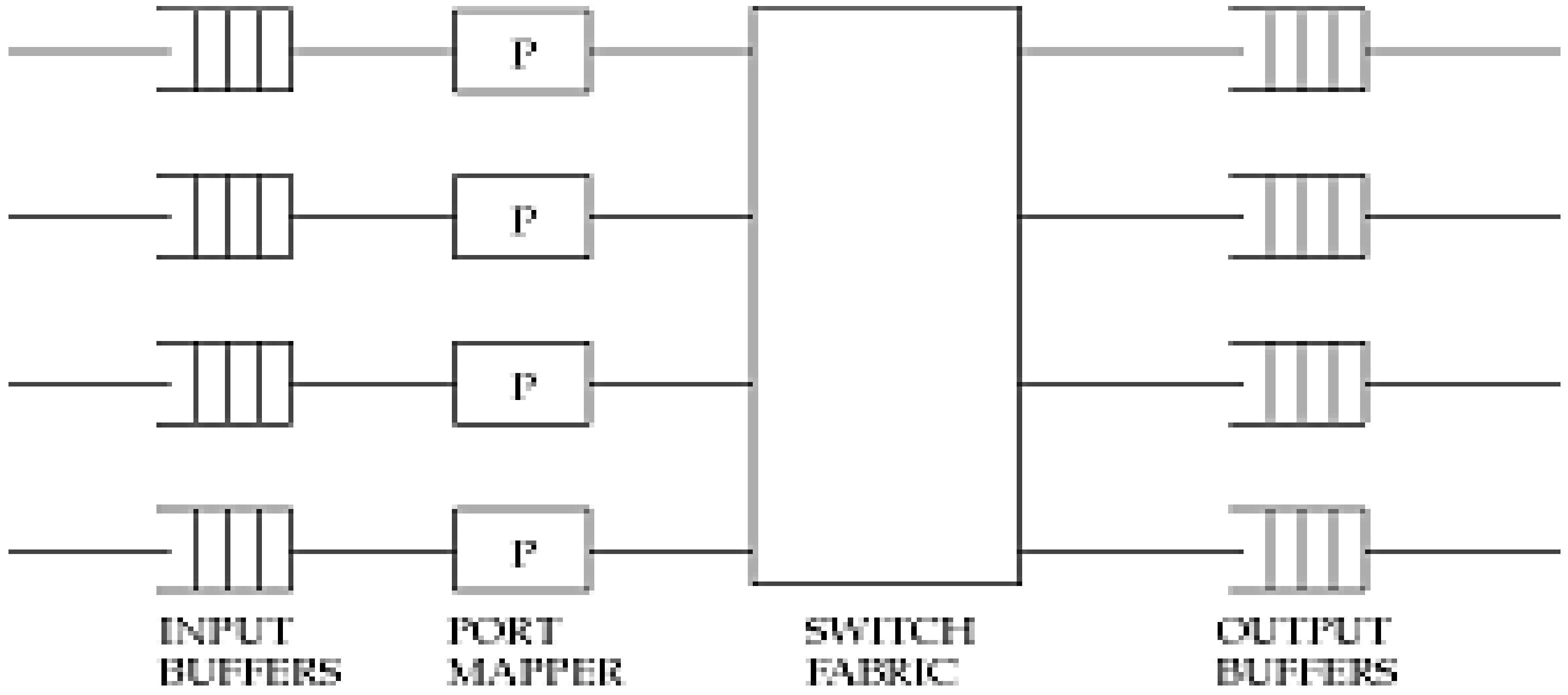
CS 348: Computer Networks

- Scheduling; 3rd Sept 2012

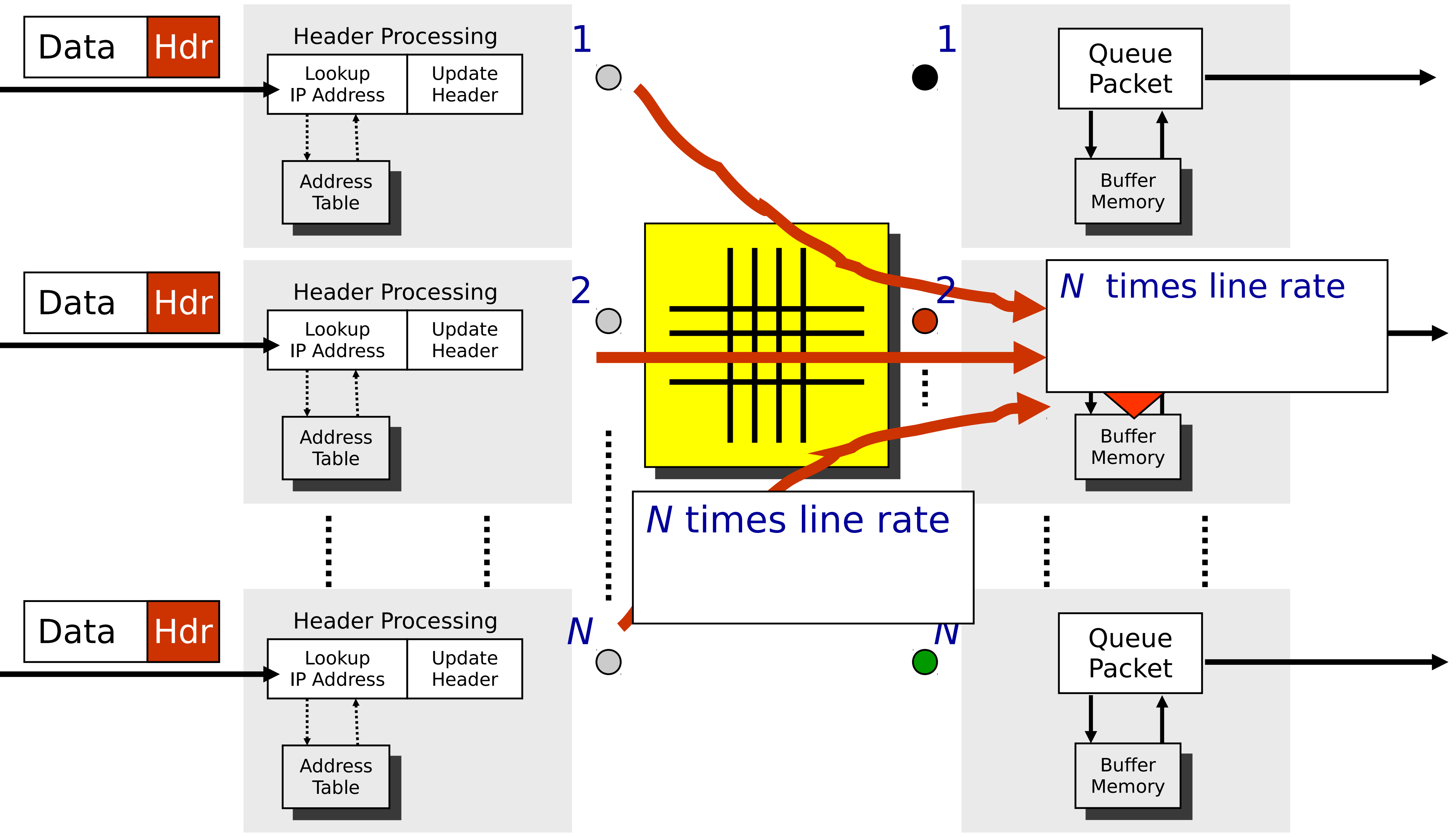
Instructor: Sridhar Iyer
IIT Bombay

Generic switch/router architecture



Latency: Time a switch/router takes to figure out where to forward an incoming data packet.

Generic Router Architecture



Blocking in routers (packet switches)

- Can have both internal and output blocking
  - Internal: no path to output.
  - Output: link unavailable.

- Unlike a circuit switch (telephone exchange), a packet switch cannot predict if packets will block.
- If packet is blocked, must either buffer or drop it.

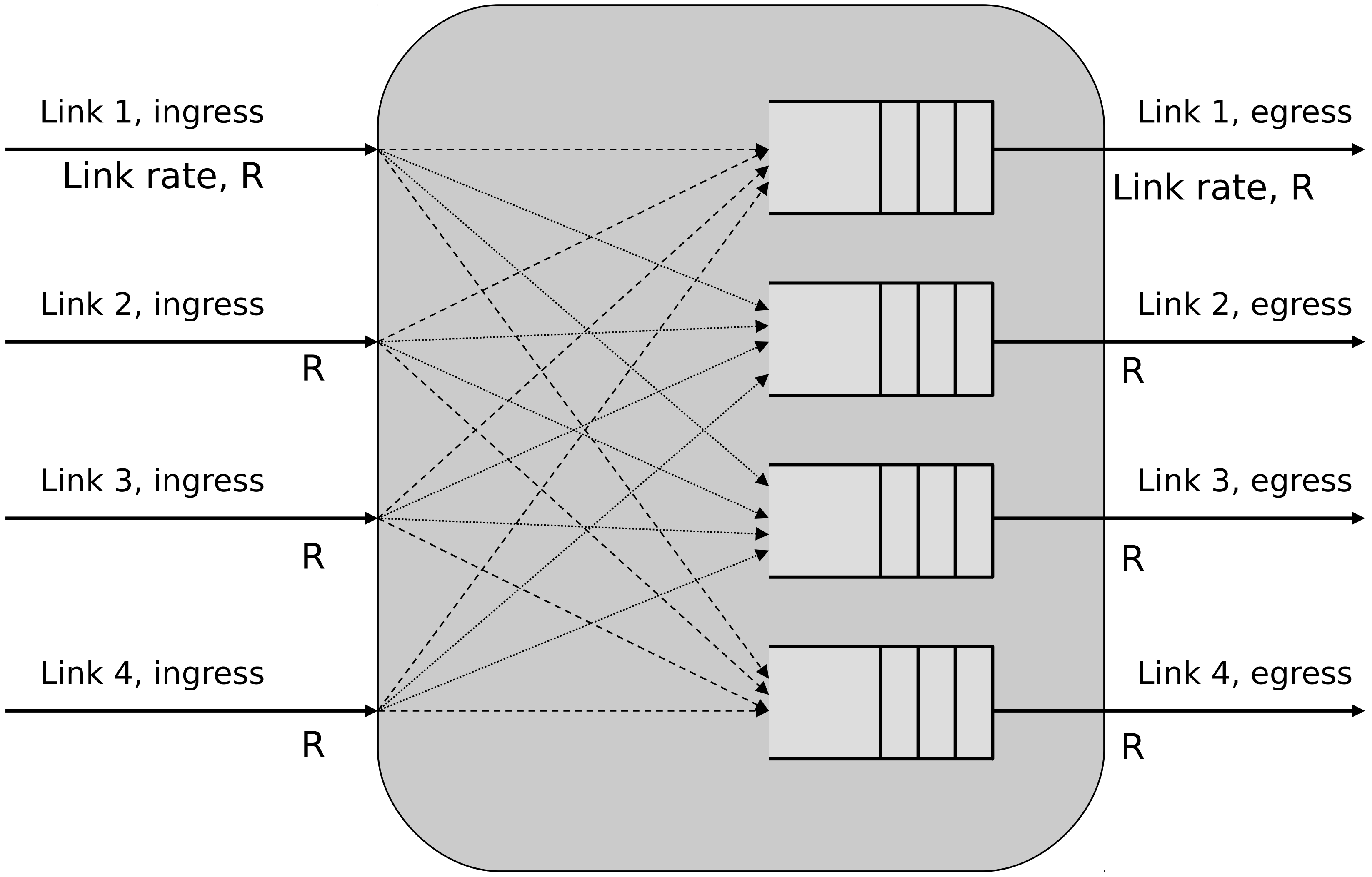
- Given that buffering is a solution to blocking, where should we place the buffer?
  - Input port and/or Switch fabric and/or Output port?

Dealing with blocking

- Match input rate to service rate
  - Over-provisioning: internal links much faster than inputs

- Buffering:
  - input port: *head of line blocking*
    - if packet at head of the queue does not have a path to output, packets behind it are stuck even if their paths are clear.
  - in the fabric: circuitry is too expensive.
  - output port: most commonly used mechanism.

Output queued switch



Buffers need to run faster than link rate.

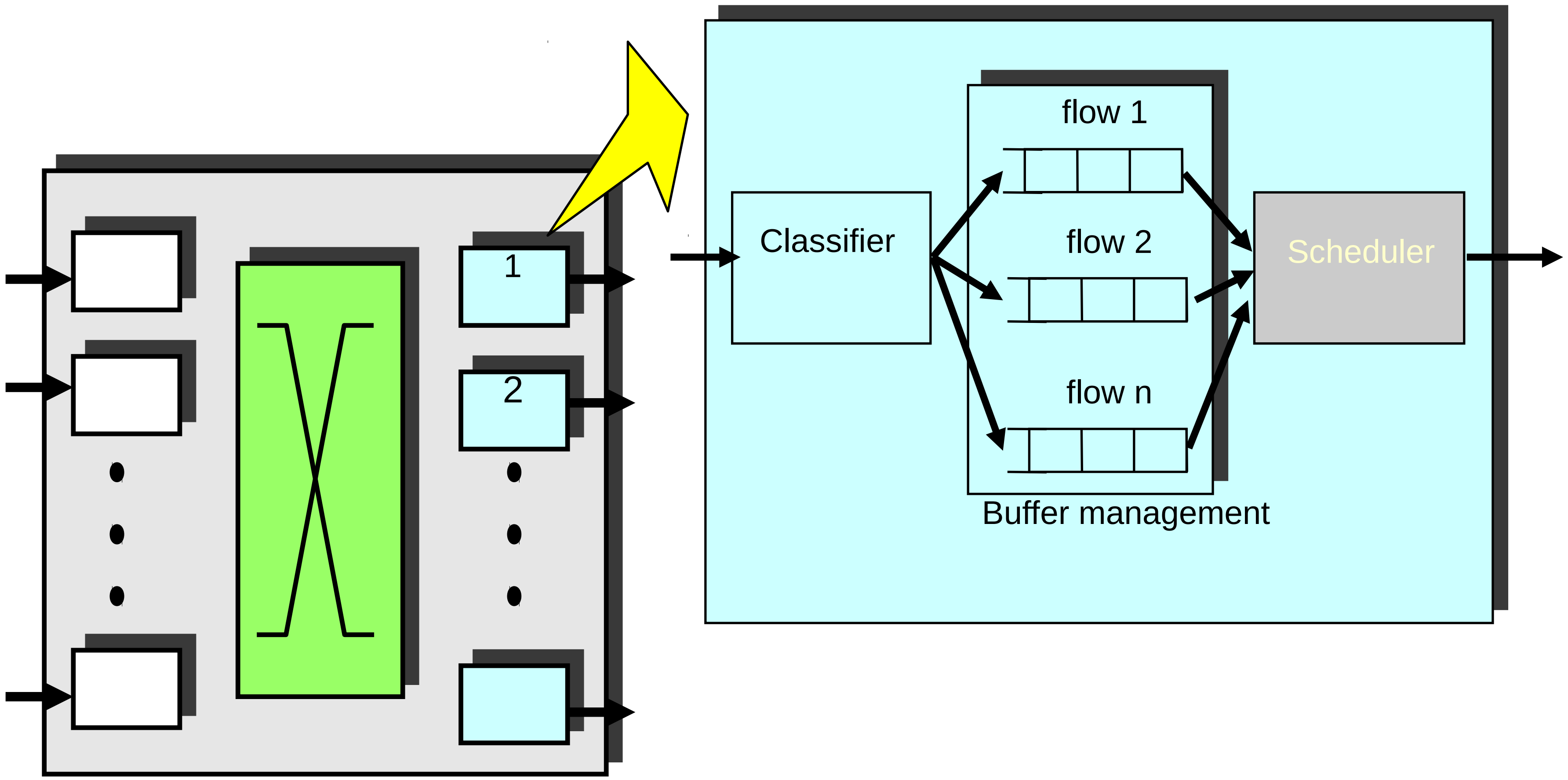
Questions to consider

- Suppose the packets arriving at all the input links have to go out on the same output link. Now one of the inputs is sending many more packets and that too more frequently than the others.
  - What will happen at the output buffer?

- Suppose the packets arriving at one of the input links need to be given a higher priority than the others.
    - higher share of output bandwidth (2 of A's packets are sent for every one of B).
    - lesser time spent in the buffer (A's pkts jump the queue).
  - What needs to be done to enable this?

Packet scheduling

- Decide when and what packet to send on output link
  - Usually implemented at output port



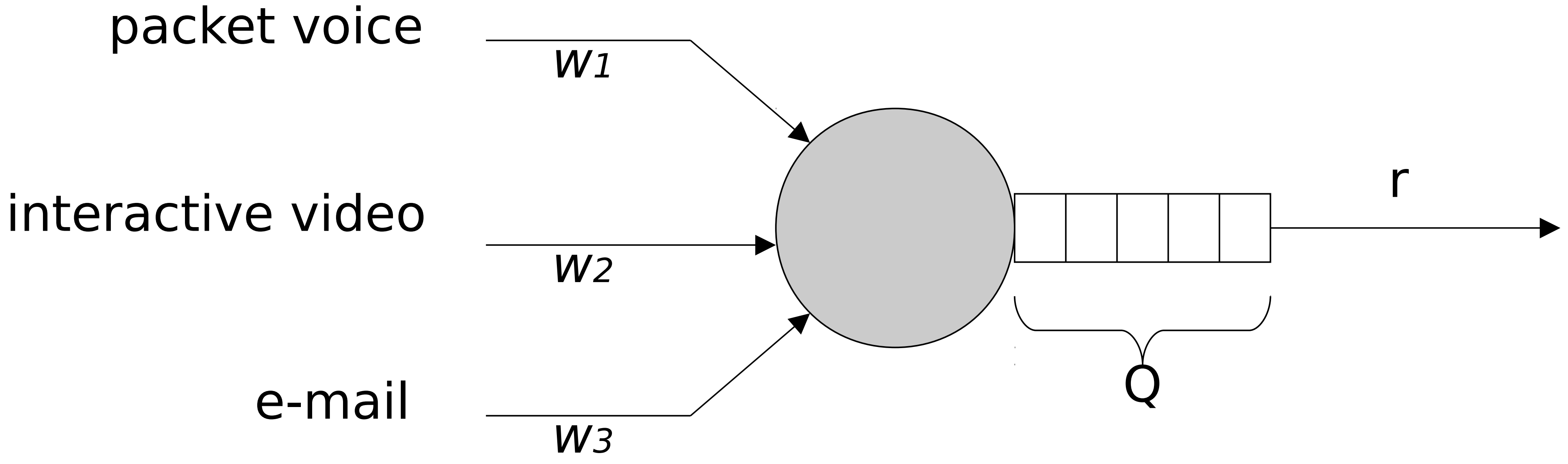
Scheduling Objectives

- Scheduling is the key to fairly sharing resources and providing performance guarantees.

- A scheduling discipline does two things:
  - decides service order.
  - manages queues of service requests.

Scheduling disciplines

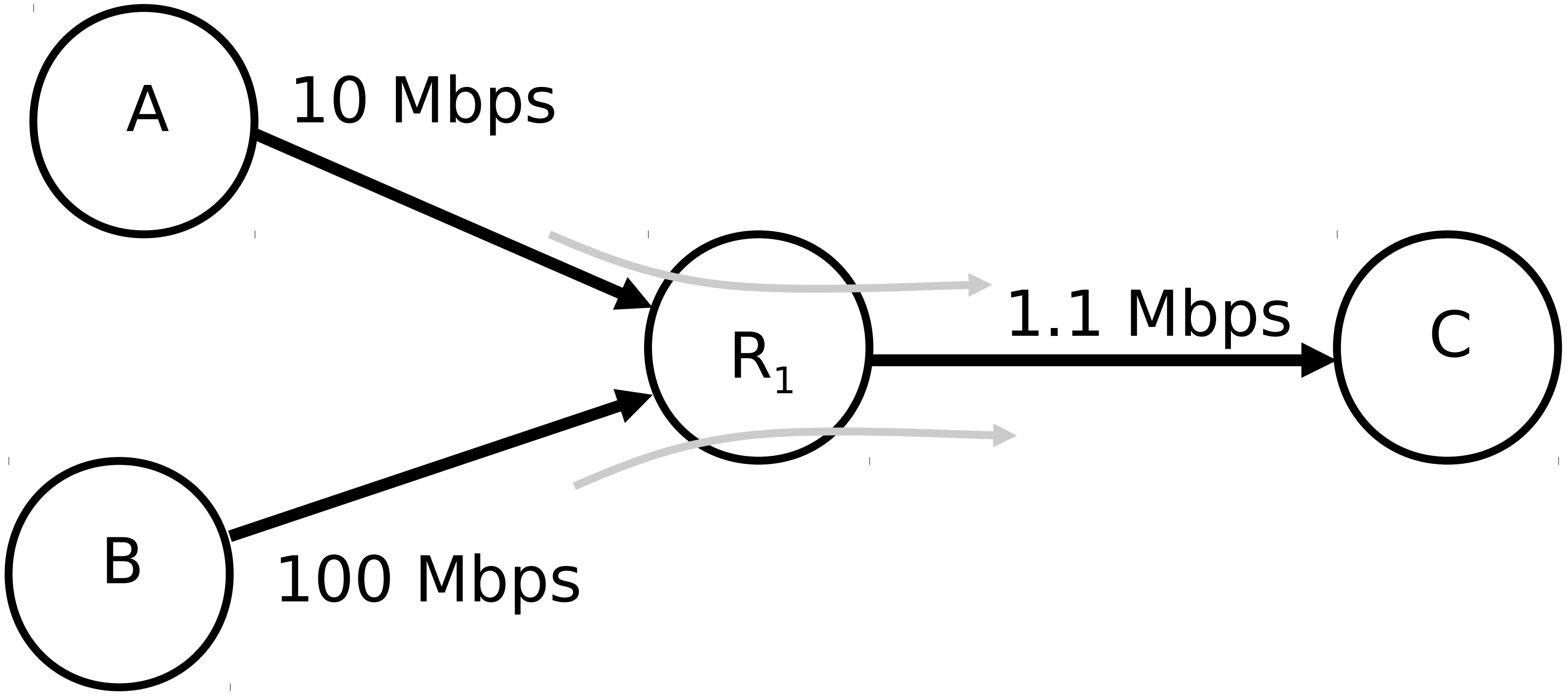
What could be some approaches to scheduling?



Problem with First-In-First-Out (FIFO)

- In order to maximize its chances, a source has incentive to maximize the rate at which it transmits.
- When many flows pass through it, a FIFO queue is “unfair” – it favors the most greedy flow.
- It is hard to control the delay of packets through a network of FIFO queues.

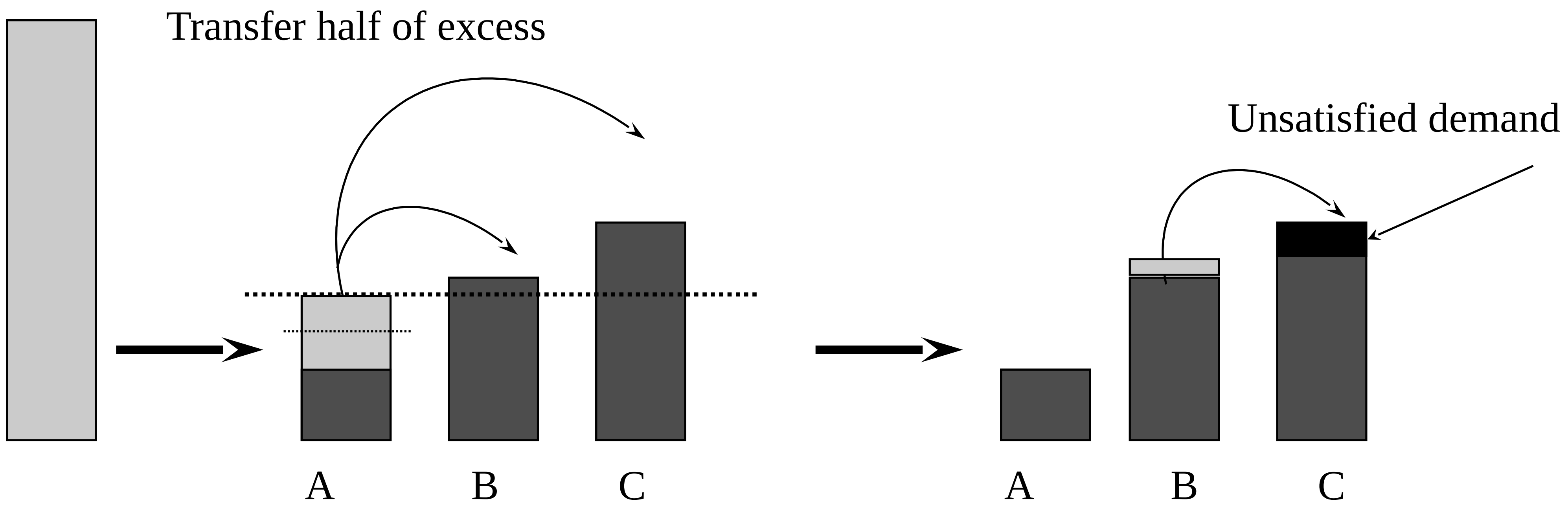
Fairness



What is the “fair” allocation:
(0.55Mbps, 0.55Mbps) or (0.1Mbps, 1Mbps)?

Max-Min Fairness

- An allocation is fair if it satisfies *max-min fairness*
  - each connection gets no more than what it wants
  - the excess, if any, is equally shared

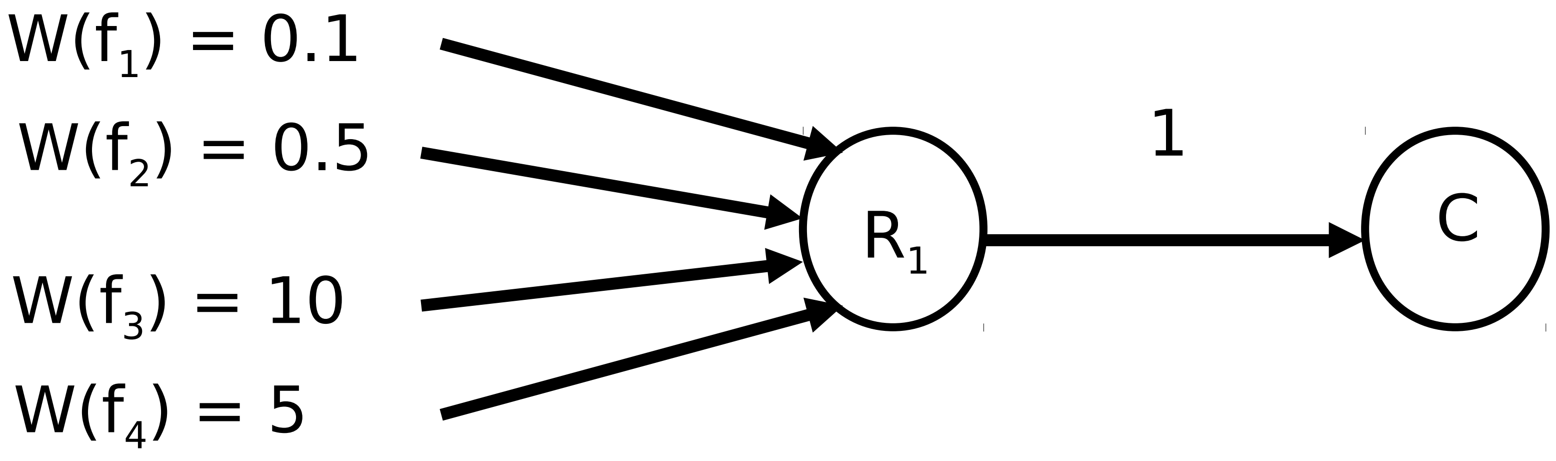


Max-Min Fairness

- *N* flows share a link of rate *C*.
- Flow *f* wishes to send at rate W(*f*), and is allocated rate R(*f*).

1. Pick the flow, f, with the smallest W(f).
2. If W(f) < C/N, then set R(f) = W(f).
3. If W(f) > C/N, then set R(f) = C/N.
4. Set N = N – 1. C = C – R(f).
5. If N > 0 goto 1.

Max-Min Fairness: example



Round 1: Set $R(f_1) = 0.1$

Round 2: Set $R(f_2) = 0.9/3 = 0.3$

Round 3: Set $R(f_4) = 0.6/2 = 0.3$

Round 4: Set $R(f_3) = 0.3/1 = 0.3$

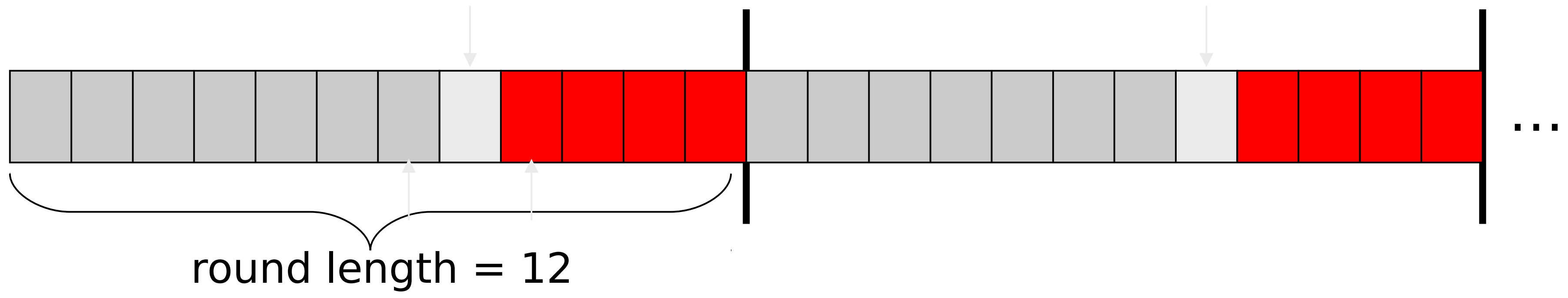
Weighted round robin

- Round robin:
  - Scan class queues serving one from each class that has a non-empty queue.
    - Provides Min-Max fairness.
    - Assumes fixed packet length.
    - Unfair if packets are of different length or weights are not equal.

- Weighted round robin:
  - Serve more than one packet per visit.
  - Number of packets are proportional to weights.

Weighted round robin

- Normalize the weights so that they become integer
- Suppose weights are: A = 1.4; B = 0.2 and C= 0.8
- Normalized weights: A = 7; B = 1 and C = 4



Weighted RR –variable length packet

If different connection have different packet size, then divide the weight of each connection with connection's mean packet size before normalizing

- weights {0.5, 0.75, 1.0},
- mean packet sizes {50, 500, 1500}

- normalize weights: {0.5/50, 0.75/500, 1.0/1500}
    - = { 0.01, 0.0015, 0.000666}

- normalize again: {60, 9, 4}