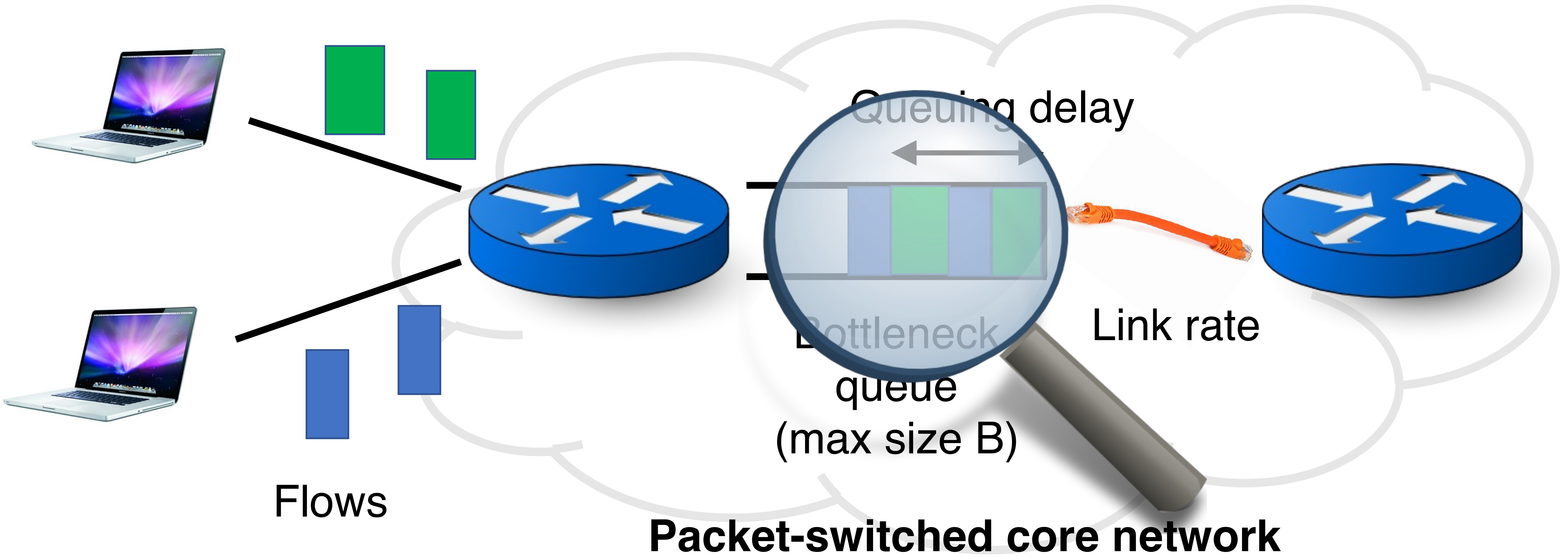
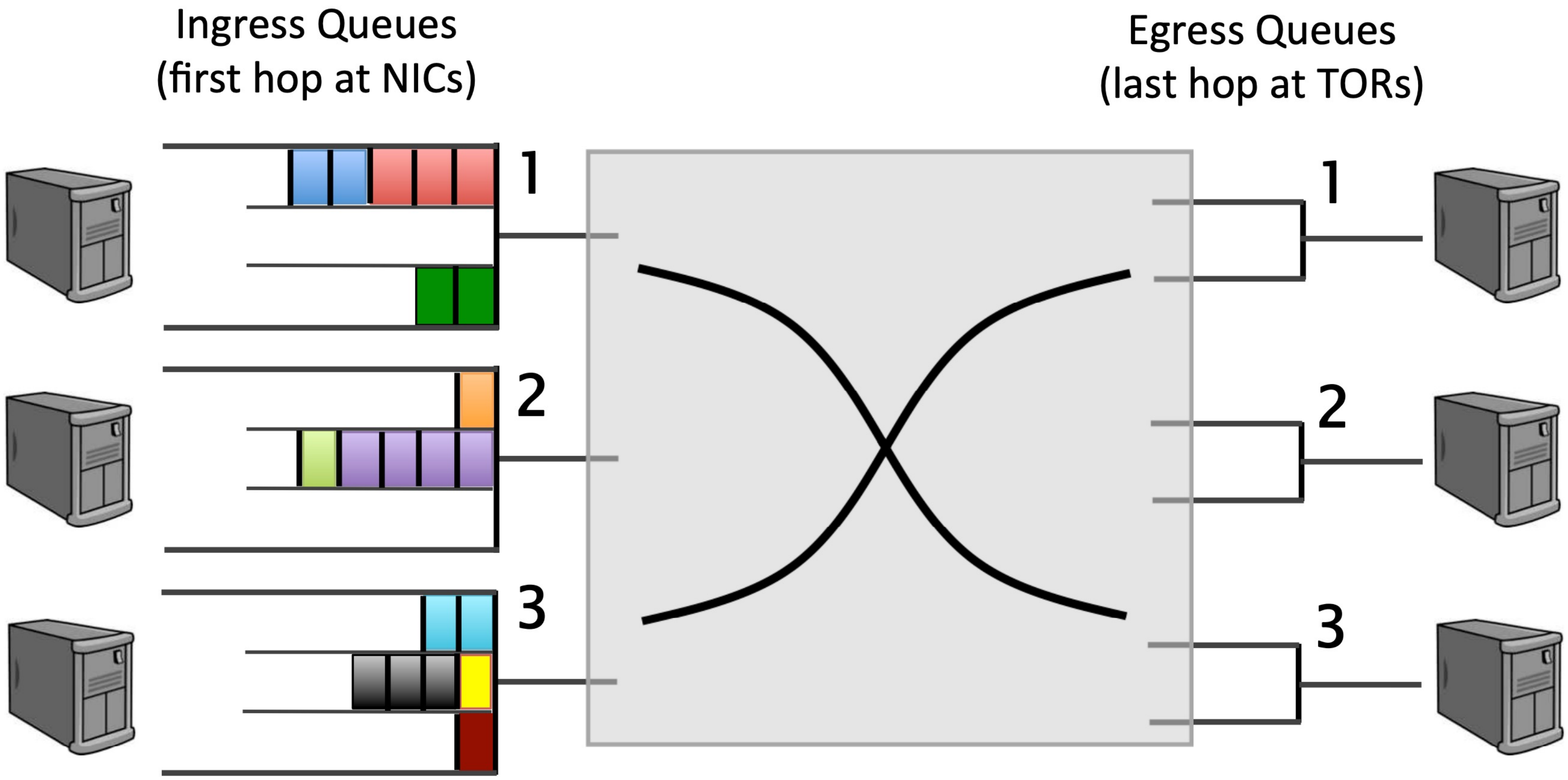
Transport

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Where scheduling operates



Transport and Scheduling

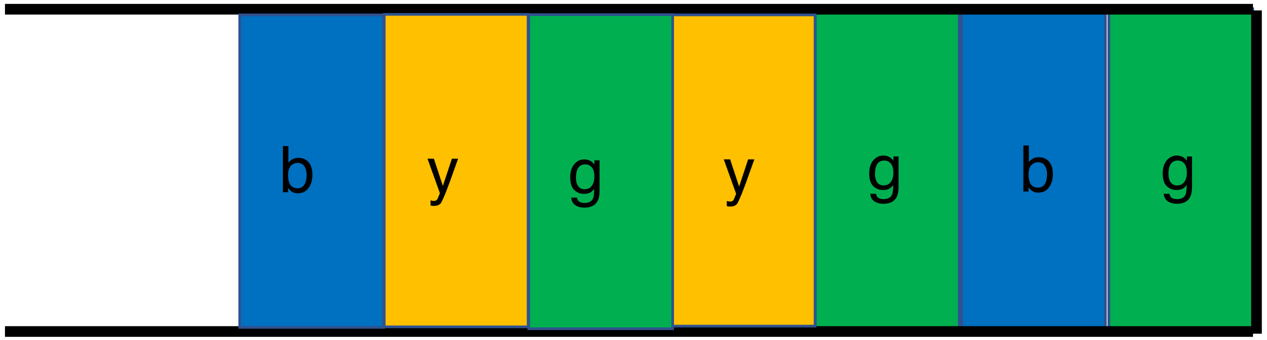


Packet Scheduling Algorithms

Which packet to send next? (order)

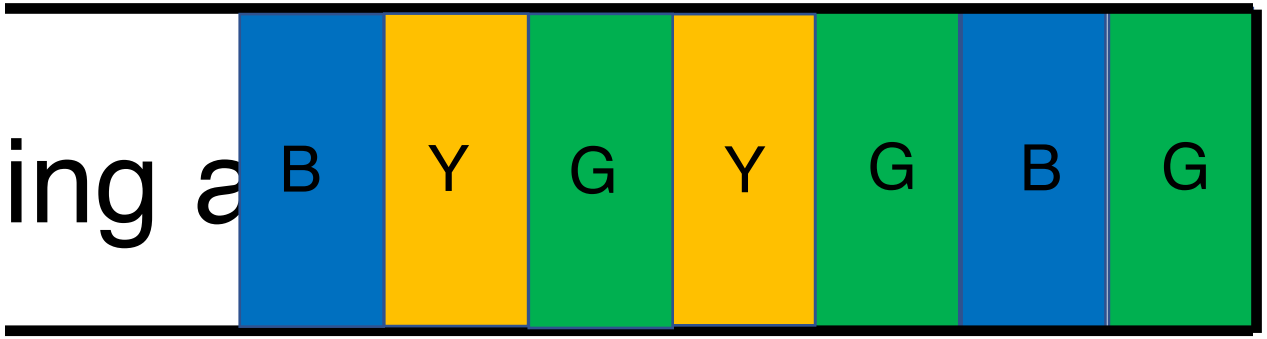
When to send the next packet? (timing)

A taxonomy



- Granularity of allocation
  - Per-packet vs. per-flow vs bit-by-bit
- Pre-emptive vs. non-pre-emptive
  - Do you interrupt the current packet/flow if another shows up?
- Size-aware vs. unaware
  - Do you consider flow or packet sizes in scheduling?
- Class-based (strict priority) vs. shared
  - Are some flows strictly higher priority than others?
- Work-conserving vs. non-work-conserving
  - Do you always use spare link capacity when there is demand?
- Metrics
  - Efficiency (completion, response); fairness; resource limiting

Examples of scheduling a



- First-In-First-Out (FIFO) over packets

- Round-robin over packets of different flows
  - G, B, Y, G, B, Y, etc. regardless of arrival order

- Shortest Remaining Processing Time (SRPT)
  - Flow-size-aware allocation which strictly prioritizes short flows
  - Variant: shortest flow first i.e., only consider (initial) remaining processing time
  - (note: it’s possible for a flow-size-unaware variant to predict remaining processing time using a known flow size *distribution)*

Examples of scheduling algorithms (2/N)

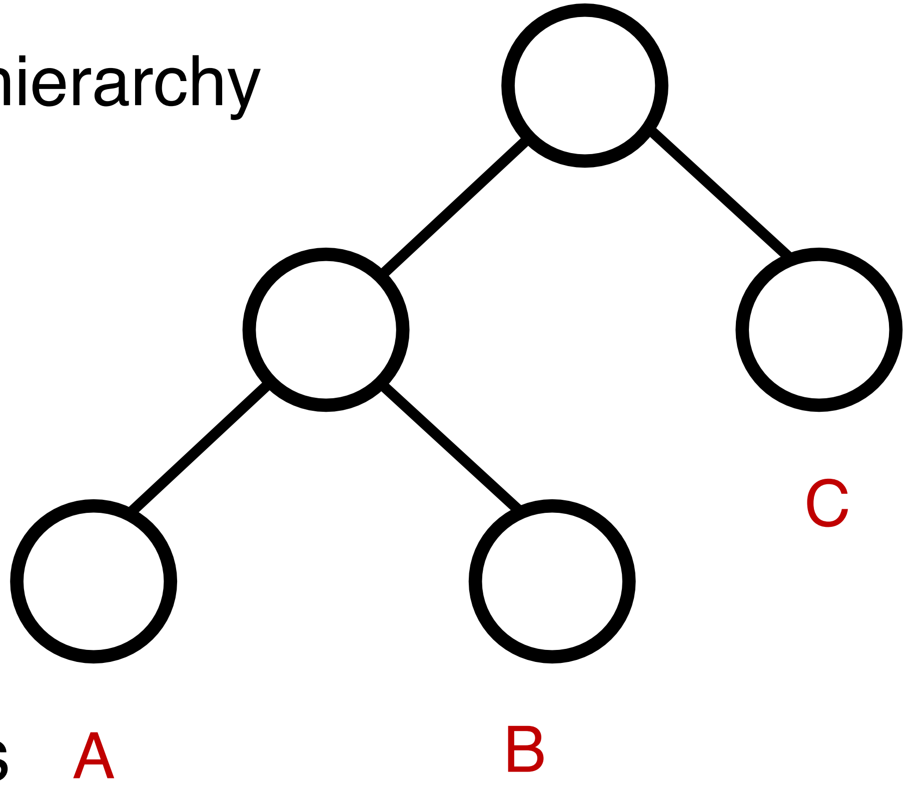
- Processor sharing
  - Assume each flow gets a fair share of the link every unit of time
  - Ideal: each flow starts receiving service immediately upon arrival
- Rate limiting
  - Non-work-conserving: flow can't send even if more demand than limit
- Class-based strict prioritization
  - Pre-determined flow classes with strict priorities over each other
  - Starve low priority flows if higher priority flows are always sending

Examples of scheduling algorithms (3/N)

- Hierarchical policies
  - Arrange scheduling policies in a tree-hierarchy

- Example:
  - Rate-limit A + B
  - Fair-share among A and B within limit
  - Fair-share among A+B and C

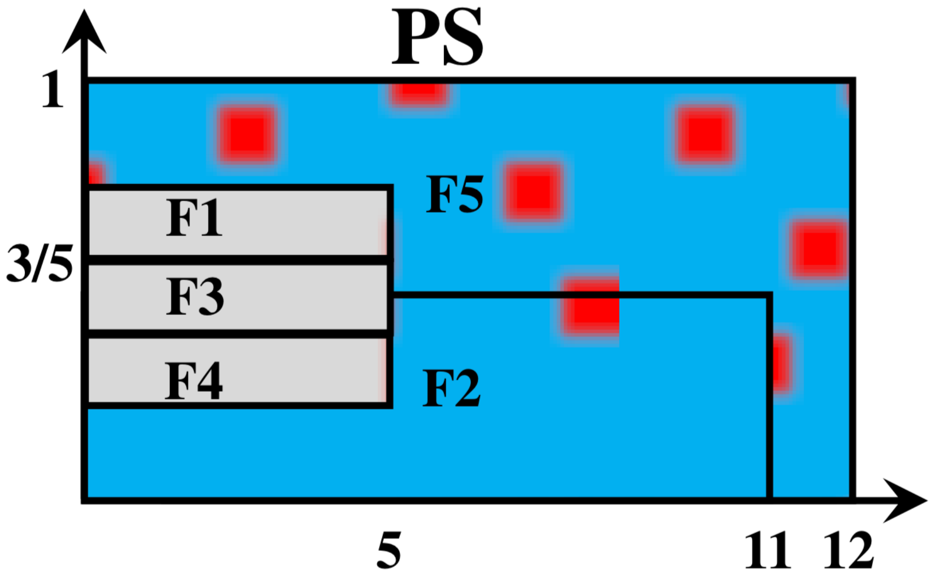
- Complex multi-tenant isolation policies
  - E.g., amazon AWS

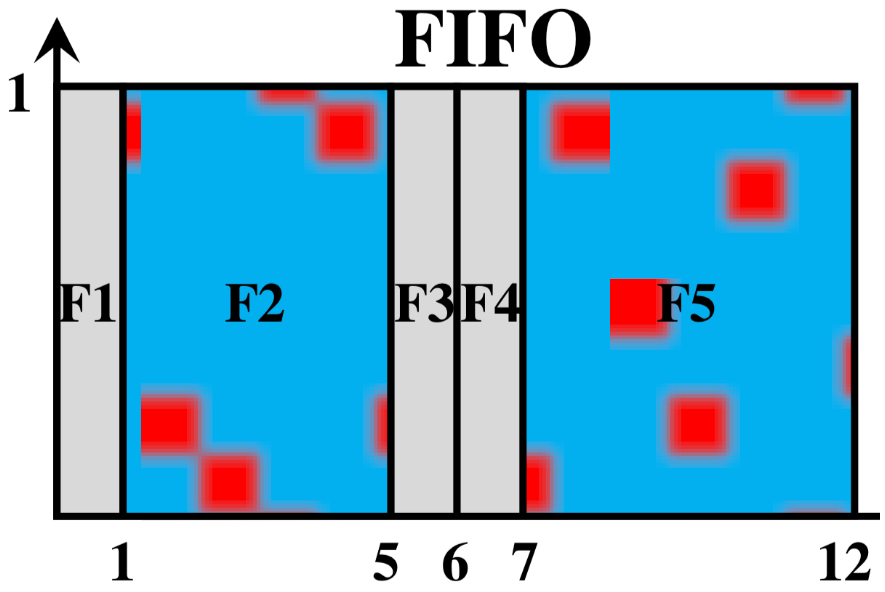


There’s no one (size-unaware) optimal scheduling

Flow ID	Size	Class
F1	1	1
F2	4	2
F3	1	1
F4	1	1
F5	5	2

The best policy depends on the distribution of flows in the workload. Suppose we consider the metric of average completion time of flows.



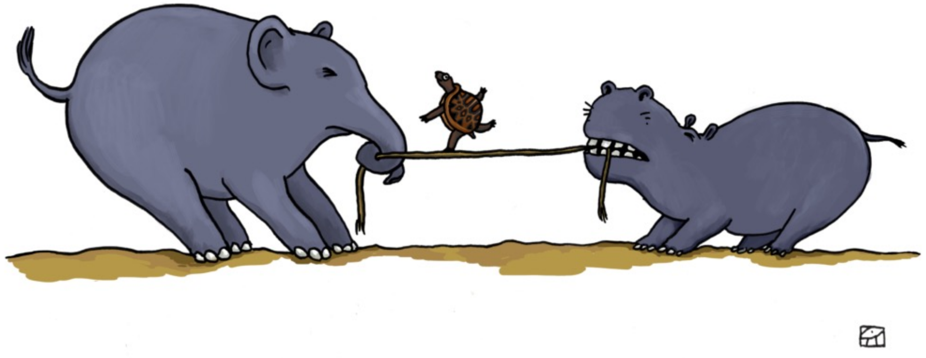


Workload adaptive flow scheduling, Faisal et al. CoNEXT 2018

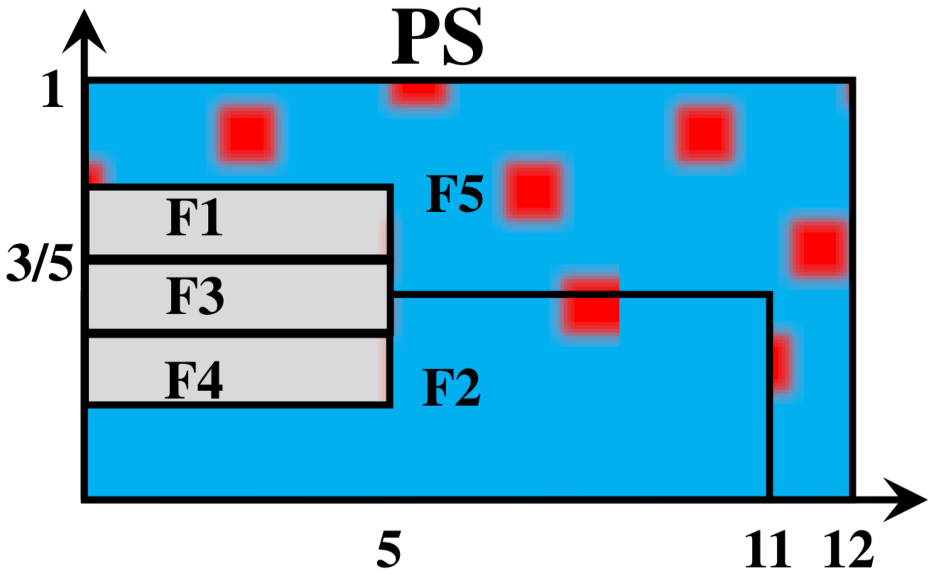
There's no one (size-unaware) optimal scheduling

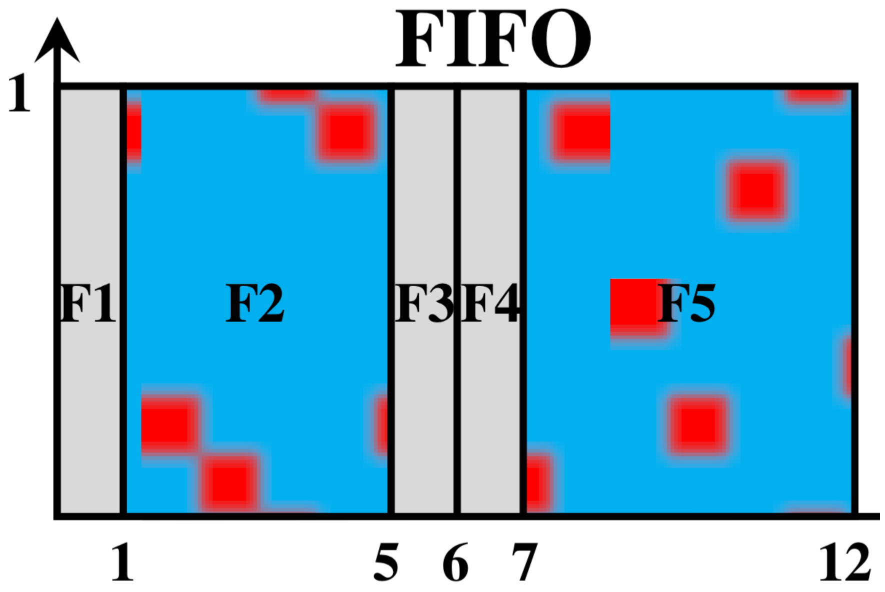
Flow ID	Size	Class
F1	1	1
F2	4	2
F3	1	1
F4	1	1
F5	5	2

Multiplexing
Avoids HOL blocking



Serialization
Reduces flow completion time





Workload adaptive flow scheduling, Faisal et al. CoNEXT 2018

When does a flow complete?

- Consider a mix of “long” and “short” flows arriving at a Q
  - Ex: A flow may have as few as 2 packets or as many as 10^6
- Suppose a scheduling algorithm provides each flow:
  - An average per-packet delay d (e.g., 50 ms)
  - An average link bandwidth share t (e.g., 10 Mbit/s)
- Which among d & t determines
  - when a short flow finishes?
  - when a long flow finishes?

Fair Queueing

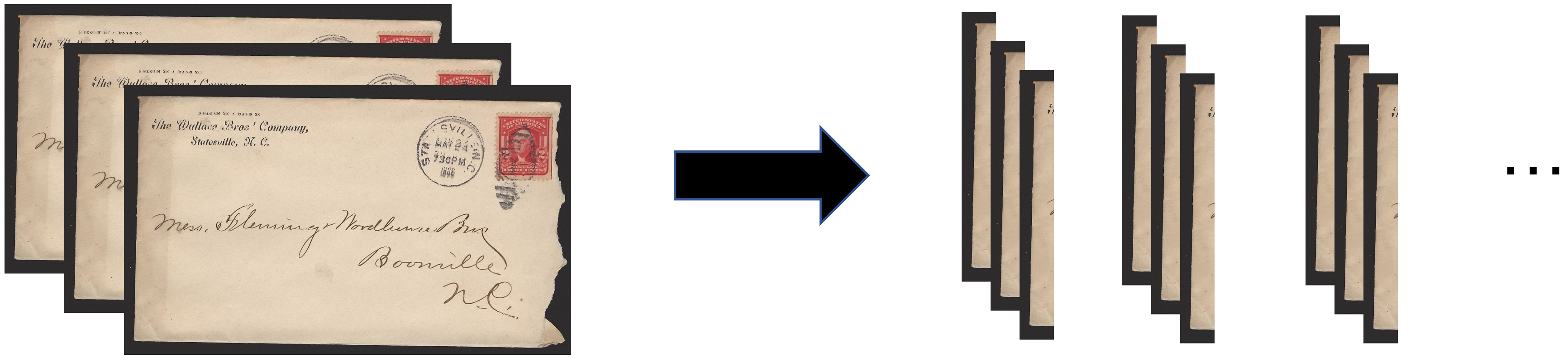
ACM SIGCOMM ’89

Alan Demers, Srinivasan Keshav, and Scott Shenker

An ideal to emulate: Processor sharing

- Fair-share bandwidth in the most fine-grained fashion possible
  - If there are N active flows, each flow gets 1/Nth of the link rate
    - N varies as flows arrive and leave
  - "Bit by bit round robin" (BR), also called processor-sharing

- Implementing BR directly on routers is unrealistic.
  - Reason: downstream router has no metadata to route the bit



Emulate processor sharing?

- How about emulating PS with round robin over packets?
- Unfair! A flow can use larger packets and gain larger bandwidth
- Instead, determine when a packet would finish with BR
  - Depends only on packet arrival time & # of active flows
  - Let’s call this the virtual finish time
- FQ: Transmit packets in the order of the virtual finish times
  - Buffer management: drop packet of flow with the largest backlog

Analysis and simulation of a fair queueing algorithm. Demers, Keshav, and Shenker

Round $R(t)$

Packet size $\mathbf{P}$

Ending round of a GPS packet $R(t) = R(t_0) + P$

Using round numbers as timestamps: $F_i^{\alpha} = S_i^{\alpha} + P_i^{\alpha}$

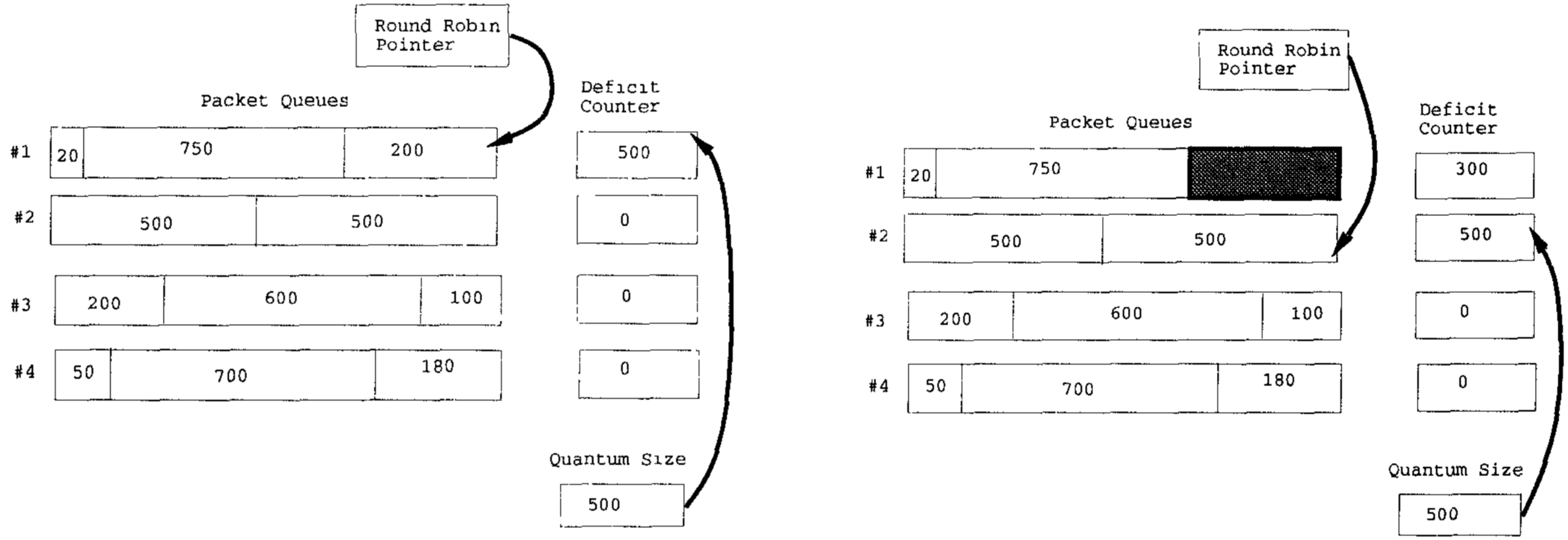
$$S_i^{\alpha} = MAX(F_{i-1}^{\alpha}, R(t_i^{\alpha}))$$

Schedule the flow and (its earliest packet) with the earliest finish time.

Finding the next flow: O(log N)

Hardware-friendly: Deficit Round Robin

- Set of queues iterated in order (O(1) next flow); fixed quantum
- Yet another approximation of BR: farther from it than WFQ

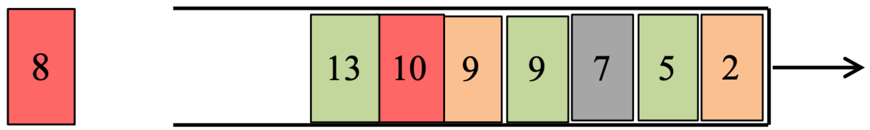


Efficient fair queueing using deficit round robin, Shreedhar and Varghese ‘95

Push in First Out

- Scheduling algorithms determine order and timing of packet departures from a queue

- Typically, relative order of buffered packets doesn't change upon new packet arrivals



- Implement scheduling through a priority-queue-based data structure (PIFO)
  - Push-In: pkts have arbitrary ranks; push anywhere into queue
  - First-Out: always dequeue from the head of the queue

Rate Limiting

Providing Isolation through Rate Limiting

Used to isolate flows from each other by giving each a fixed data rate through a link

Three commonly used terms:

- *(long term) average rate:* how many pkts can be sent per unit time (in the long run)
  - crucial question: what is the interval length? 100 packets per sec or 6000 packets per min have same average, but instantaneous behaviors can be very different
- *peak rate:* e.g., 6000 pkts per min (ppm) avg.; 1500 ppm peak rate
- *(max.) burst size:* max number of pkts sent consecutively (with no intervening idle)

Shaping and Policing

Policing	Enforces rate by *dropping* excess packets immediately – Can result in high loss rates + Does not require memory buffer + No RTT inflation
Shaping	Enforces rate by *queueing* excess packets + Only drops packets when buffer is full – Requires memory to buffer packets – Can inflate RTTs due to high queueing delay