Class Meeting, Lectures 7 & 8: Buffer Management and Packet Scheduling

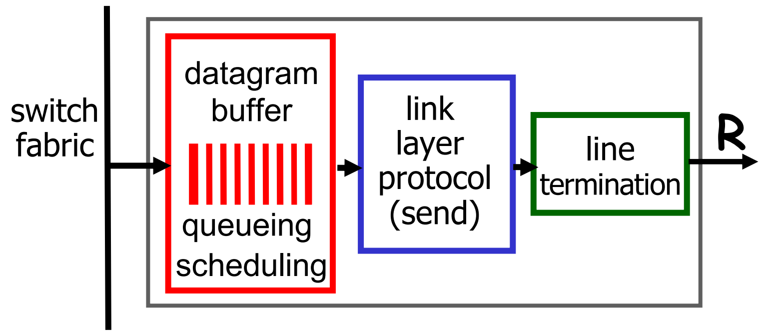
Kyle Jamieson

COS 461: Computer Networks

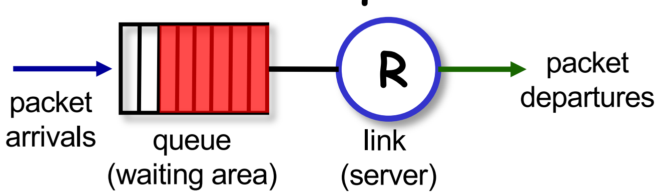
Buffer Management
(a.k.a., Queue Management)

- **Problem: Coping with excess load**

Context: Router line card



Abstraction: queue



Buffer management:

- **drop:** which packet to add, drop when buffers are full
- **mark:** which packets to mark to signal congestion)

How much buffering?

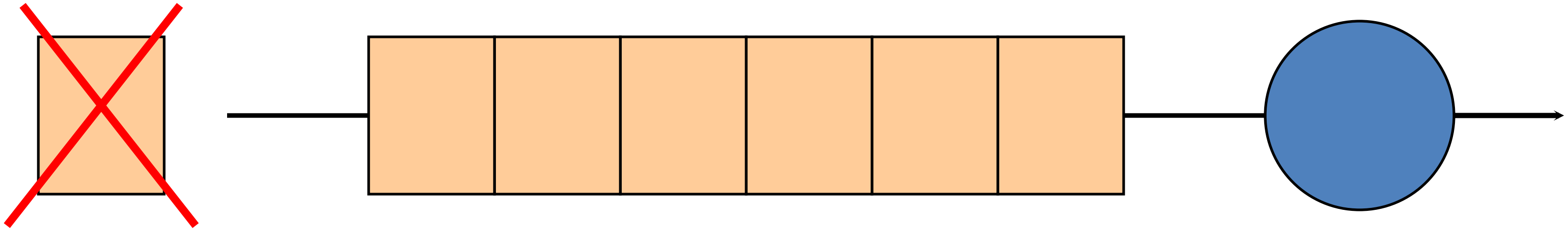
- RFC 3439 guidance circa 2002: average buffering equal to "typical" RTT (say 250 msec) times link capacity C
  - e.g., C = 10 Gbps link → 2.5 Gbit buffer

- More recent recommendation: with N flows, buffering equal to:

$$\frac{RTT \cdot C}{\sqrt{N}}$$

- But too much buffering increases delays (esp. in home routers)
  - Long RTTs: poor performance, esp. for realtime apps
  - Recall delay-based congestion control: "keep bottleneck link just full enough (busy) but no fuller"

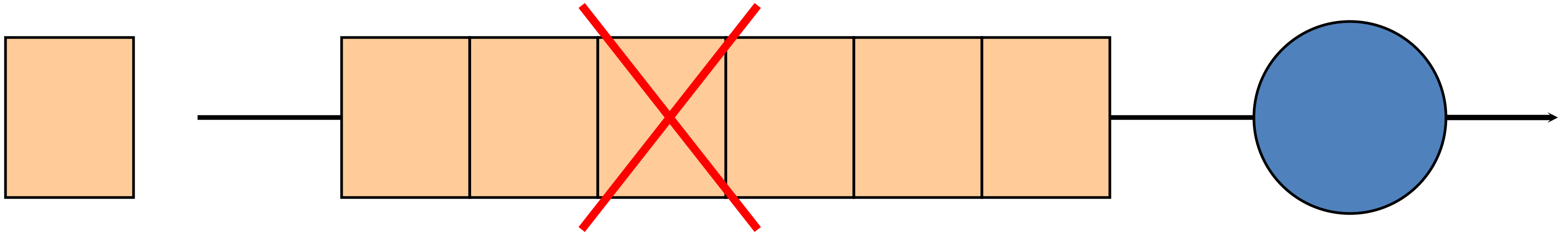
Drop-Tail Queue Management



- First-In First-Out (FIFO) queuing mechanism
- Drop packet at the tail if & when queue overflows

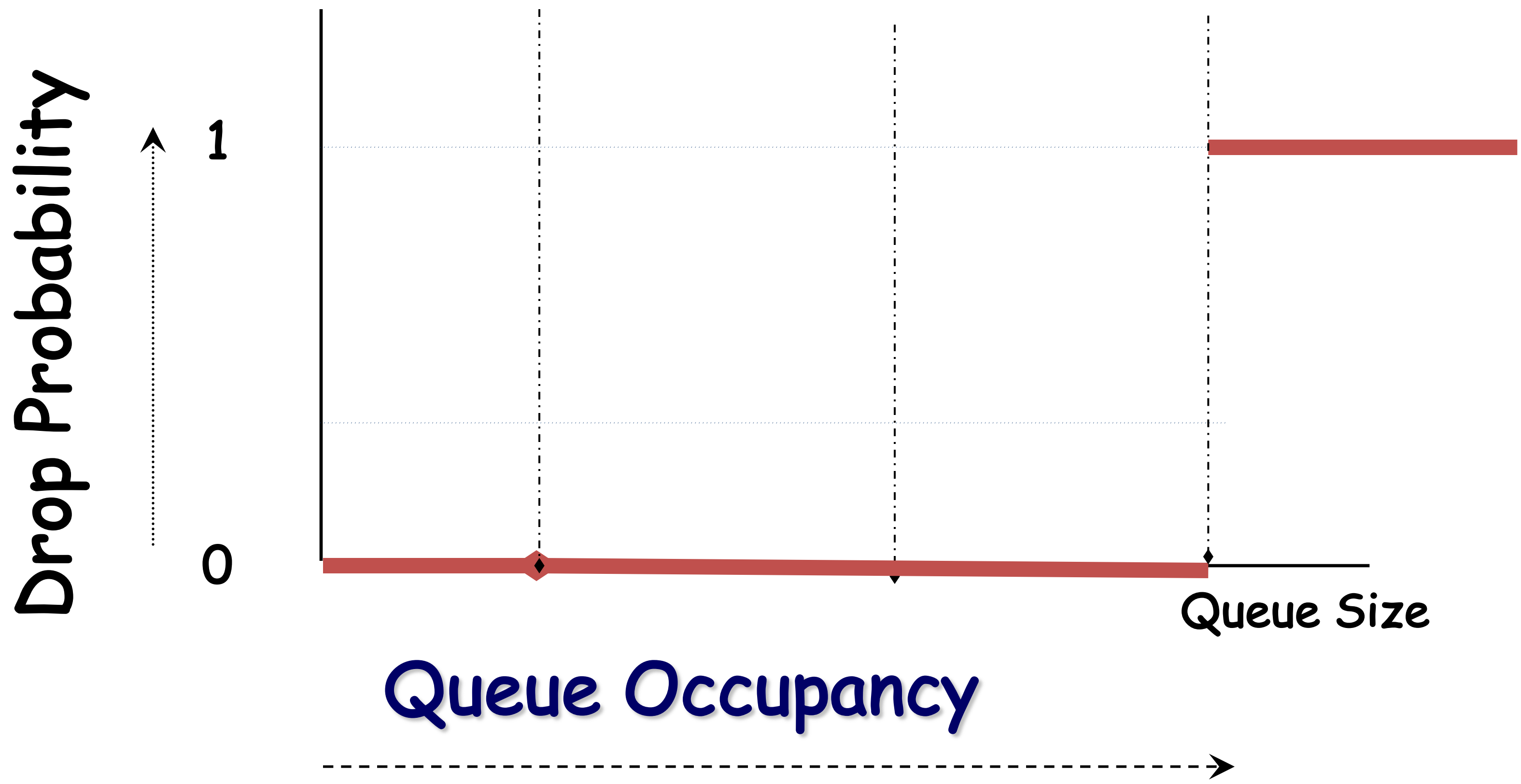
- Introduces global synchronization when packets are dropped from several connections.

Random Drop Queue Management

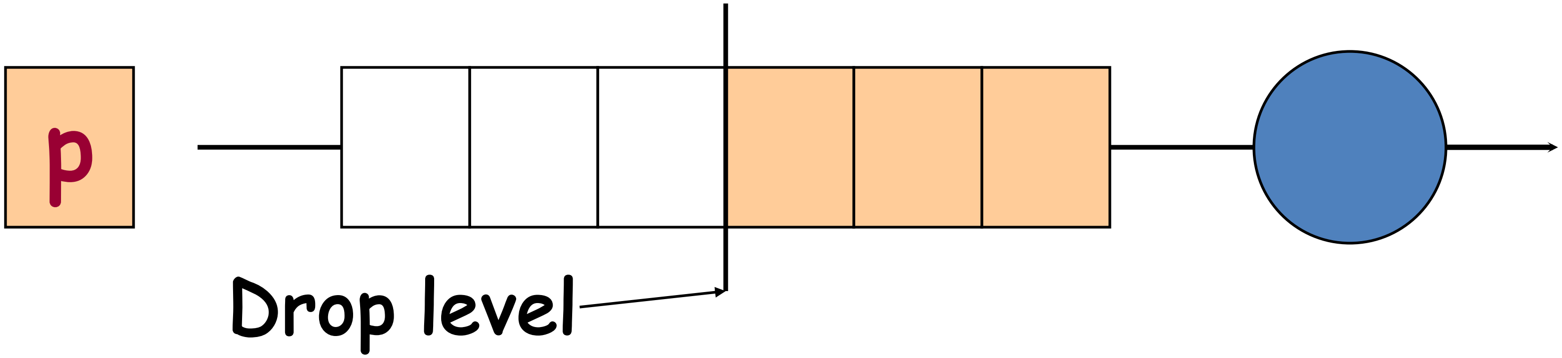


When a packet arrives and the queue is full, randomly choose a packet in the queue to drop.

Drop-Tail/Random Drop
Queue Management



Random Early Drop Router



- If queue length exceeds *drop level*, then router drops each arriving packet w/probability *p*
  - Reduces global flow synchronization
  - Does not control misbehaving users (UDP flows)

RED Algorithm (simplified)

```
for each packet arrival:
    calculate the average queue size avg
    if min_th ≤ avg < max_th
        calculate the drop probability p
        with probability p:
            mark the arriving packet
    else if max_th ≤ avg
        mark the arriving packet
    else ( avg < min_th ):
        do nothing: queue the arriving packet
```

avg - Average Queue Length

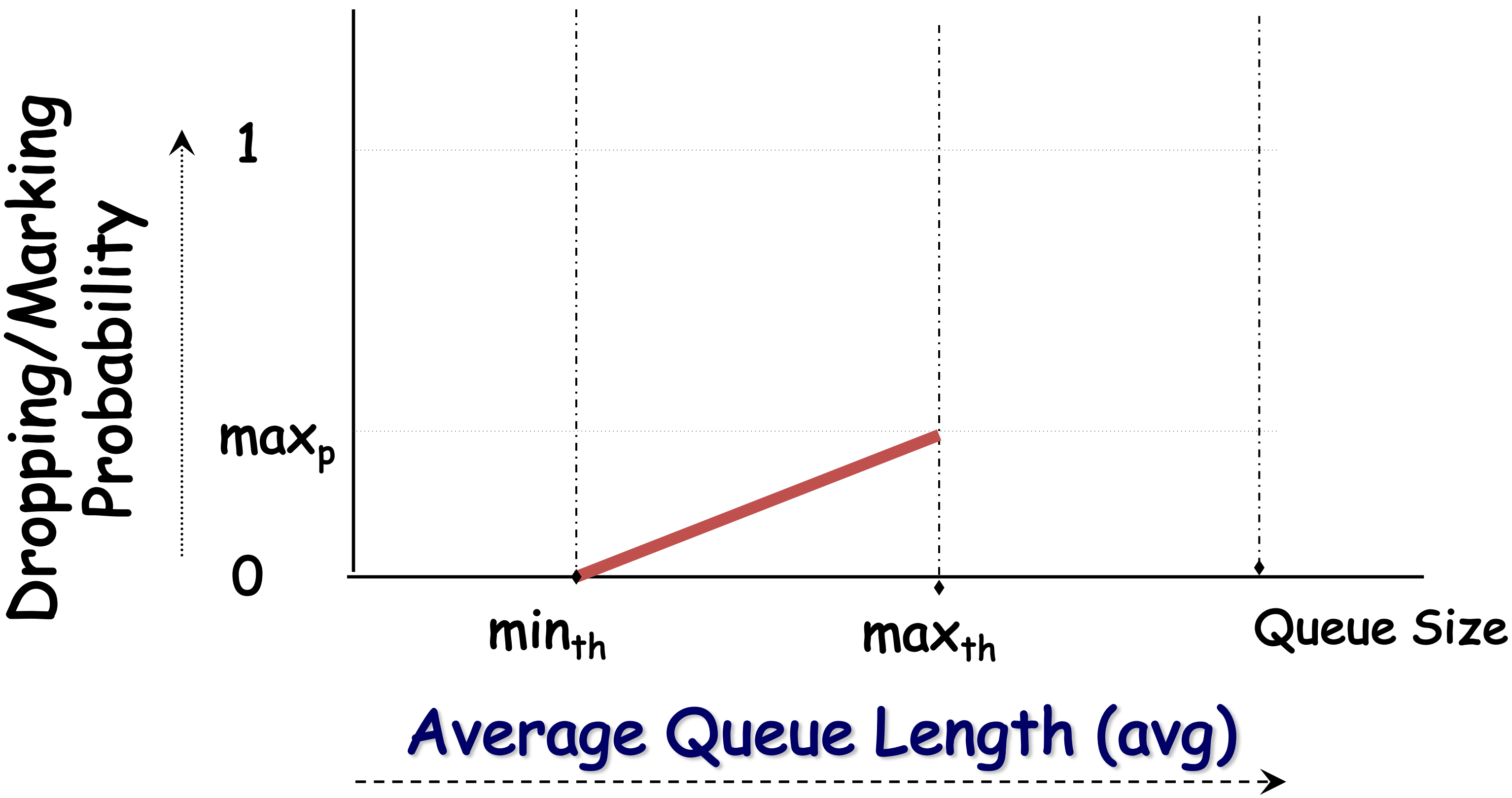
$$avg = (1 - w_q) \times avg + w_q \times q$$

where q is the newly-measured queue length.

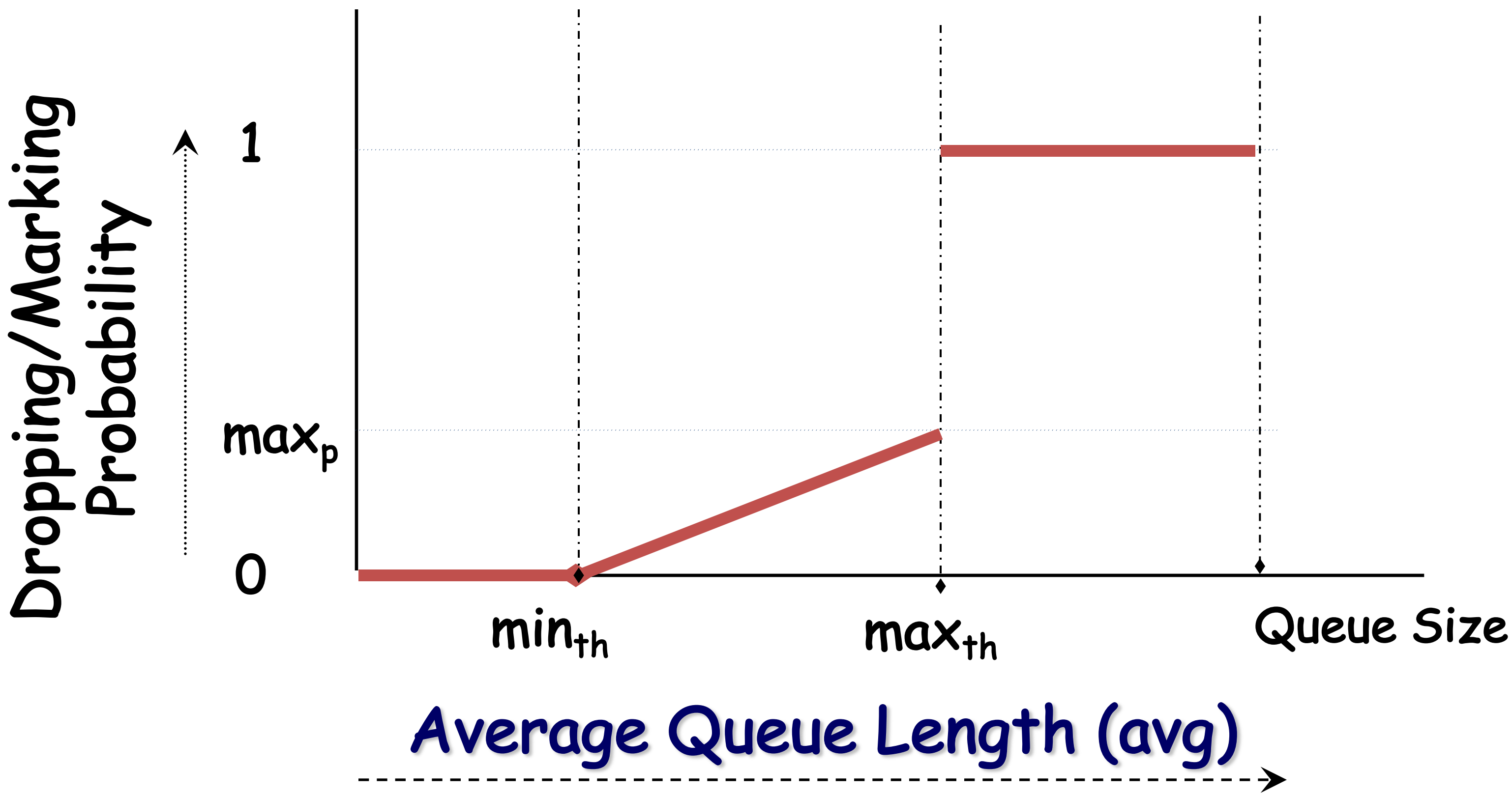
Exponential Weighted Moving Average (EWMA): short-term increases in queue size (bursty traffic, transient congestion) don't significantly increase avg

Calculating RED *Drop Probability p*

$$p = max_p \times (avg - min_{th})/(max_{th} - min_{th})$$



Putting it all together: RED/ECN Router Mechanism



Problems With RED

- Hard to get **tunable parameters** just right
  - **How early** to start dropping packets?
  - **What slope** for increase in drop probability?
  - **What time scale** for averaging queue length?
  - How about the upper threshold?

- RED has mixed adoption in practice
  - If parameters aren't set right, RED doesn't help

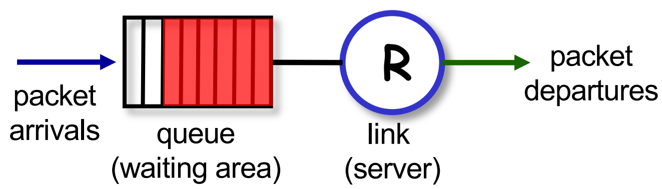
Deciding which packet to send next on a link

PACKET SCHEDULING

Packet Scheduling: FCFS

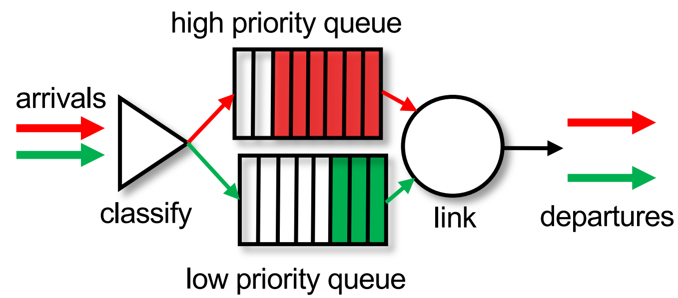
- First-Come, First-Served (FCFS): packets transmitted in order of arrival to output port
  - Also known as: First-in-first-out (FIFO)
  - Real world examples

Abstraction: queue



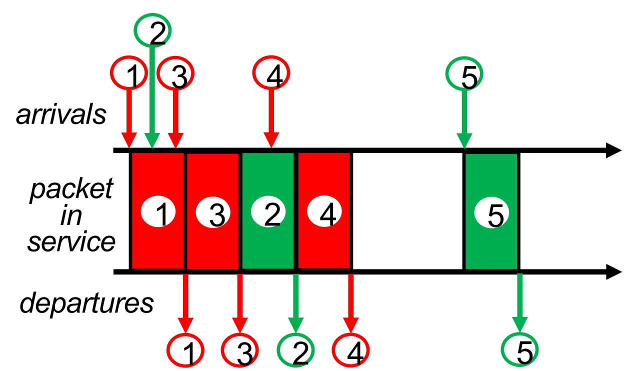
Scheduling policies: priority

- ***Priority scheduling:*** arriving traffic classified, queued by class
  - any header fields can be used for classification



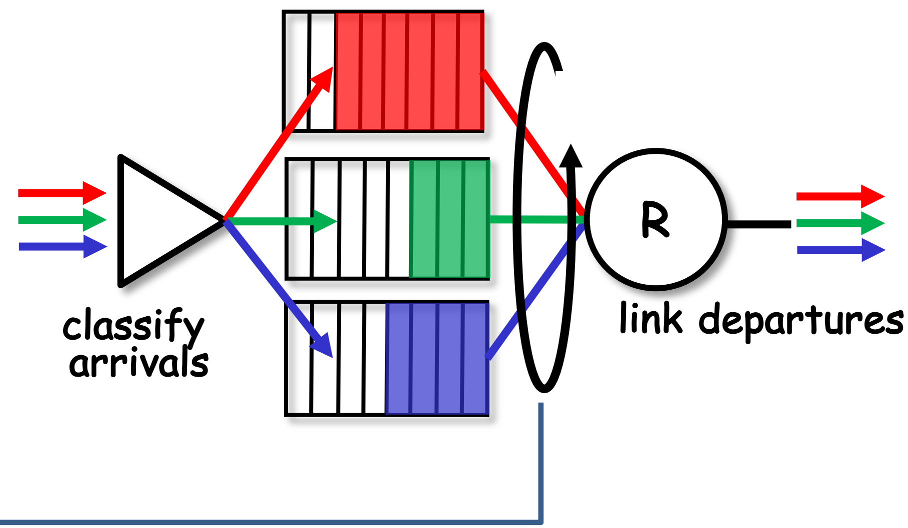
- Rule: Send packet from highest priority queue w/waiting packets

  - (FCFS within priority class)



Scheduling policies: round robin

- Round Robin (RR) scheduling:
- arriving traffic classified, queued by class
  - any header fields can be used for classification



- Output cycles thru class queues, sending one complete packet from each class (if available) in turn

Scheduling policies: weighted fair queueing

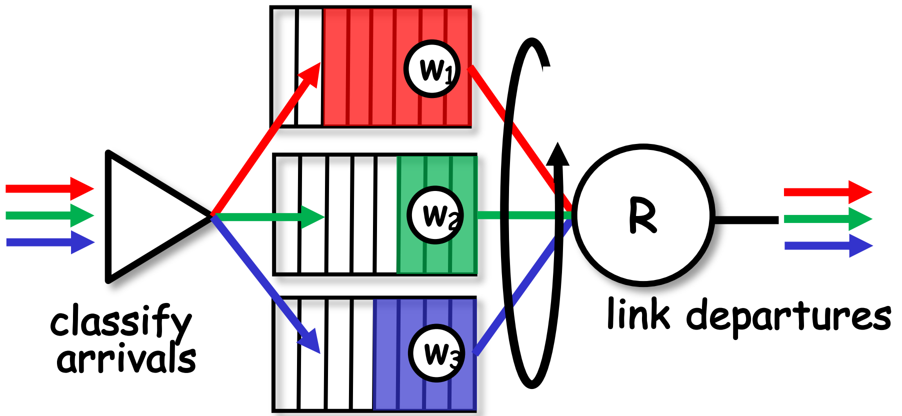
Weighted Fair Queuing (WFQ):

- generalized Round Robin

  - each class, i, has weight, w_i, and gets weighted amount of service in each cycle:

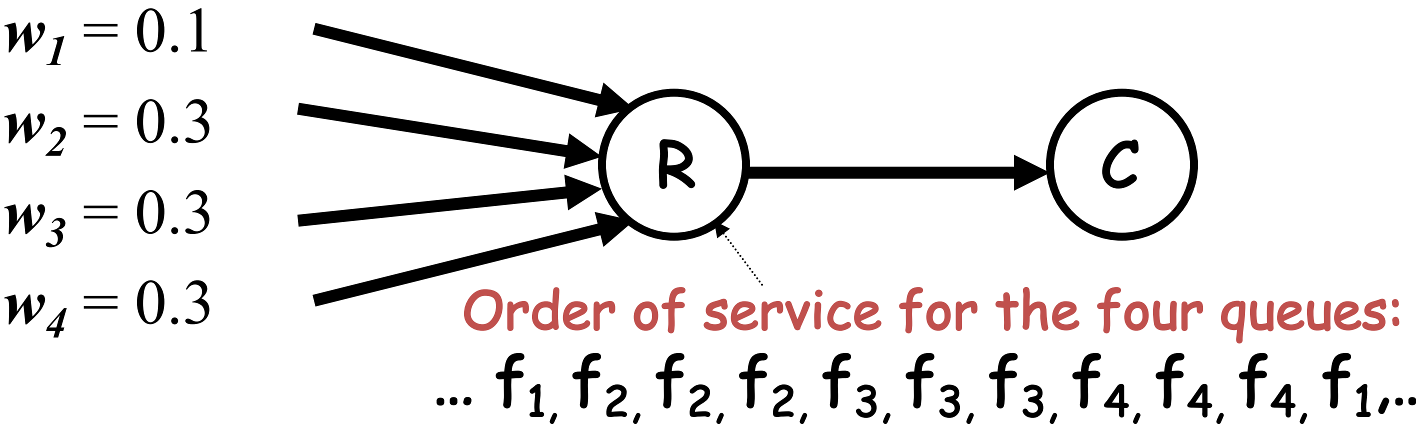
$$\frac{w_i}{\sum_j w_j}$$

  - minimum bandwidth guarantee (per-traffic-class)



Bit-by-Bit Weighted Fair Queuing

Flows allocated different rates by servicing <u>different number of bits</u> for each flow during each round.



…Like a fluid!

Packet vs. Bit-by-Bit "Fluid" System

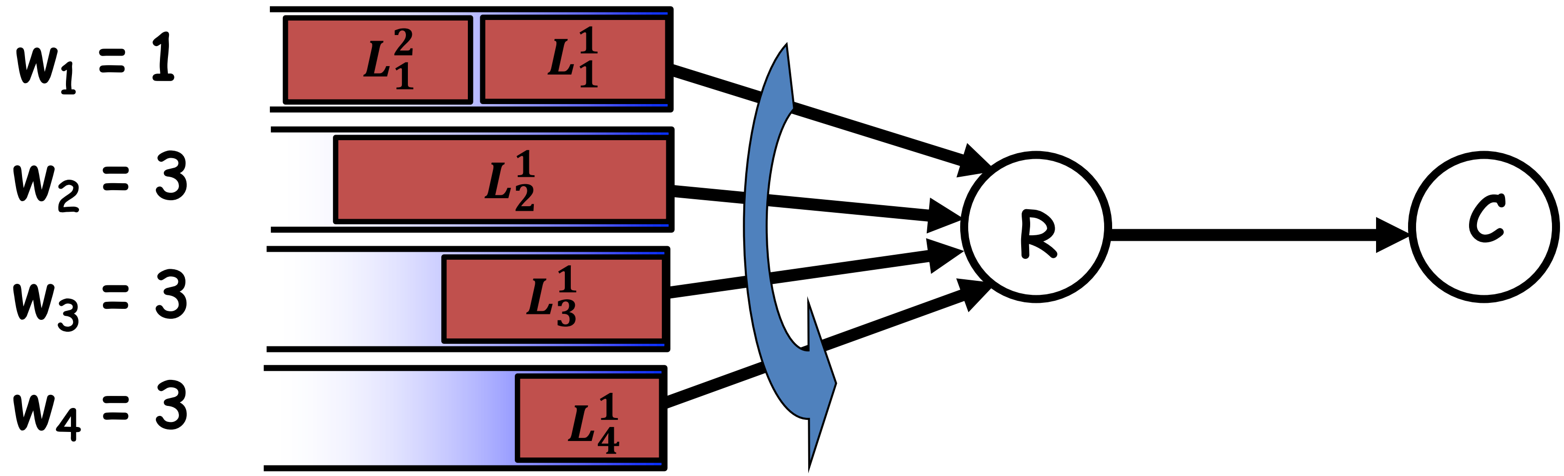
- Bit-by-bit FQ is not implementable:

  …In real packet-based systems:
  - One queue is served at any given time
  - Packet transmission cannot be preempted

- Goal: A packet scheme close to fluid system
  - Bound performance w.r.t. fluid system

Virtual Time Implementation of WFQ

L_i^k: length of k^{th} packet of flow i



Round – One complete cycle through all the queues, sending w_i bits per queue

Serve packets in order of their finishing round.

Next Up in 461

Next Class Meeting

Lectures 9 (Routing Algorithms) and 10 (Routing Convergence)

Precepts this Thursday and Friday:

Topics in Congestion Control