Lecture 20: Scheduling and QoS

CSE 123: Computer Networks

Alex C. Snoeren

Lecture 20 Overview

- TCP Bandwidth Probing
- Scheduling
  - (Weighted) Fair Queuing

TCP Bandwidth Probing

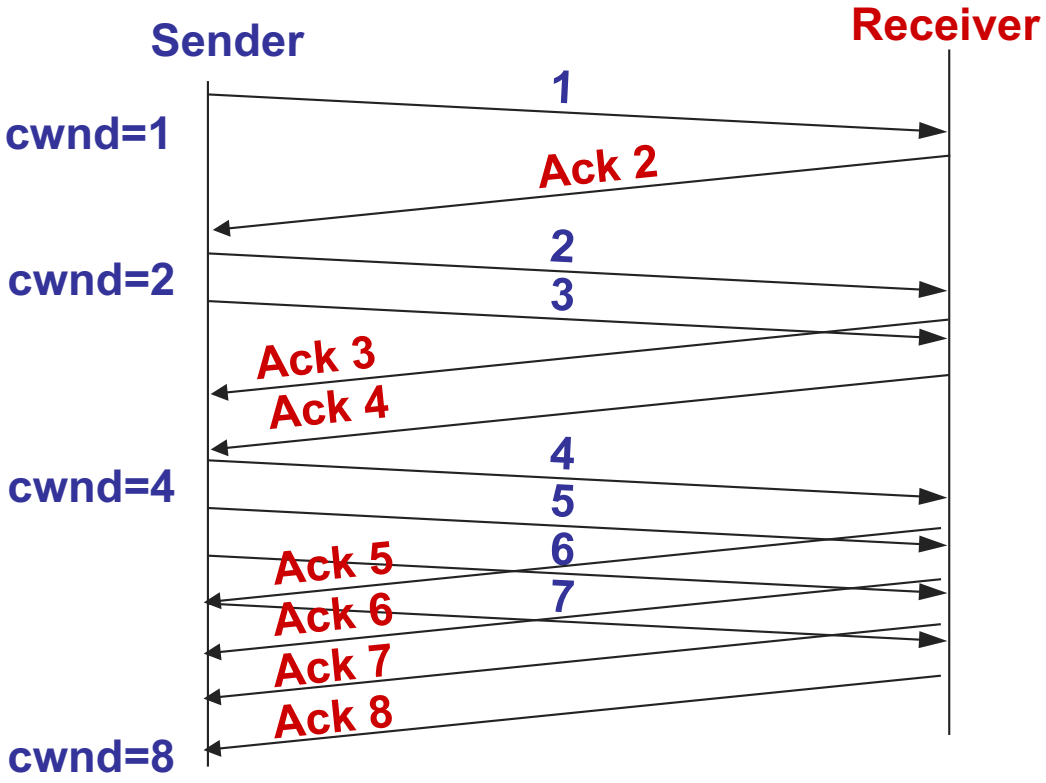
- TCP uses AIMD to adjust congestion window
  - Converges to fair share of bottleneck link
  - Increases modestly in good times
  - Cuts drastically in bad times

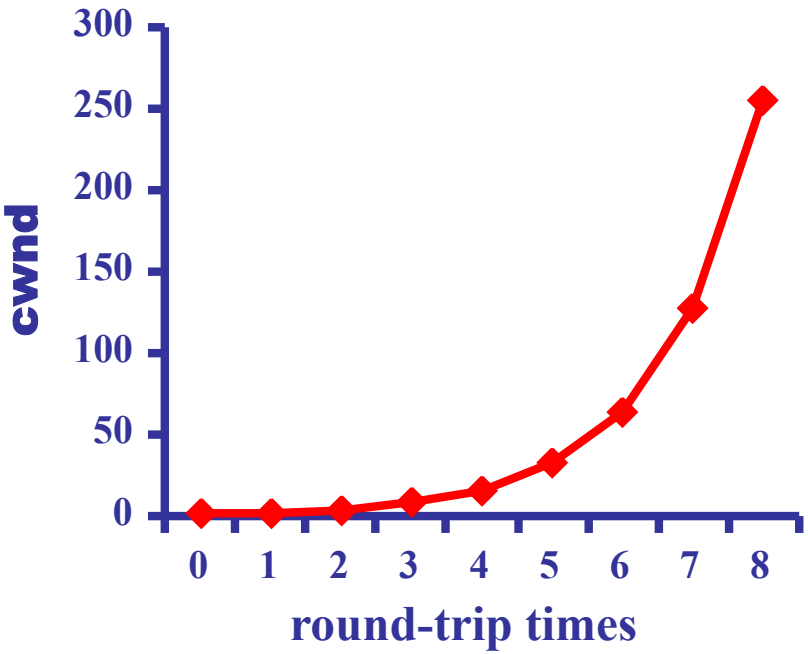
- But what rate should a TCP flow use initially?
  - Need some initial congestion window
  - We'd like to TCP to work on all manner of links
  - Need to span 6+ orders of magnitude, e.g., 10 K to 10 Gbps.
  - Starting too fast is catastrophic!

Slow Start

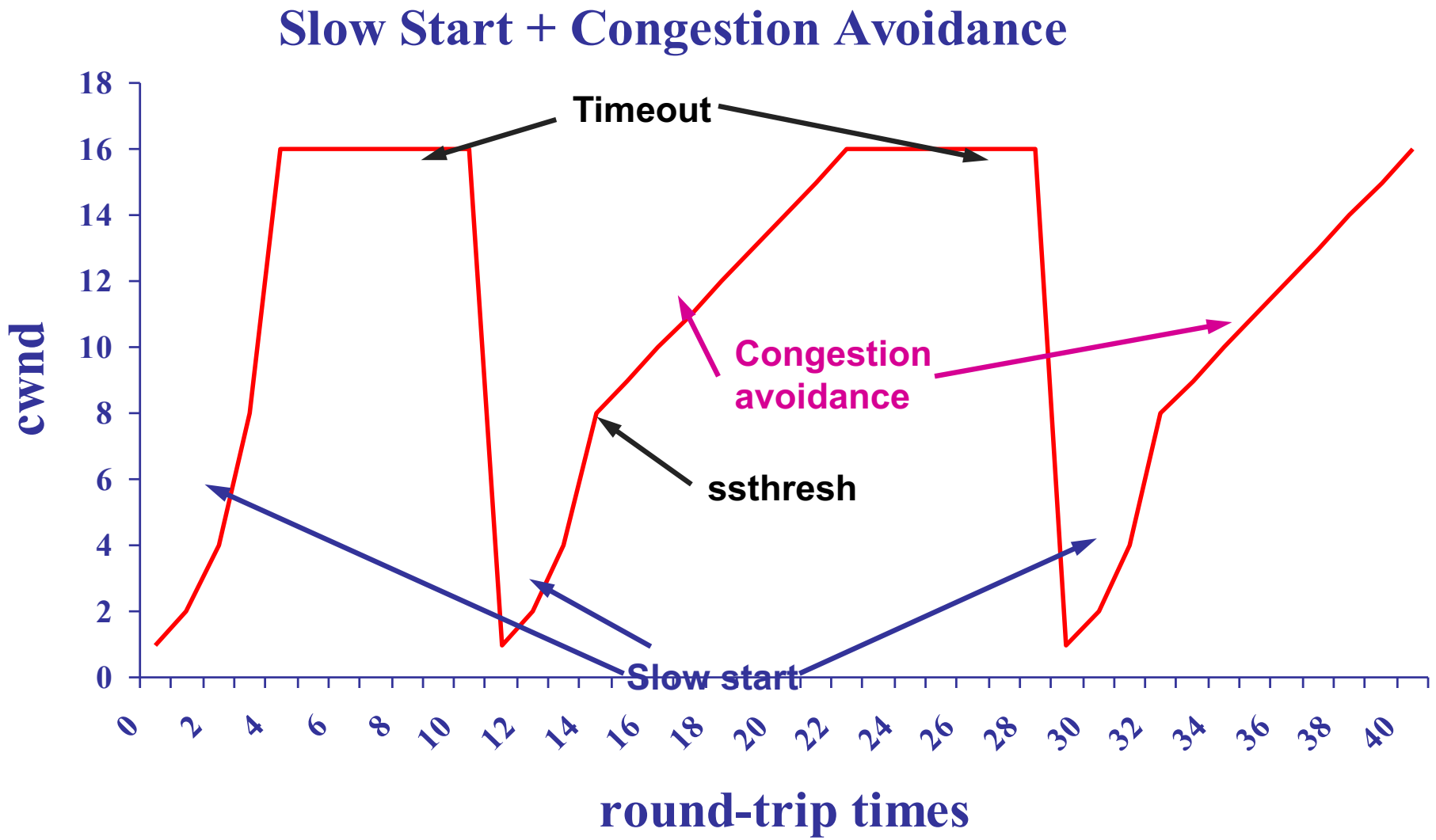
- Goal: quickly find the equilibrium sending rate
- Quickly increase sending rate until congestion detected
  - Remember last rate that worked and don't overshoot it
- TCP Reno Algorithm:
  - On new connection, or after timeout, set *cwnd*=1 MSS
  - For each segment acknowledged, increment *cwnd* by 1 MSS
  - If timeout then divide *cwnd* by 2, and set *ssthresh* = *cwnd*
  - If *cwnd* >= *ssthresh* then exit slow start
- Why called slow? Its exponential after all…

Slow Start Example





Basic Mechanisms

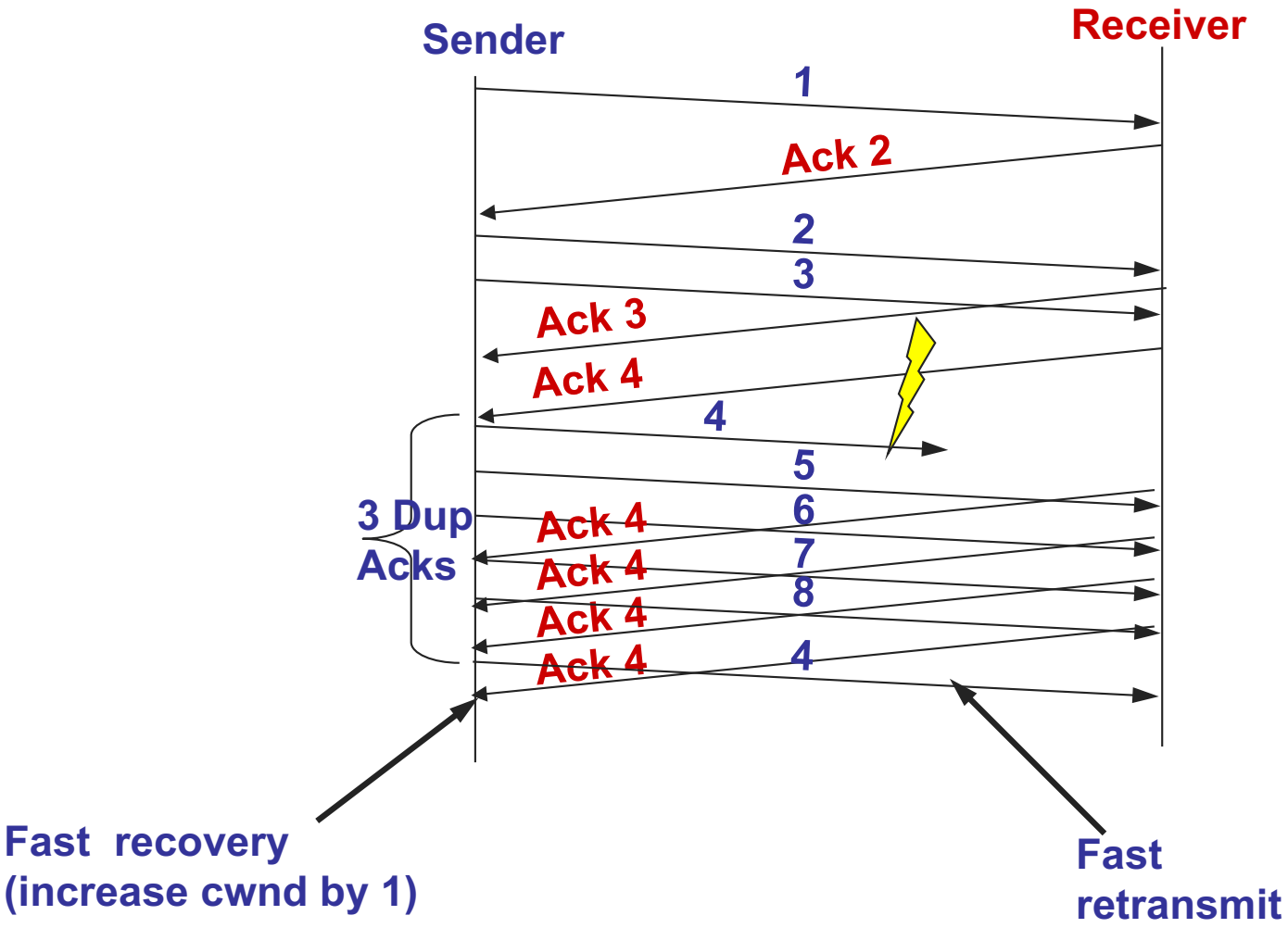


Fast Retransmit & Recovery

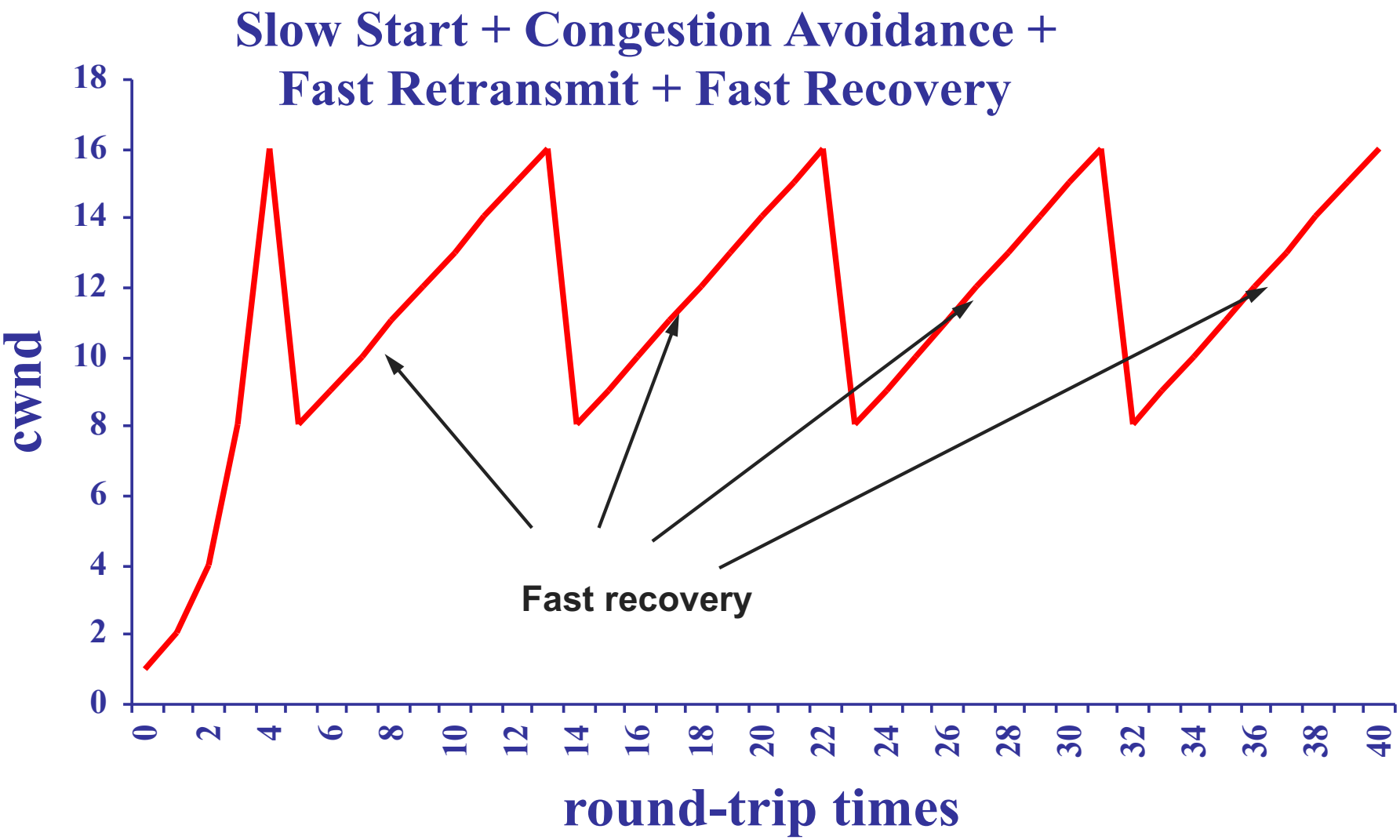
- Fast retransmit
  - Timeouts are slow (default often 200 ms or 1 second)
  - When packet is lost, receiver still ACKs last in-order packet
  - Use 3 duplicate ACKs to indicate a loss; detect losses quickly
    - » Why 3? When wouldn't this work?

- Fast recovery
  - Goal: avoid stalling after loss
  - If there are still ACKs coming in, then no need for slow start
  - If a packet has made it through -> we can send another one
  - Divide *cwnd* by 2 after fast retransmit
  - Increment cwnd by 1 MSS for each additional duplicate ACK

Fast Retransmit Example



More Sophistication



Short Connections

- Short connection: only contains a few pkts
- How do short connections and Slow-Start interact?
  - What happens when a packet is lost during Slow-Start?
  - What happens when the SYN is dropped?
- Bottom line: Which packet gets dropped matters a lot
  - SYN
  - Slow-Start
  - Congestion avoidance
- Do you think most flows are short or long?
- Do you think most traffic is in short flows or long flows?

Open Issues

- TCP is designed around the premise of cooperation
  - What happens to TCP if it competes with a UDP flow?
  - What if we divide *cwnd* by 3 instead of 2 after a loss?
- There are a bunch of magic numbers
  - Decrease by 2x, increase by 1/*cwnd*, 3 duplicate acks, initial timeout = 3 seconds, etc.
- But overall it works really well!
  - Still being constantly tweaked…

TCP CC Summary

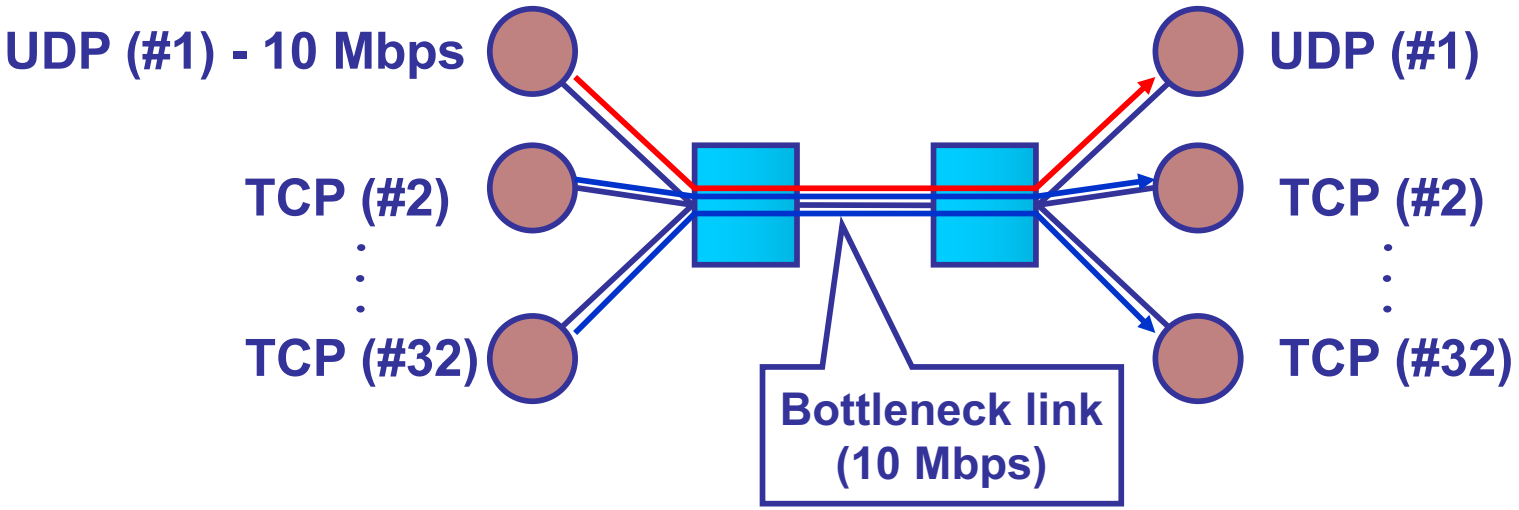
- TCP Probes the network for bandwidth, assuming that loss signals congestion
- The congestion window is managed with an additive increase/multiplicative decrease policy
  - It took fast retransmit and fast recovery to get there
  - Fast recovery keeps pipe “full” while recovering from a loss
- Slow start is used to avoid lengthy initial delays
  - Ramp up to near target rate, then switch to AIMD

Scheduling

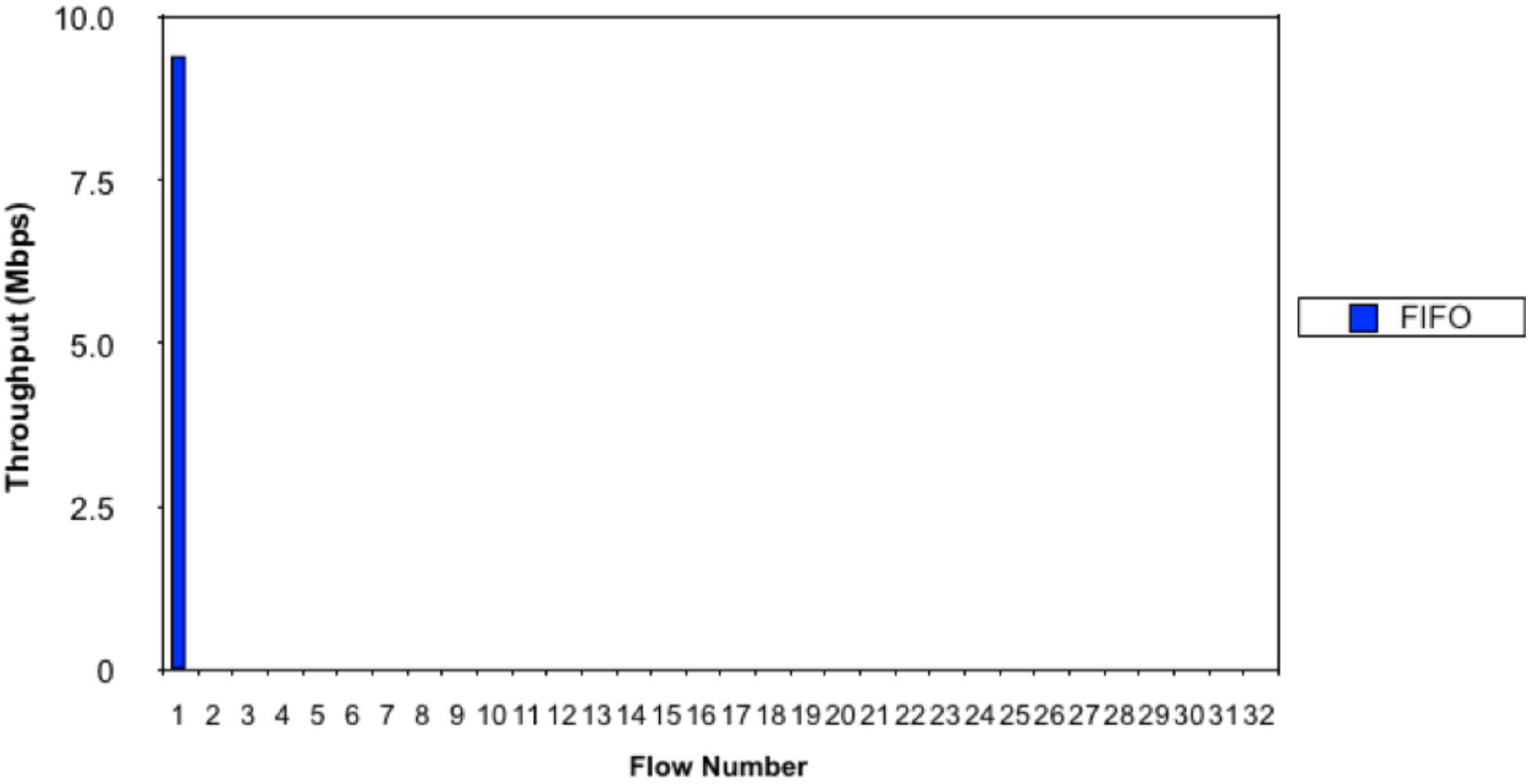
- So far we've done flow-based traffic policing
  - Limit the rate of one flow regardless of the load in the network
- In general, need scheduling
  - Dynamically allocate resources when multiple flows compete
  - Give each "flow" (or traffic class) own queue (at least theoretically)
- Weighted fair queuing
  - Proportional share scheduling
  - Schedule round-robins among queues in proportion to some weight parameter

Our Previous Example

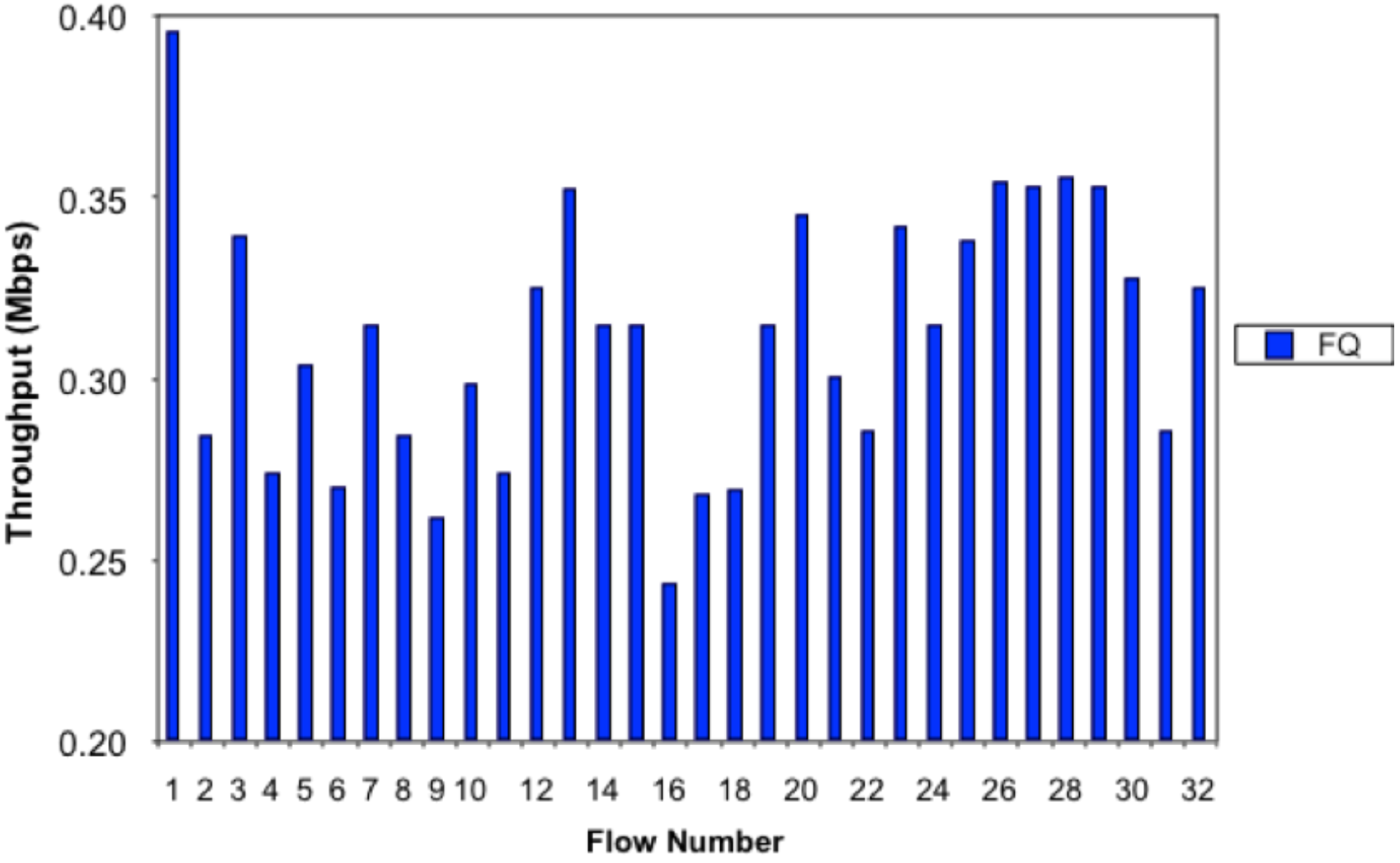
1 UDP (10 Mbps) and 31 TCPs sharing a 10 Mbps line



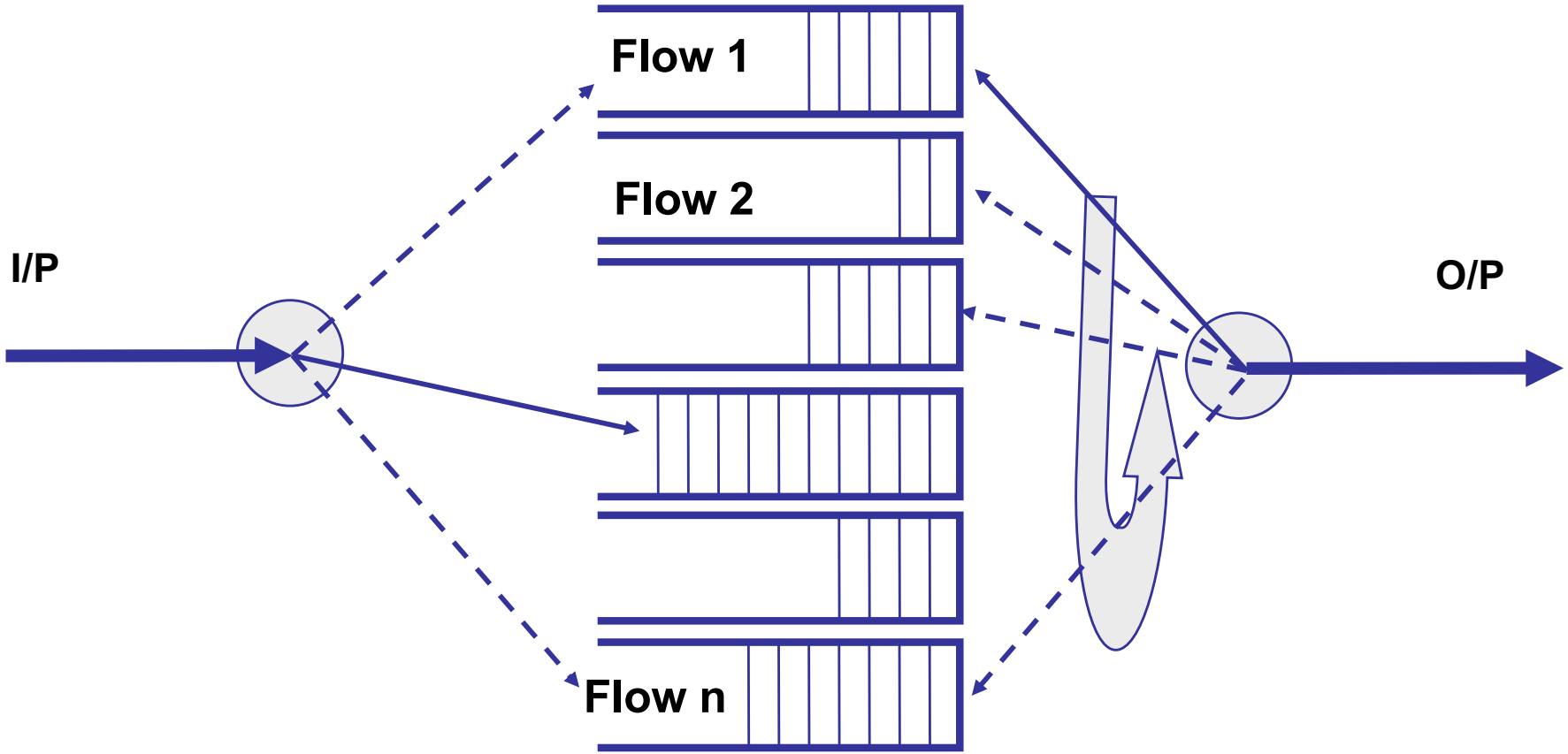
UDP *vs.* TCP w/FIFO



TCP vs. UDP w/Fair Queuing



(Weighted) Fair Queuing



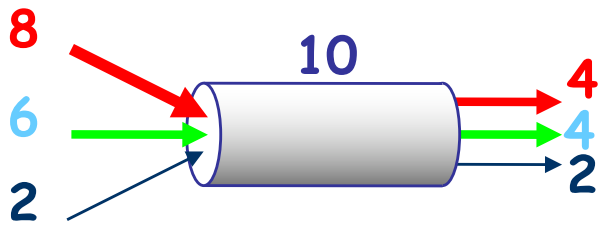
Fair Queuing

- Maintain a queue for each flow
  - What is a flow?
- Implements max-min fairness: each flow receives $min(r_i, f)$, where
  - r_i – flow arrival rate
  - f – link fair rate (see next slide)
- Weighted Fair Queuing (WFQ) – associate a weight with each flow to divvy bandwidth up non-equally

Fair Rate Computation

- If link congested, compute f such that

$$\sum_i \min(r_i, f) = C$$

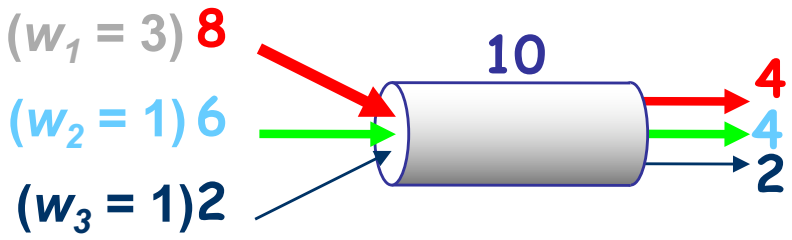


$f = 4$:
min(8, 4) = 4
min(6, 4) = 4
min(2, 4) = 2

Another Example

- Associate a weight w_i with each flow i
- If link congested, compute f such that

$$\sum_i \min(r_i, f \times w_i) = C$$



$f = 2$:
min(8, 2*3) = 6
min(6, 2*1) = 2
min(2, 2*1) = 2

Flow i is guaranteed to be allocated a rate >= w_i*C/(Σ_k w_k)

⇩

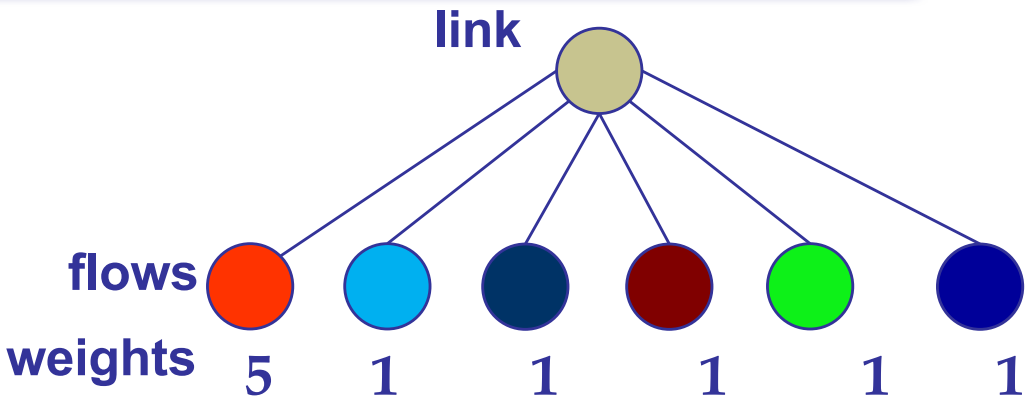
If Σ_k w_k <= C, flow i is guaranteed to be allocated a rate >= w_i

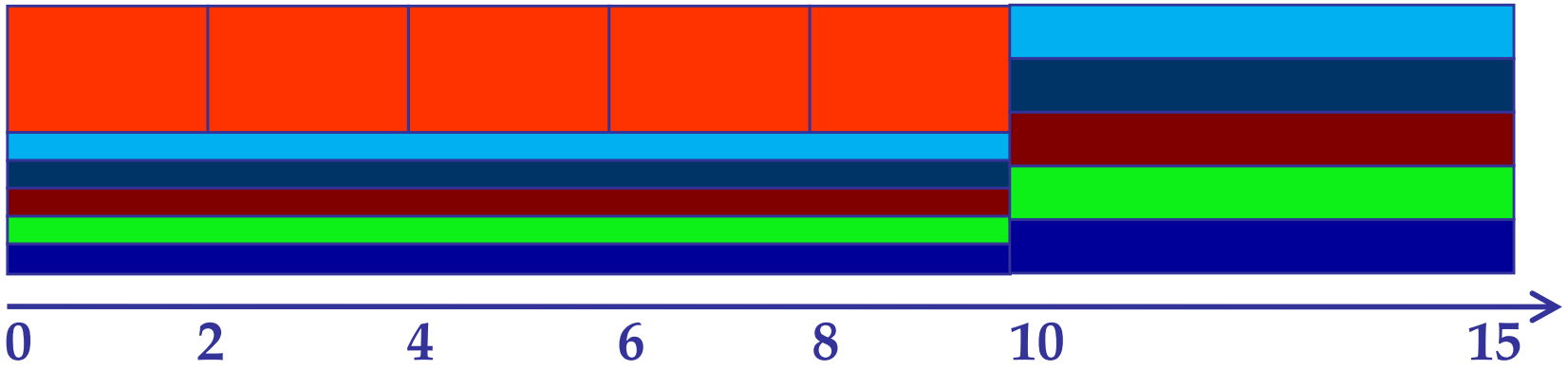
Fluid Flow

- Flows can be served one bit at a time
- WFQ can be implemented using bit-by-bit weighted round robin
  - During each round from each flow that has data to send, send a number of bits equal to the flow's weight

Fluid Flow Example

- Orange flow has packets backlogged between time 0 and 10
- Other flows have packets continuously backlogged
- All packets have the same size



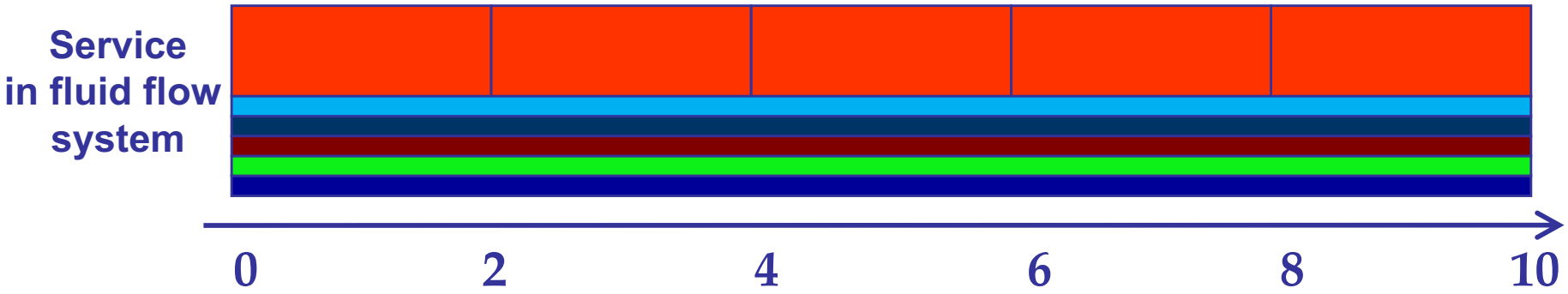


Packet-Based Implementation

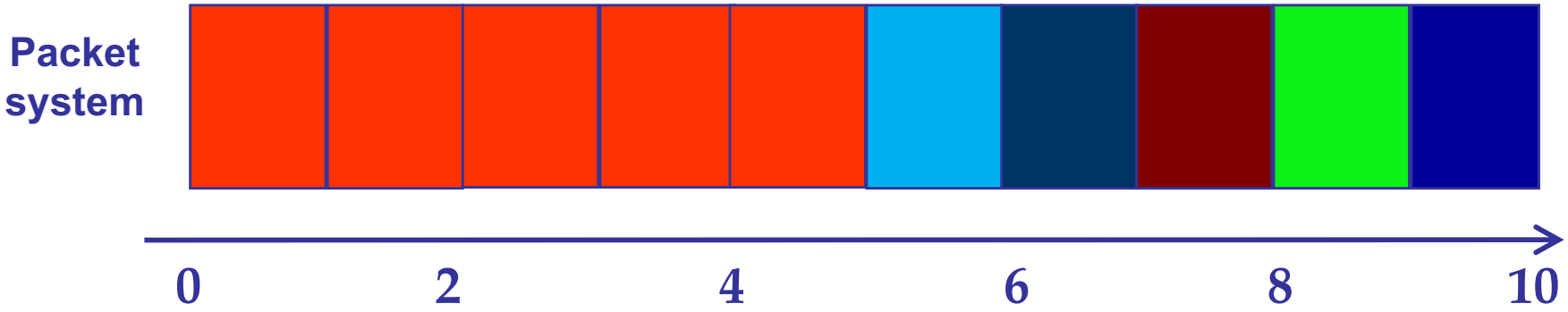
- Packet (Real) system: packet transmission cannot be preempted. Why?

- Solution: serve packets in the order in which they would have finished being transmitted in the fluid flow system

Packet-Based Example



- Select the first packet that finishes in the fluid flow system



For next time…

- Changing gears severely next lecture!
- Read Ch. 1.5 in P&D