بسمه تعالی

School of Computer Engineering

Advanced Computer Networks

Transport Layer and Congestion Control

Part 5

Seyed Hamed Rastegar

Fall 1401

Congestion Control: Evaluation Criteria

Fair Resource Allocation: Fairness Criteria

- There are various criteria to fairly allocate the resources.
- They consider an objective(Utility) function of the form $\sum_i U_i(x_i)$.
- Several wide-spread ones:
  - Proportional fairness: $U_i(x_i) = \log(x_i)$
  - Weighted Proportional fairness: $U_i(x_i) = w_i \log(x_i)$
  - α-fairness: $U_i(x_i) = \frac{x_i^{1-\alpha}}{1-\alpha}$
  - Max-min fairness: Overall Objective: $\max \min_i x_i$

Congestion Control: Evaluation Criteria

Fair Resource Allocation: Jain’s Fairness Index

- Assuming that fair implies equal and that all paths are of equal length, **Raj Jain** proposed a **metric** that can be used to quantify the fairness of a congestion control mechanism.

- Given a set of flow throughputs (*x_1*, *x_2*, . . . , *x_n*), (measured in consistent units such as bits/second), the following function assigns a fairness index to the flows:

$$f(x_1, x_2, \ldots, x_n) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i^2}.$$

Jain, R. K., Chiu, D. M. W., & Hawe, W. R. (1984). A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, *21*.

Congestion Control: Evaluation Criteria

Fair Resource Allocation: Jain's Fairness Index

- **Example 1**: consider a set of resource allocations as x = (1,0,1).
  - Jain's Index: 2/3

- **Example 2**: consider a set of resource allocations as x_i = 1, i = 1,…,k, and 0 oth.
  - Jain's Index: k/n
  - ➢ If y% of users treated fairly (equally resourced), and others starved (no resources), the index will be equal to y%.

- **Example 3**: consider a set of resource allocations as 50, 30, 50 while the optimal ones are 50, 10, 10.
  - Note: An approach is to first normalize the allocations according to optimal ones, and then compute the index: x = (50/50,30/10,50/10) = (1,3,5).
  - Jain's Index: 0.81

Congestion Control: Evaluation Criteria

Fair Resource Allocation: Other metrics

- Mean

$$\mu = \frac{1 + 3 \ + 5}{3} = 3$$

- Variance

$$\sigma^2 = 4$$

- Max-Min ratio

5/1 = 5

- Normalized distance from the optimal

Normalized Distance = 0.86

Congestion Control: Evaluation Criteria

Fair Resource Allocation: Properties of Jain's Fairness Index

- What is the range of the Jain's index?
  - It has minimum value of 1/*n* and maximum value of 1.
  - ➢ It is bounded.

➢ Scale independent

➢ Continuous

➢ Direct Relationship

Congestion Control: Queueing Disciplines

- Regardless of how simple or how sophisticated the rest of the resource allocation mechanism is, **each router must implement some queuing discipline** that governs how packets are buffered while waiting to be transmitted.

- The queuing algorithm can be thought of as **allocating** both **bandwidth** (which packets get transmitted) and **buffer space** (which packets get discarded).

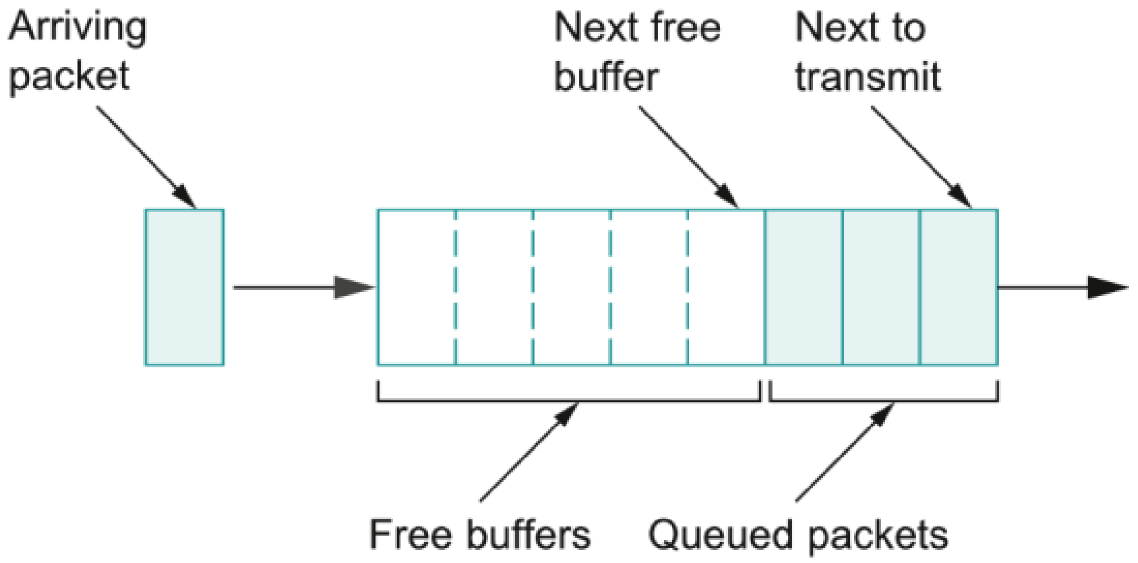
- It also directly affects the **latency** experienced by a packet by determining how long a packet waits to be transmitted.

- We investigate several queueing disciplines (aka *scheduling* and more completely *packet scheduling algorithms*).

Congestion Control: Queueing Disciplines

First In First Out (FIFO)

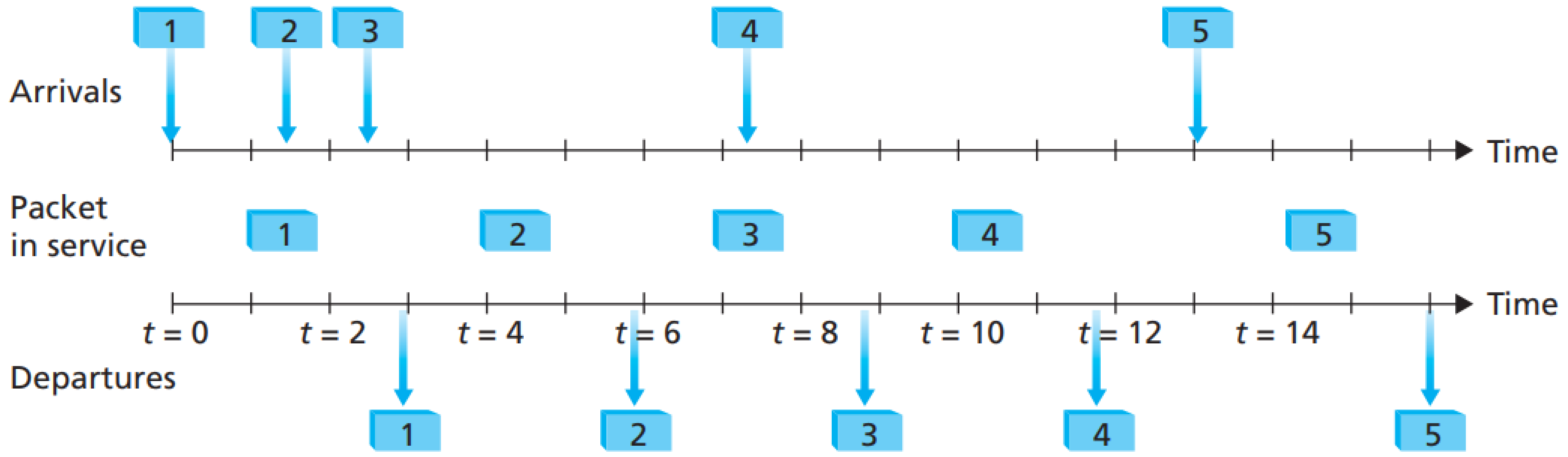
- The idea of FIFO queuing, also called first-come, first-served (FCFS) queuing, is simple:
  - the first packet that arrives at a router is the first packet to be transmitted.



Congestion Control: Queueing Disciplines

First In First Out (FIFO)

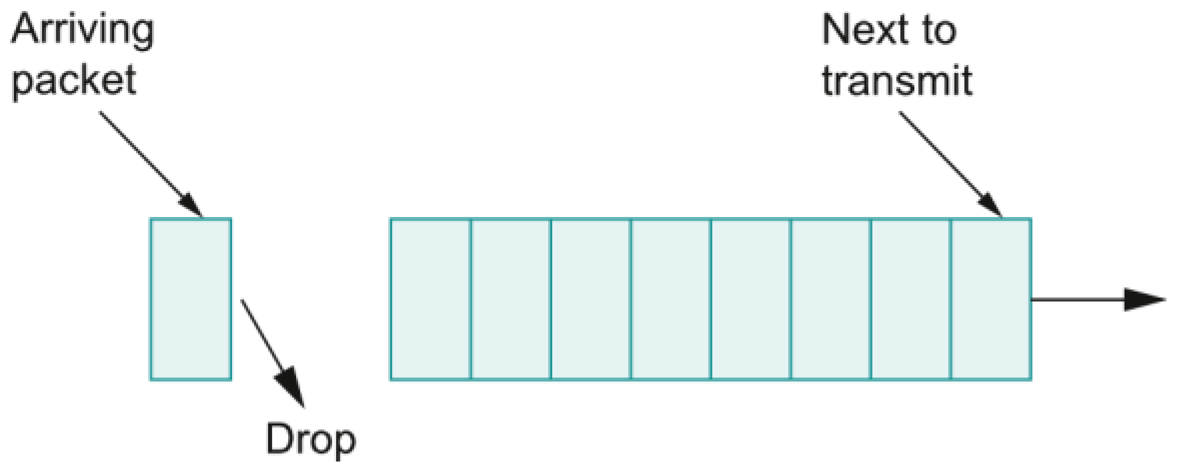
- FIFO queue in operation



Congestion Control: Queueing Disciplines

First In First Out (FIFO)

- Given that the amount of buffer space at each router is **finite**, if a packet arrives and the queue (buffer space) is **full**, then the **router discards that packet**
- This is done **without** regard to which flow the packet belongs to or how important the packet is. This is sometimes called **tail drop**, since packets that arrive at the tail end of the FIFO are dropped



Congestion Control: Queueing Disciplines

First In First Out (FIFO)

- Note that tail drop and FIFO are **two separable ideas**.
- **FIFO** is a **scheduling discipline**—it determines the order in which packets are transmitted.
- **Tail drop** is a **drop policy**—it determines which packets get dropped. Because FIFO and tail drop are the simplest instances of scheduling discipline and drop policy, respectively, they are sometimes viewed as a bundle—**the vanilla queuing implementation.**
- However, the bundle is often referred to simply as FIFO queuing, when it should more precisely be called FIFO with tail drop.

Congestion Control: Queueing Disciplines

First In First Out (FIFO)

- FIFO with tail drop, as the simplest of all queuing algorithms, is the **most widely used in Internet** routers currently.
- This simple approach to queuing pushes all responsibility for congestion control and resource allocation out to the edges of the network.
- Thus, the prevalent form of congestion control in the Internet currently assumes no help from the routers: TCP takes responsibility for detecting and responding to congestion.

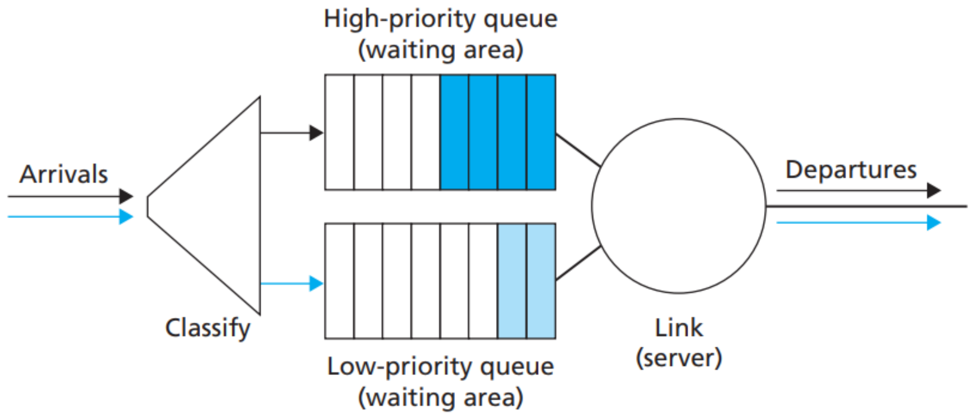
Congestion Control: Queueing Disciplines

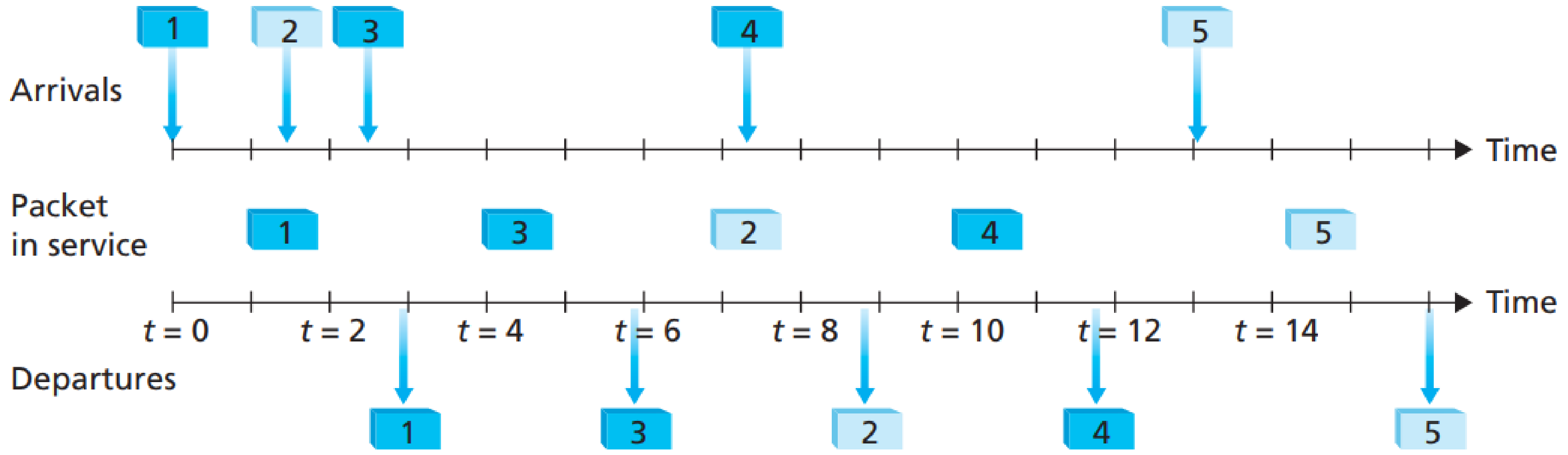
First In First Out (FIFO)

- A simple variation on basic FIFO queuing is **priority queuing**.
- The idea is to mark each packet with a priority; the mark could be carried, for example, in the IP header.
    - The routers then implement multiple FIFO queues, one for each priority class.
- The router always transmits packets out of the highest-priority queue if that queue is nonempty before moving on to the next priority queue.
    - Within each priority, packets are still managed in a FIFO manner.
- This idea is a small departure from the best-effort delivery model, but it does not go so far as to make guarantees to any particular priority class. It just allows high-priority packets to cut to the front of the line.

Congestion Control: Queueing Disciplines

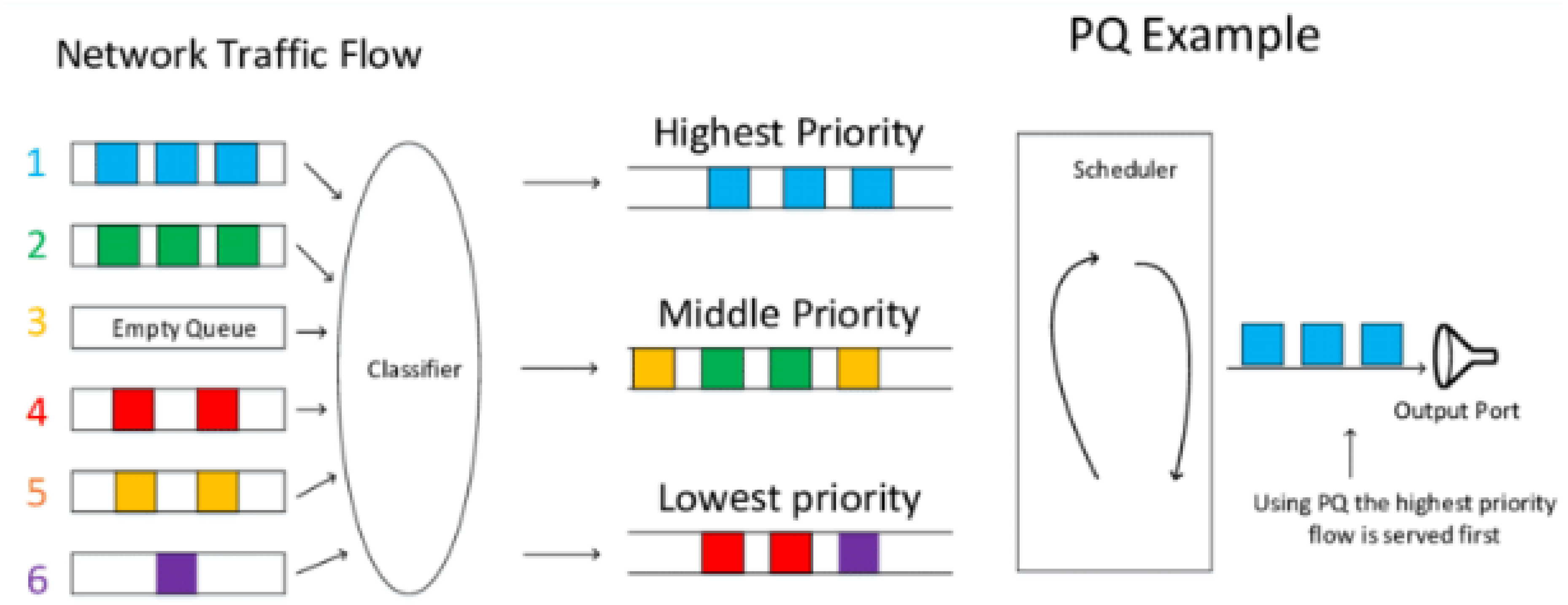
Priority queuing for two classes





Congestion Control: Queueing Disciplines

Priority queuing for multiple classes



Congestion Control: Queueing Disciplines

First In First Out (FIFO)

- The problem with priority queuing, of course, is that the high-priority queue can starve out all the other queues; that is, as long as there is at least one high-priority packet in the high-priority queue, lower-priority queues do not get served.
- For this to be viable, there need to be hard limits on how much high-priority traffic is inserted in the queue. It should be immediately clear that we cannot allow users to set their own packets to high priority in an uncontrolled way; we must either prevent them from doing this altogether or provide some form of "pushback" on users.
- One obvious way to do this is to use economics—the network could charge more to deliver high-priority packets than low-priority packets.
- However, there are significant challenges to implementing such a scheme in a decentralized environment such as the Internet.

Congestion Control: Queueing Disciplines

First In First Out (FIFO)

- One situation in which priority queuing is used in the Internet is to protect the most important packets—typically, the routing updates that are necessary to stabilize the routing tables after a topology change.
- Often there is a special queue for such packets, which can be identified by the Differentiated Services Code Point (formerly the **TOS** field) in the IP header.
- This is in fact a simple case of the idea of "**Differentiated Services (Diffserv)**."

Congestion Control: Queueing Disciplines

Fair Queueing

- The main problem with FIFO queuing is that it does not discriminate **between different traffic sources**, or, in the language introduced previously, it does not separate packets **according to the flow** to which they belong.

- This is a problem at two different levels:
  - **At one level**, it is not clear that any congestion control algorithm implemented entirely at the source will be able to adequately control congestion with so little help from the routers.
  - **At another level**, because the entire congestion control mechanism is implemented at the sources and FIFO queuing does not provide a means to police how well the sources adhere to this mechanism, it is possible for an ill-behaved source (flow) to capture an arbitrarily large fraction of the network capacity.

Congestion Control: Queueing Disciplines

Fair Queueing

- Considering the Internet again, it is certainly possible for a given application not to use TCP and, as a consequence, to bypass its end-to-end congestion control mechanism.
- Such an application is able to flood the Internet's routers with its own packets, thereby causing other applications' packets to be discarded.

➢ Fair queuing (FQ) is an algorithm that has been designed to address this problem.

Congestion Control: Queueing Disciplines

Fair Queueing

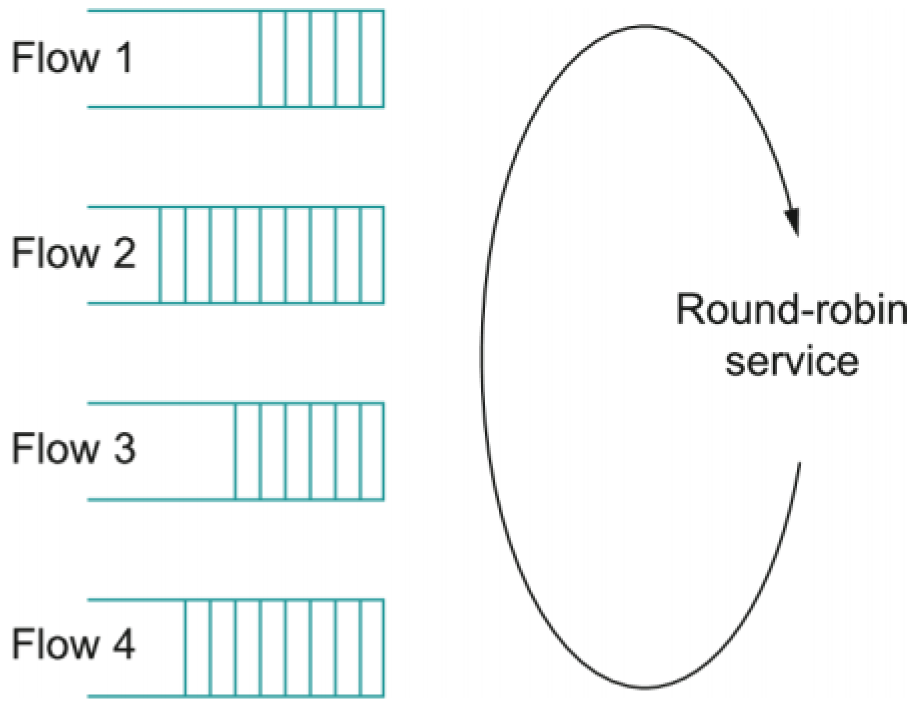
- The idea of FQ is to maintain a separate queue for each flow (or traffic class) currently being handled by the router.

- The router then services these queues **in a sort of round-robin**.
  - When a flow sends packets too quickly, then its queue fills up.
  - When a queue reaches a particular length, additional packets belonging to that flow’s queue are discarded.
  - In this way, a given source cannot arbitrarily increase its share of the network’s capacity at the expense of other flows.

Congestion Control: Queueing Disciplines

Fair Queueing

- Fair queuing (FQ)



Congestion Control: Queueing Disciplines

Fair Queueing

- Note that FQ does not involve the router telling the traffic sources anything about the state of the router or in any way limiting how quickly a given source sends packets.
- In other words, FQ is still designed to be used in conjunction with an end-to-end congestion control mechanism.
- It simply segregates traffic so that ill-behaved traffic sources do not interfere with those that are faithfully implementing the end-to-end algorithm.
- FQ also enforces fairness among a collection of flows managed by a well-behaved congestion control algorithm.

Congestion Control: Queueing Disciplines

Fair Queueing: Some Issues

- The main complication is that the packets being processed at a router are not necessarily the same length.
- To truly allocate the bandwidth of the outgoing link in a fair manner, it is necessary to take packet length into consideration.
- For example, if a router is managing two flows, one with 1000-byte packets and the other with 500-byte packets (perhaps because of fragmentation upstream from this router), then a simple round-robin servicing of packets from each flow's queue will give the first flow two-thirds of the link's bandwidth and the second flow only one-third of its bandwidth.

Congestion Control: Queueing Disciplines

Fair Queueing: Some Issues

- What we really want is **bit-by-bit round-robin**, where the router transmits a bit from flow 1, then a bit from flow 2, and so on.
- The bit-by-bit round-robin sometimes is also called **processor sharing (PS)**.
- Clearly, it is not feasible to interleave the bits from different packets.
- The FQ mechanism therefore simulates this behavior by first determining when a given packet would finish being transmitted if it were being sent using bit-by-bit round-robin and then using this finishing time to sequence the packets for transmission.

Congestion Control: Queueing Disciplines

Fair Queueing: Some Issues

- Calculating the finished time of packet *i*

$$F_i = \max(F_{i-1}, A_i) + P_i.$$

where A_i and P_i are the arrival time and length of packet *i.*

- Now, for every flow, we calculate F_i for each packet that arrives using the above formula.
- We then treat all the F_i as timestamps, and the next packet to transmit is always the packet that has the lowest timestamp— the packet that, based on the above reasoning, should finish transmission before all others.

Congestion Control: Queueing Disciplines

Fair Queueing: Some Issues

- Note that this means that a packet can arrive on a flow, and, because it is shorter than a packet from some other flow that is already in the queue waiting to be transmitted, it can be inserted into the queue in front of that longer packet.

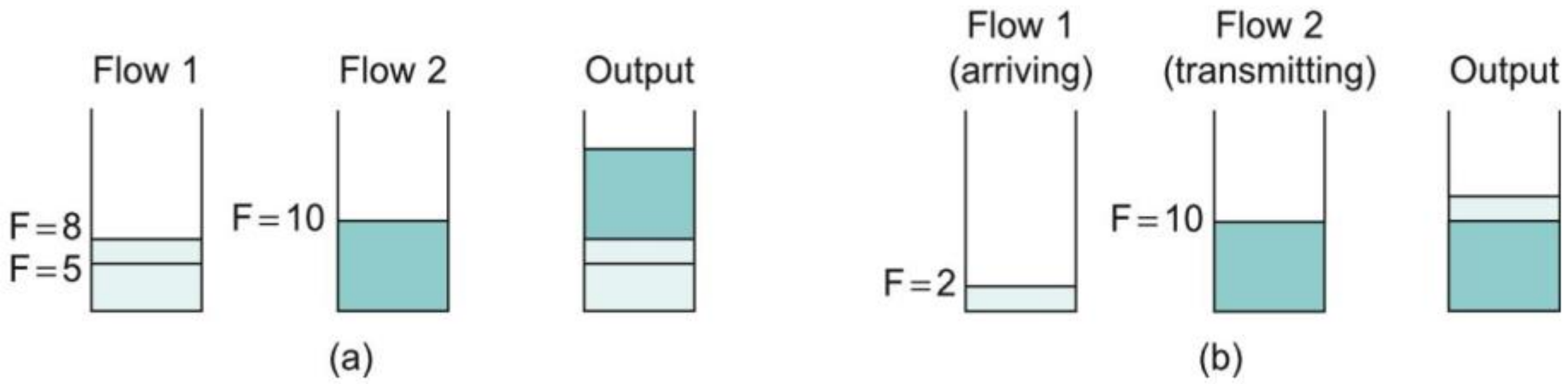
- However, this does not mean that a newly arriving packet can preempt a packet that is currently being transmitted.

- It is this lack of preemption that keeps the implementation of FQ just described from exactly simulating the bit-by-bit round-robin scheme that we are attempting to approximate.

Congestion Control: Queueing Disciplines

Fair Queueing: Some Issues

- Example



Congestion Control: Queueing Disciplines

Fair Queueing: Some Issues

- There are two things to note about fair queuing.

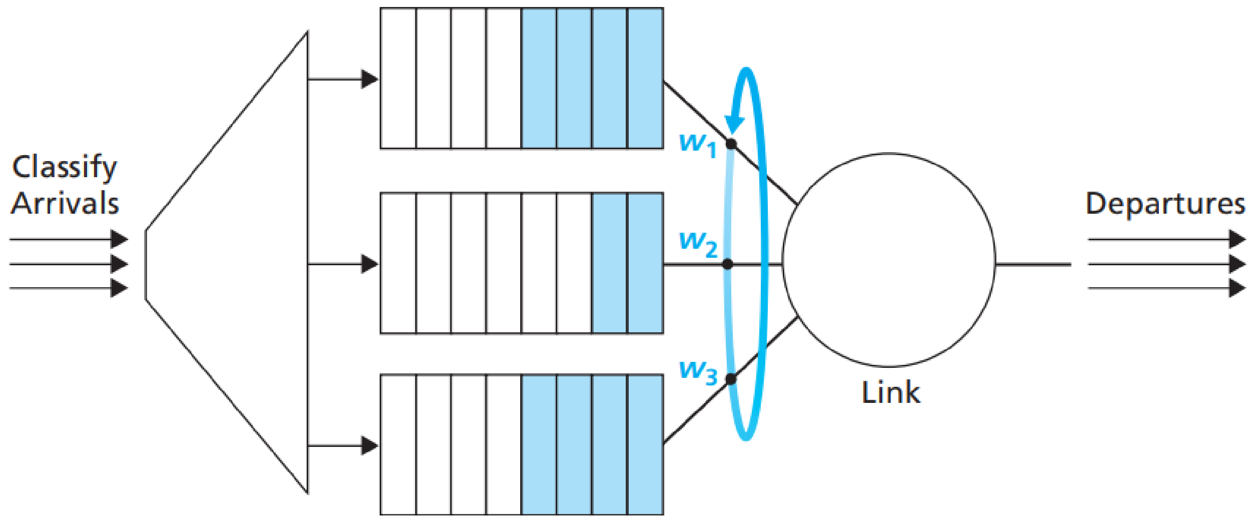
- **First**, the link is never left idle as long as there is at least one packet in the queue. Any queuing scheme with this characteristic is said to be **work-conserving**.
  - One effect of being work-conserving is that if I am sharing a link with a lot of flows that are not sending any data, then I can use the full link capacity for my flow. As soon as the other flows start sending, however, they will start to use their share, and the capacity available to my flow will drop.

- The **second** thing to note is that if the link is fully loaded and there are *n* flows sending data, I cannot use more than (1/*n*) of the link bandwidth.

Congestion Control: Queueing Disciplines

Fair Queueing: WFQ

- It is possible to implement a variation of FQ, called weighted fair queuing (WFQ), that allows a weight to be assigned to each flow (queue).



Congestion Control: Queueing Disciplines

Fair Queueing: WFQ

- WFQ differs from round robin in that each class (or flow) may receive a differential amount of service in any interval of time.
  - Specifically, each class, *i*, is assigned a weight, w_i .

- Under WFQ, during any interval of time during which there are class *i* packets to send, class *i* will then be guaranteed to receive a fraction of service equal to $w_i/(\sum w_j)$, where the sum in the denominator is taken over all classes that also have packets queued for transmission.

- Thus, for a link with transmission rate R, class *i* will always achieve a throughput of at least $R \cdot w_i/(\sum w_j)$.

Congestion Control: Queueing Disciplines

Fair Queueing: WFQ

- **Question**: How to implement WFQ in practice as we described for FQ?

- **Ideal case**: Assume packets each size of one bit
  - ➢ **Answer:** Send w_1 bits from queue 1, then w_2 bits of queue 2, then w_3 bits of queue 3, and so on.
    - o This, however **unrealistic** scheme, is known to **Generalized Processor Sharing (GPS).**

- **Practical Case:** Packets of arbitrary sizes
  - ➢ **Answer**: Instead of F_i as follows in FQ:

$$F_i = \max(F_{i-1}, A_i) + P_i.$$

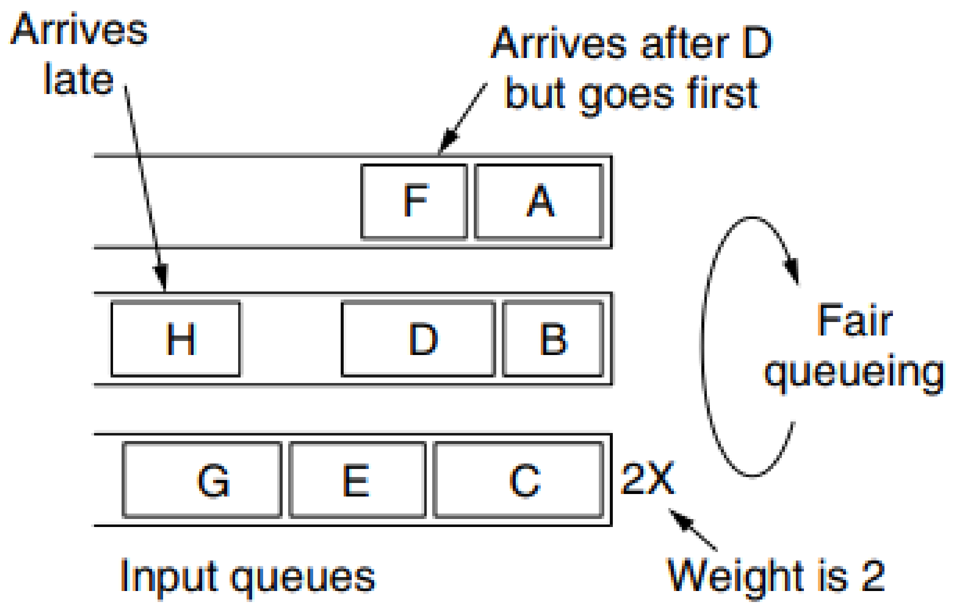
we consider its weighted version of

$$F_i^{WFQ} = \max\left(F_{i-1}^{WFQ}, A_i\right) + P_i/w_i.$$

Congestion Control: Queueing Disciplines

Fair Queueing: WFQ

- **Example**



(a)

Packet	Arrival time	Length	Finish time	Output order
A	0	8	8	1
B	5	6	11	3
C	5	10	10	2
D	8	9	20	7
E	8	8	14	4
F	10	6	16	5
G	11	10	19	6
H	20	8	28	8

(b)

Congestion Control: Queueing Disciplines

Fair Queueing: WFQ

- WFQ does need to make a sort of flows in the queues.
- For N active flows, the complexity of sorting would be O(log(*N*)) which is difficult to achieve for many flows in high-speed routers.
- To handle this problem a method called **deficit round robin (DRR)** has been proposed, which has O(1) complexity.
- In DRR, a quantity named **Quantum** is assigned to each class ($Quantum_i$ for class *i*) and in each round at most $Quantum_i$ number of bytes transmitted from class *i*.

Shreedhar, Madhavapeddi, and George Varghese. "Efficient fair queuing using deficit round-robin." *IEEE/ACM Transactions on networking* 4, no. 3 (1996): 375-385.