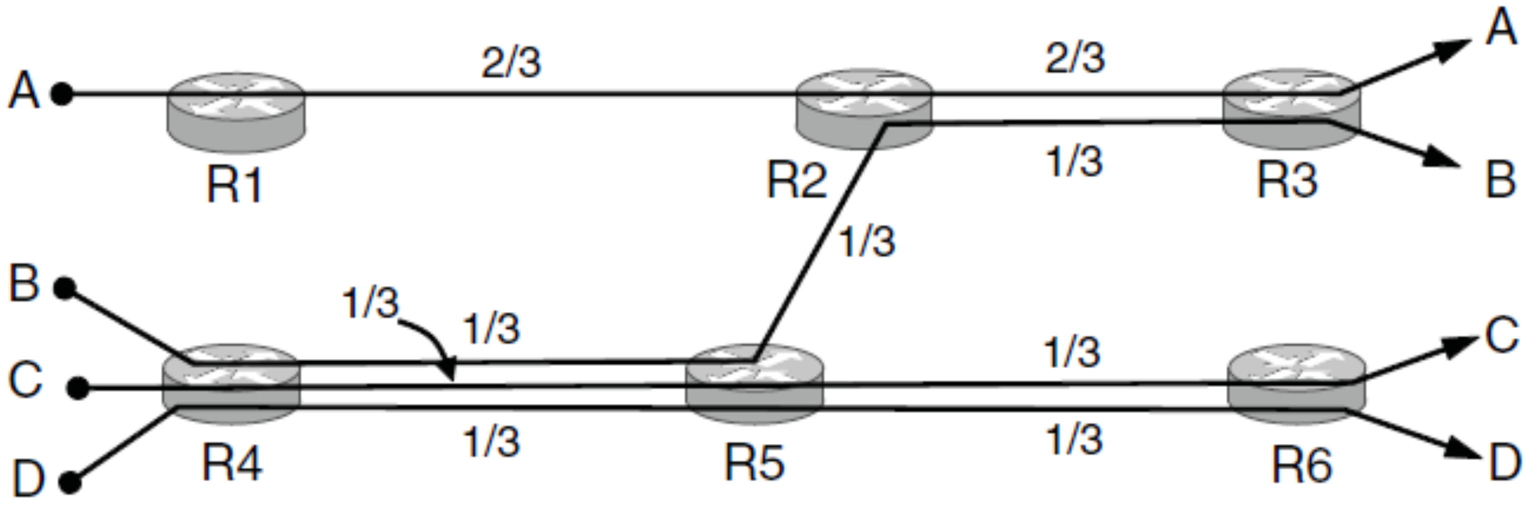


Max-Min Example (3)

- When rate=2/3, flow A bottlenecks R2—R3. Done.

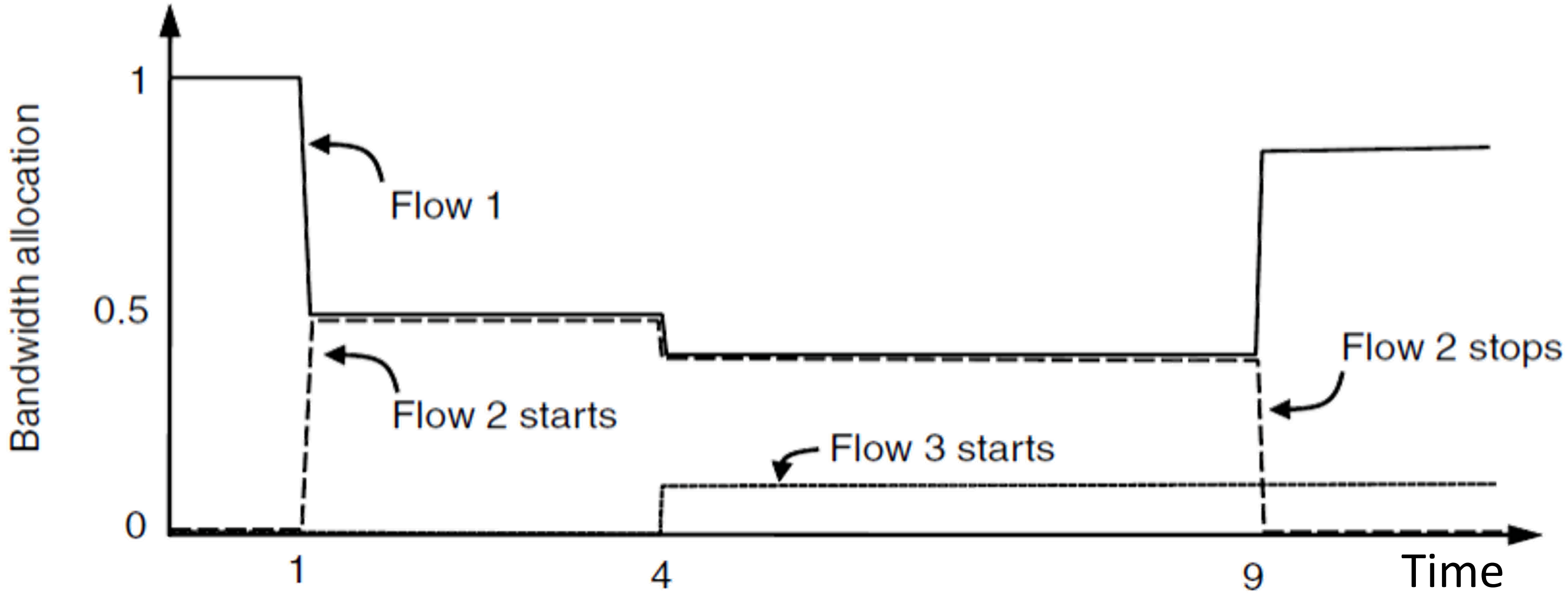


Max-Min Example (4)

- End with A=2/3, B, C, D=1/3, and R2—R3, R4—R5 full
  - Other links have extra capacity that can't be used
- 

Adapting over Time

- Allocation changes as flows start and stop



Adapting over Time (2)