

Lecture 21:

Buffering & Scheduling

CSE 123: Computer Networks

Aaron Schulman





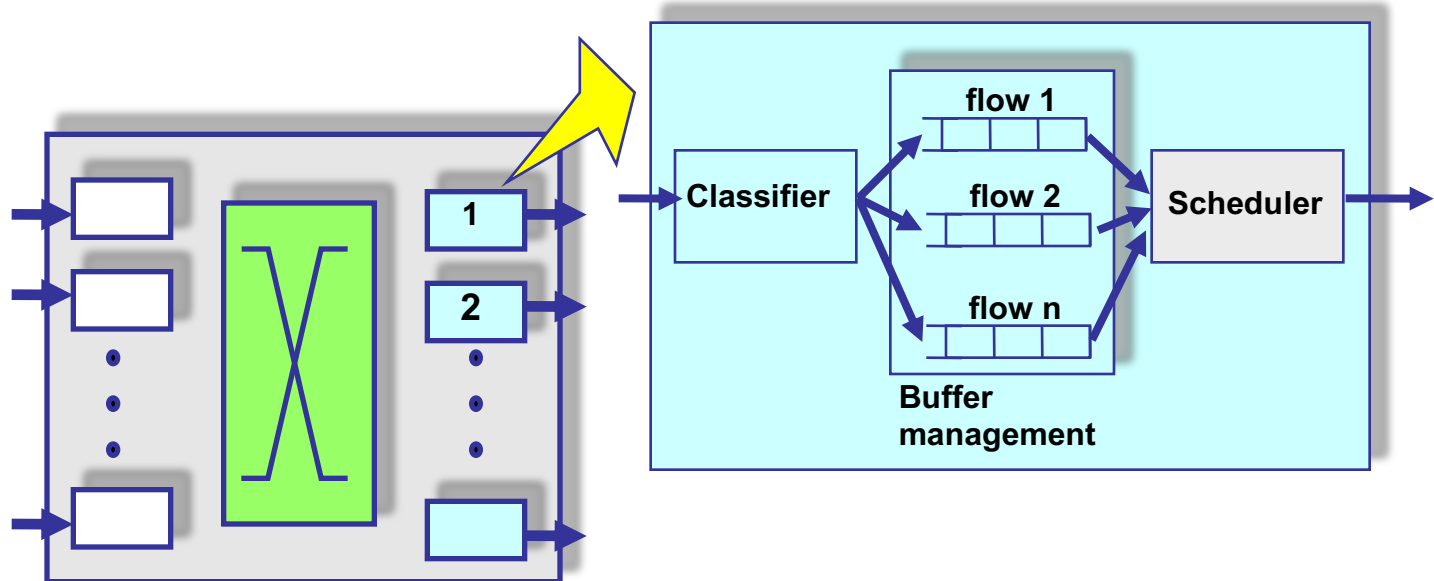
Lecture 21 Overview

- Buffer Management
 - ◆ FIFO
 - ◆ RED
- Traffic Policing/Scheduling



Key Router Challenges

- **Buffer management**: which packet to drop when?
 - ◆ We only have finite-length queues
- **Scheduling**: which packet to transmit next?





Basic Buffer Management

- FIFO + drop-tail
 - ◆ Simplest choice
 - ◆ Used widely in the Internet
- FIFO (first-in-first-out)
 - ◆ Implies single class of traffic
- Drop-tail
 - ◆ Arriving packets get dropped when queue is full regardless of flow or importance
- Important distinction:
 - ◆ FIFO: scheduling discipline
 - ◆ Drop-tail: drop policy



FIFO/Drop-Tail Problems

- Leaves responsibility of congestion control completely to the edges (e.g., TCP)
- Does not separate between different flows
- No policing: send more packets → get more service
- Synchronization: end hosts react to same events

Active Queue Management



- Design active router queue management to aid congestion control
- Why?
 - ◆ Router has unified view of queuing behavior
 - ◆ Routers see actual queue occupancy (distinguish queue delay and propagation delay)
 - ◆ Routers can decide on transient congestion, based on workload



Design Objectives

- Keep throughput high and delay low
 - ◆ High power (throughput/delay)
- Accommodate bursts
- Queue size should reflect ability to accept bursts rather than steady-state queuing
- Improve TCP performance with minimal hardware changes in router



Random Early Detection

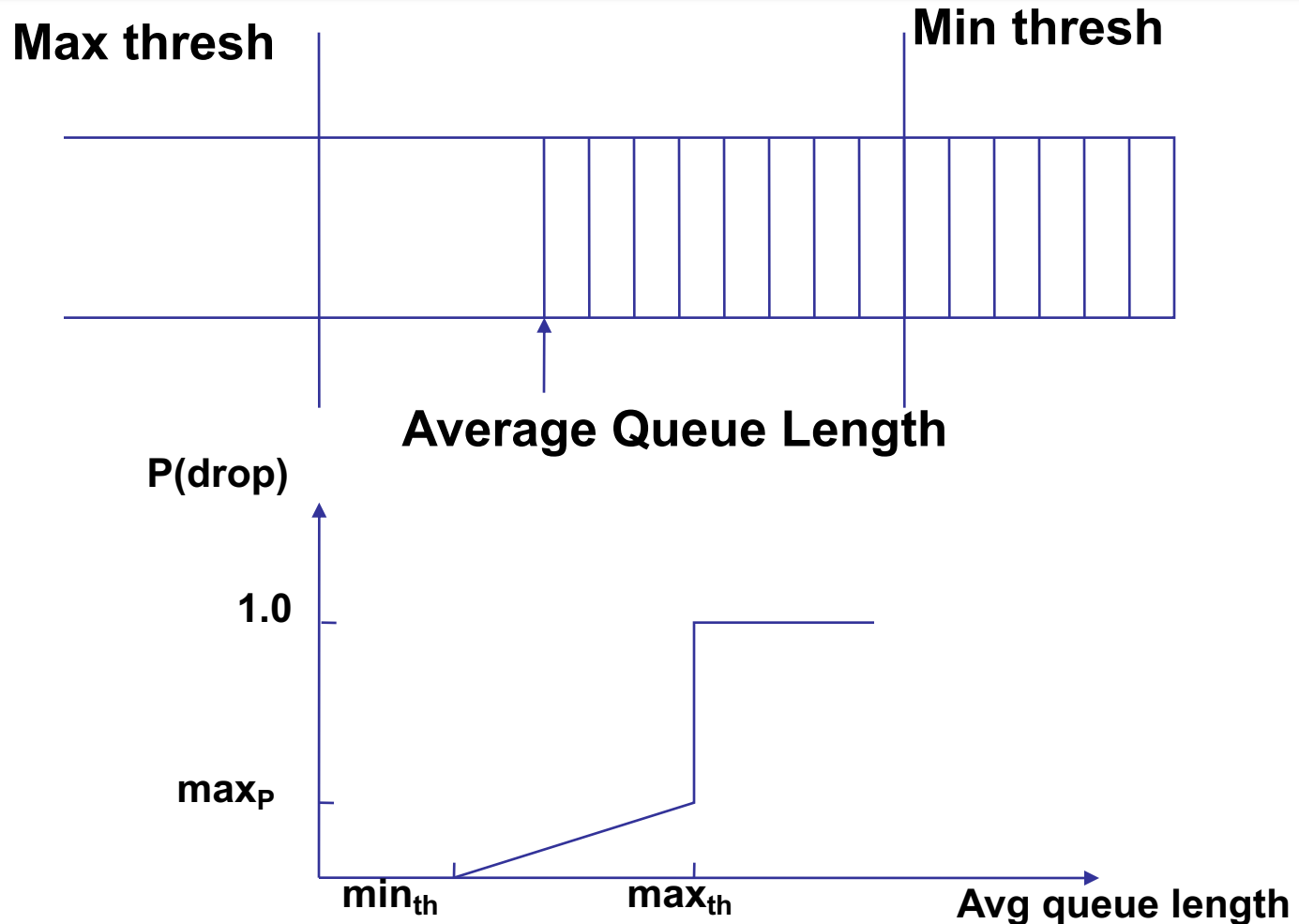
- Detect incipient congestion
- *Assume* hosts respond to lost packets
- Avoid window synchronization
 - ◆ Randomly mark packets
- Avoid bias against bursty traffic



RED Algorithm

- Maintain running average of queue length in router
- If $\text{avg} < \text{min}_{\text{th}}$ do nothing
 - ◆ Low queuing, send packets through
- If $\text{avg} > \text{max}_{\text{th}}$, drop packet
 - ◆ Protection from misbehaving sources
- Else drop/mark packet in a manner proportional to queue length
 - ◆ Notify sources of incipient congestion
 - ◆ Dropping vs Marking tradeoff (Explicit Congestion Notification)

RED Operation



Scheduling

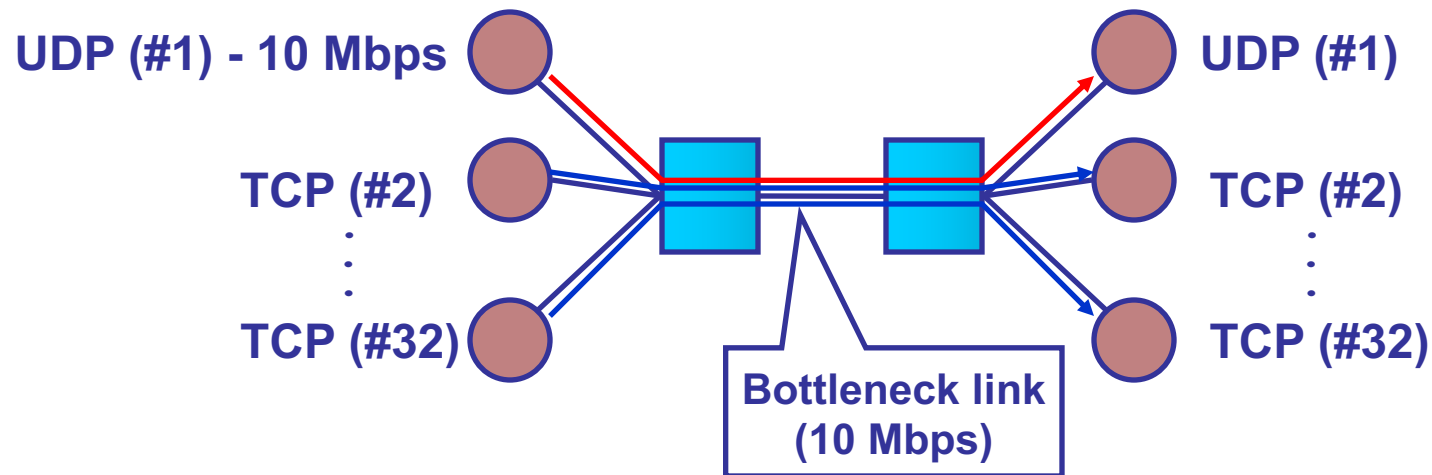


- So far we've done flow-based **traffic policing**
 - ◆ Limit the rate of one flow regardless of the load in the network
- In general, need **scheduling**
 - ◆ Dynamically allocate resources when multiple flows compete
 - ◆ Give each “flow” (or traffic class) own queue (at least theoretically)
- Weighted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



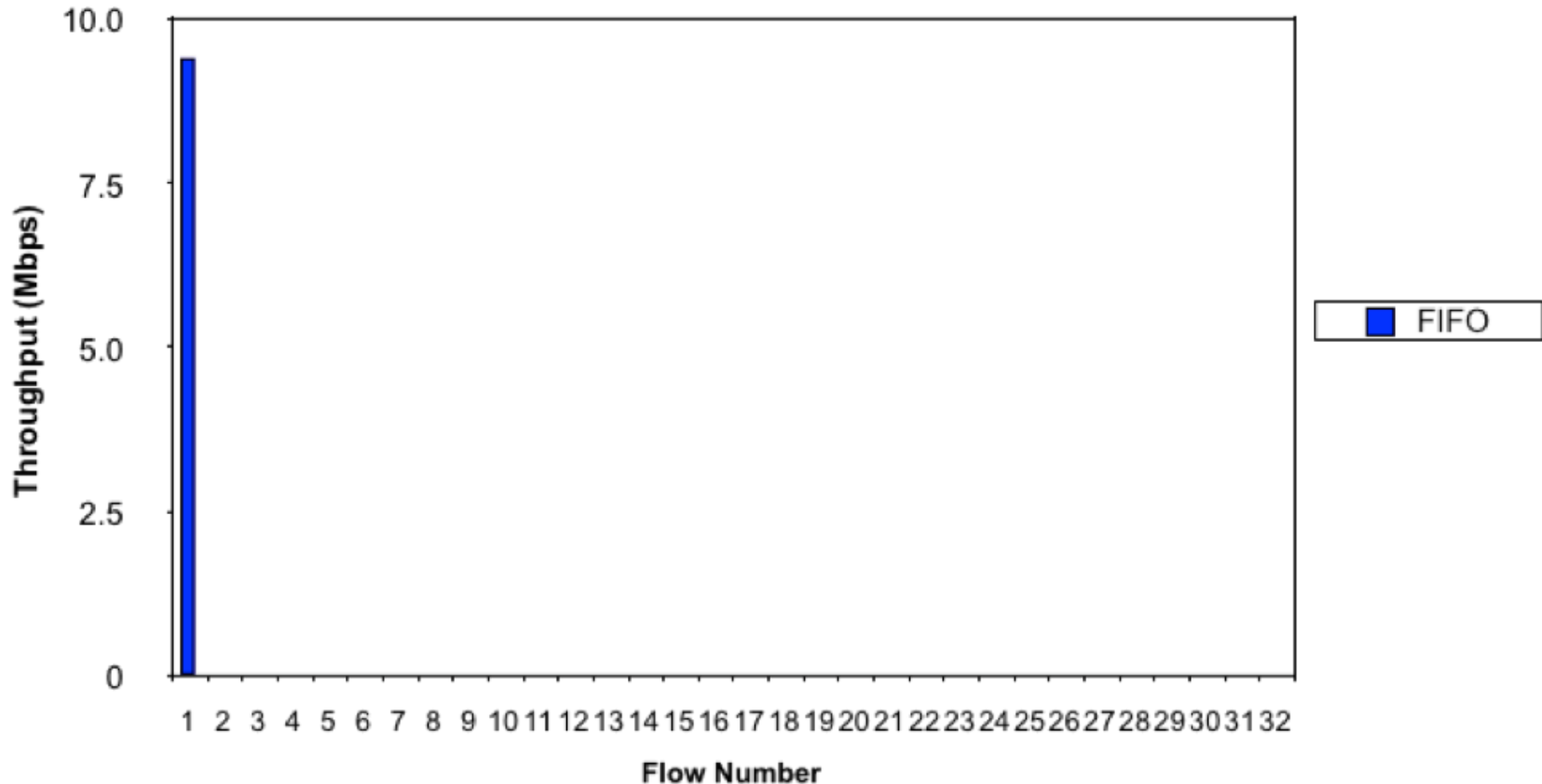
Our Previous Example

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

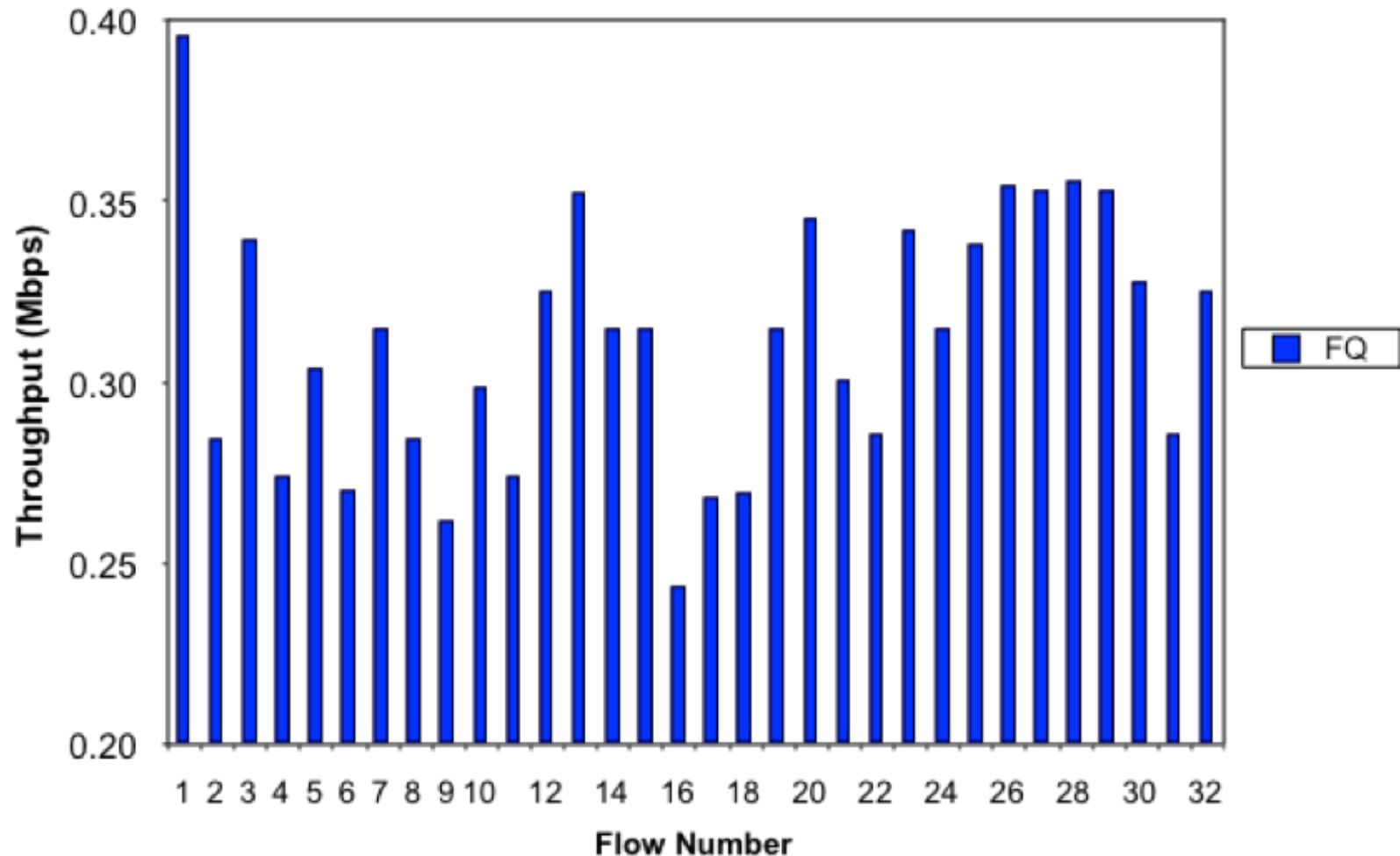




UDP vs. TCP w/FIFO

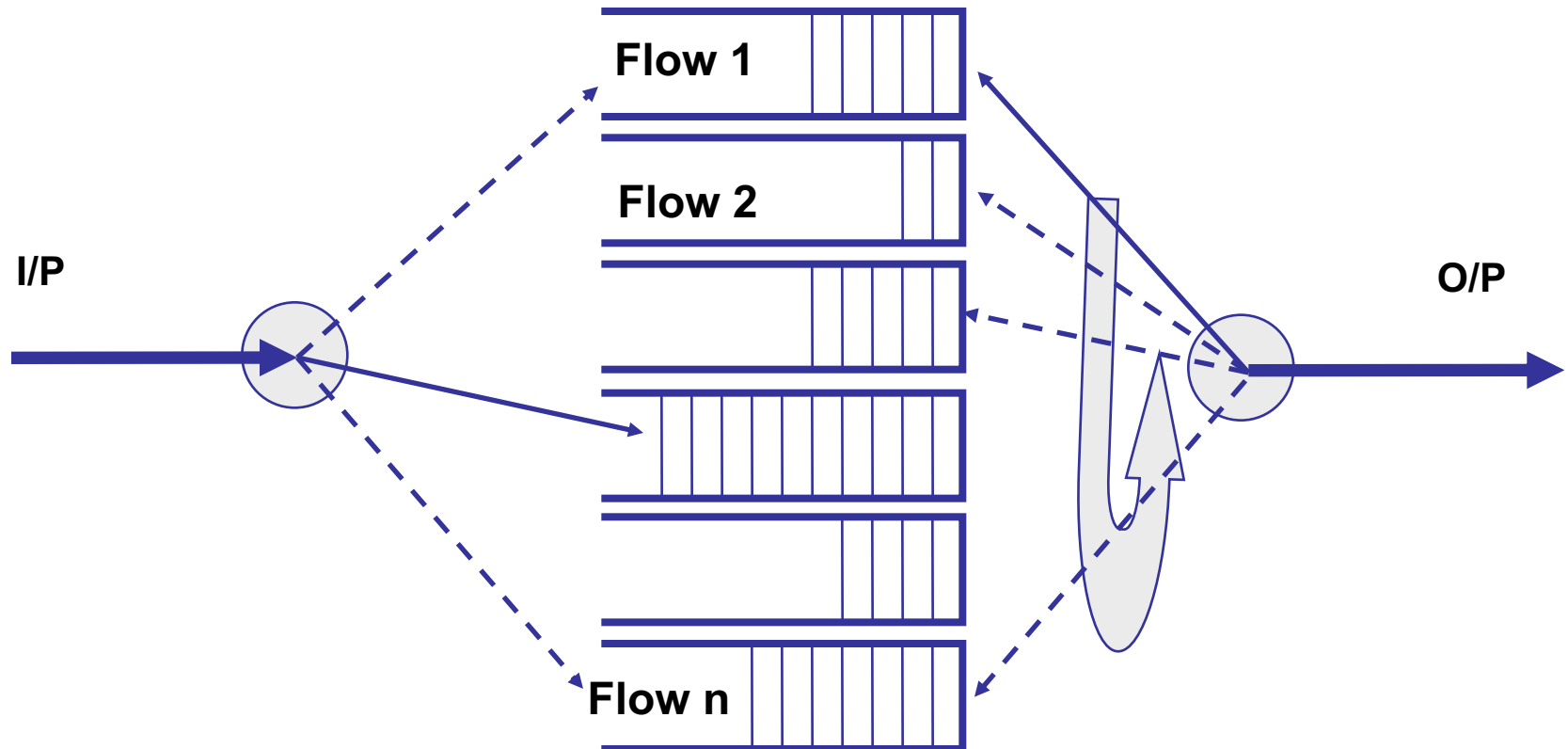


TCP vs. UDP w/Fair Queuing





(Weighted) Fair Queuing





Fair Queuing

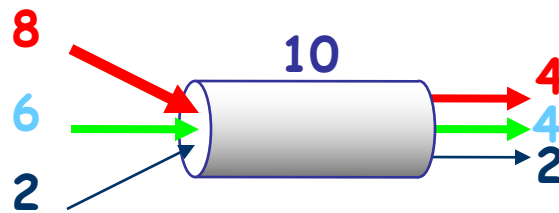
- Maintain a queue for each flow
 - ◆ What is a flow?
- Implements **max-min fairness**: each flow receives $\min(r_i, f)$, where
 - ◆ r_i – flow arrival rate
 - ◆ f – link fair rate (see next slide)
- **Weighted Fair Queuing** (WFQ) – associate a weight with each flow to divvy bandwidth up non-equally



Fair Rate Computation

- If link congested, compute f such that

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



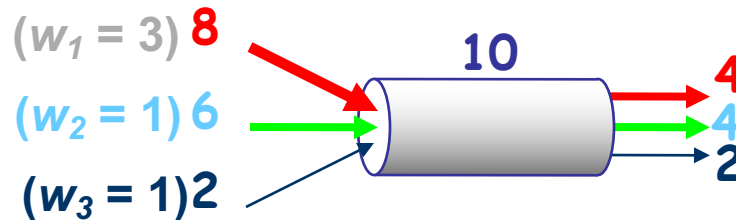
$f = 4$:
 $\min(8, 4) = 4$
 $\min(6, 4) = 4$
 $\min(2, 4) = 2$



Another Example

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

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



$f = 2$:

$$\begin{aligned} \min(8, 2 \times 3) &= 6 \\ \min(6, 2 \times 1) &= 2 \\ \min(2, 2 \times 1) &= 2 \end{aligned}$$

Flow i is guaranteed to be allocated a rate $\geq w_i \cdot C / (\sum_k w_k)$



If $\sum_k w_k \leq C$, flow i is guaranteed to be allocated a rate $\geq w_i$

Fluid Flow

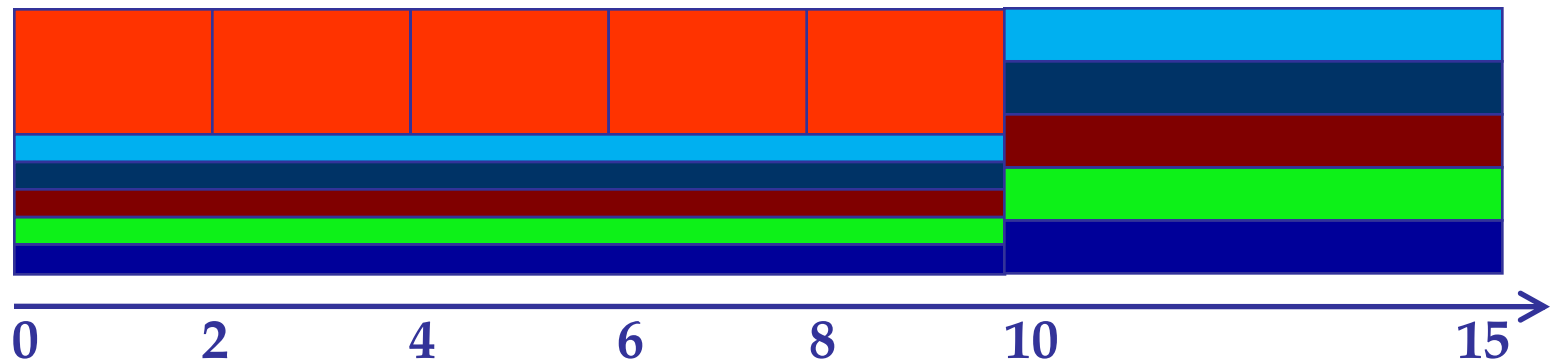
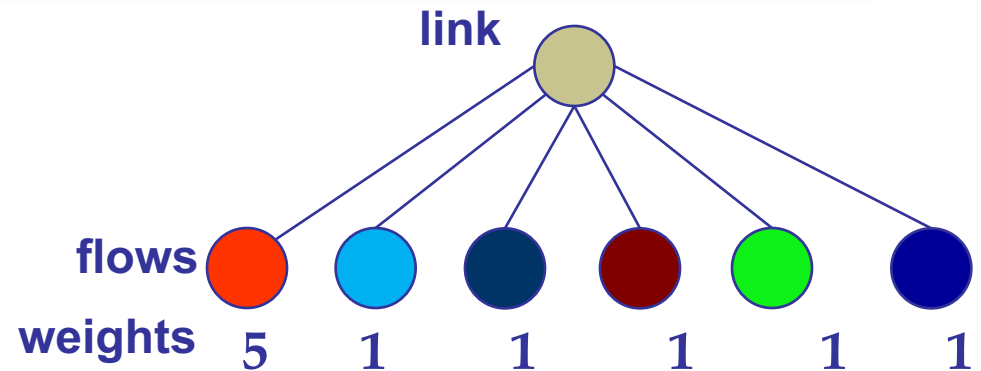


- Flows can be served one bit at a time
- WFQ can be implemented using bit-by-bit weighted round robin
 - ◆ During each round from each flow that has data to send, send a number of bits equal to the flow's weight



Fluid Flow Example

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



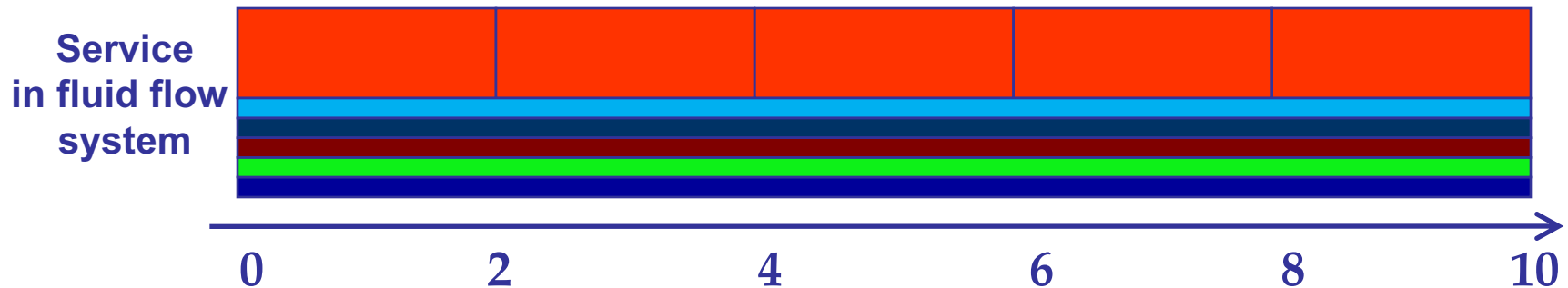
Packet-Based Implementation



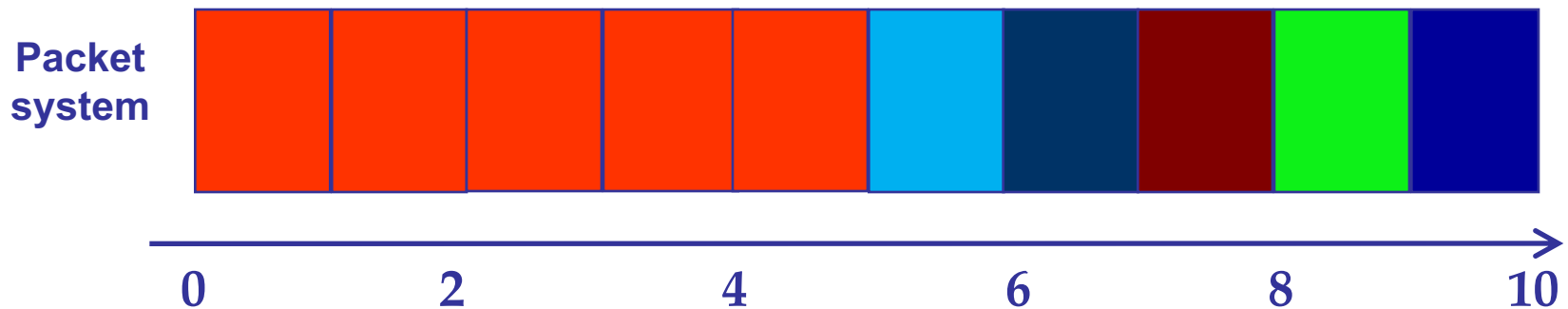
- Packet (Real) system: packet transmission cannot be preempted. Why?
- Solution: serve packets in the order in which they would have finished being transmitted in the fluid flow system



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