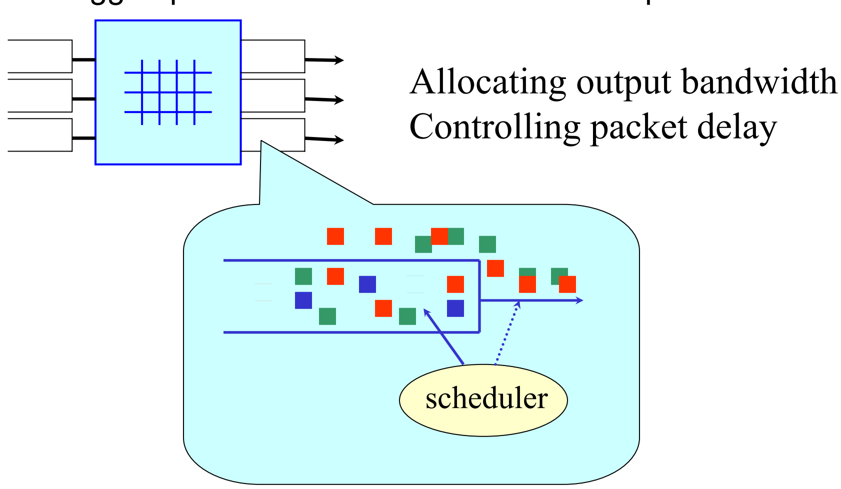
Packet Scheduling

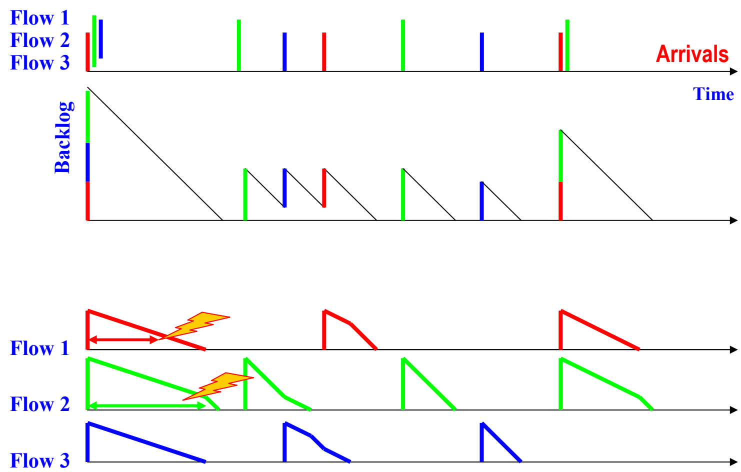
Packet Scheduling

- Packet scheduling algorithm determines the order in which backlogged packets are transmitted on an output link



FIFO Scheduling

Suppose we have QoS constraints on delay:
- Flow 1:
- Flow 2:
- Flow 3:

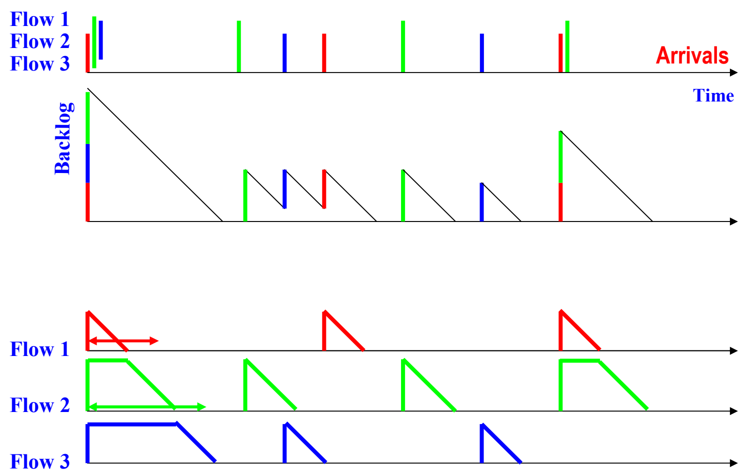


3

Priority Scheduling

QoS constraints on delay:
- Flow 1:
- Flow 2:
- Flow 3:

- Here: flow with short delay bound gets highest priority



4

Flows and Classes

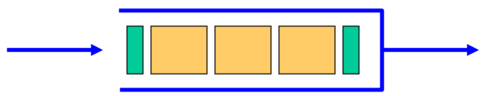
- A **flow** is a single end-to-end data stream
  - In a VC network, it is a connection
  - In a datagram network it is a transport level flow
- A **class** is a group of flows with common characteristics
  - Belong to the same application (video, data)
  - Belong to the same service (port number)
  - Have common header fields (e.g., DiffServ field)

Desirable Properties of a Packet Scheduler

- Protection among flows
  - Misbehaving flows do not affect well-behaving flows
- Guarantees
  - Differentiate between different types of traffic
  - Give guarantees on delay and rate
- Flexible
  - Accommodate wide range of service requests
- Simple
  - Easy to implement

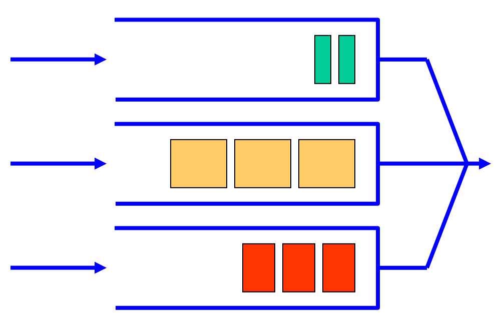
First-Come-First Served (FIFO)

- Packets are transmitted in the order of their arrival
- **Advantage:**
  - Very simple to implement
- **Disadvantage:**
  - Cannot give different service to different types of connections
  - Each flow (even with low data rate) can experience long delays



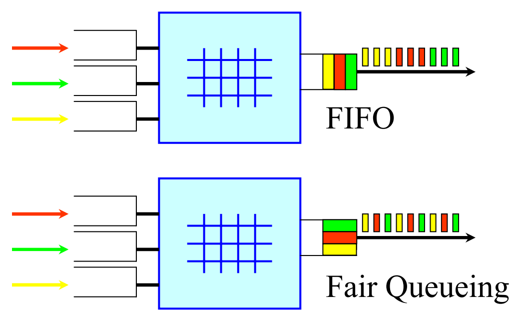
Static Priority

- Also called Head-of-Line (HOL) queueing:
  - Each traffic flow belong to a class
  - Each class has a priority
  - One FIFO queue for each class
  - Transmit from the highest priority queue with a backlog
- **Advantage:**
  - Simple
- **Disadvantage:**
  - Tends to “starve” the lower priority classes



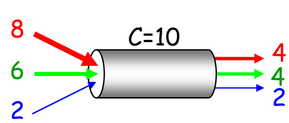
Fair Queueing

- Attempts to implement a scheduler that serves all flows with a backlog at the same rate
- Emulates a bitwise Round Robin scheduling algorithm
- Not easy to implement Fair Queuing in a packet network



Processor Sharing

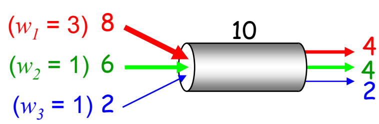
- **Processor Sharing: Each flow receives a "fair share" of bandwidth**
  - r_i – flow arrival rate
  - f –fair rate of a link
  - C – link capacity
- If link congested, compute **fair share** f such that $\sum_i \min(r_i, f) = C$



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

Generalized Processor Sharing (GPS)

- Associate a weight w_i with each flow i
- If link congested, $\sum_i \min(r_i, f \times w_i) = C$

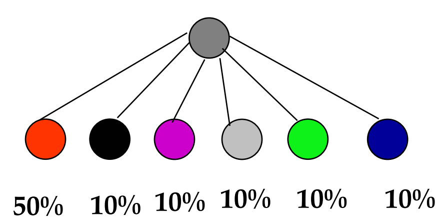


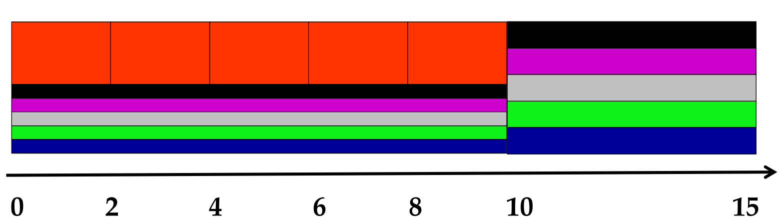
f = 2:
min(8, 2*3) = 6
min(6, 2*1) = 2
min(2, 2*1) = 2

- Flow i has a rate guarantee g_i of $g_i \geq \frac{w_i}{\sum_k w_k} C$

Generalized Processor Sharing (GPS)

- **Red session has packets backlogged between time 0 and 10**
- **Other sessions have packets continuously backlogged**

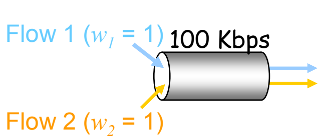




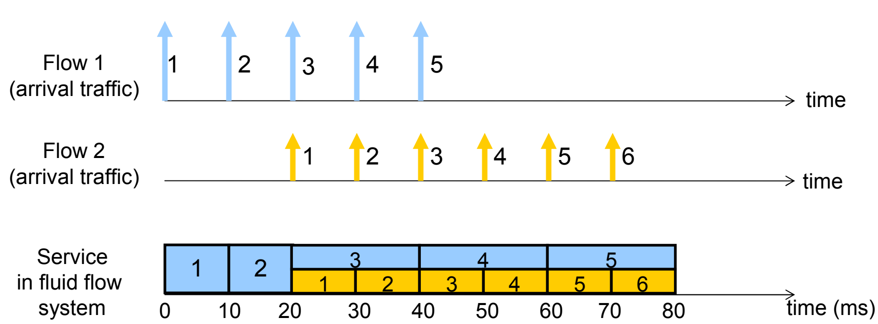
How to implement processor sharing?

- PS and GPS take a fluid-flow view of traffic, i.e., all flows can be served at the same time
- In practice:
  - Traffic is transmitted in packets with a variable size
  - Transmission of a packet cannot be interrupted
- PS and GPS are emulated by packet-based scheduling algorithms
  - Many such algorithms exists
  - Most routers implement one version

Fluid Flow System: Example



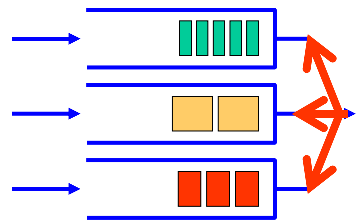
	Packet Size (bits)	Packet inter-arrival time (ms)	Rate (Kbps)
Flow 1	1000	10	100
Flow 2	500	10	50



Slide from Ion Stoica

Weighted Round Robin (WRR)

- Simple emulation of GPS
  - Operates in “rounds”
  - L_i is the average packet size of flow i
- Calculate the number of packets to be served in each round:
  - *For each flow i:* $x_i = w_i / L_i$
  - $x = min_i \{ x_i \}$
  - *For each flow i:* $packets\_per\_round_i = x_i / x$

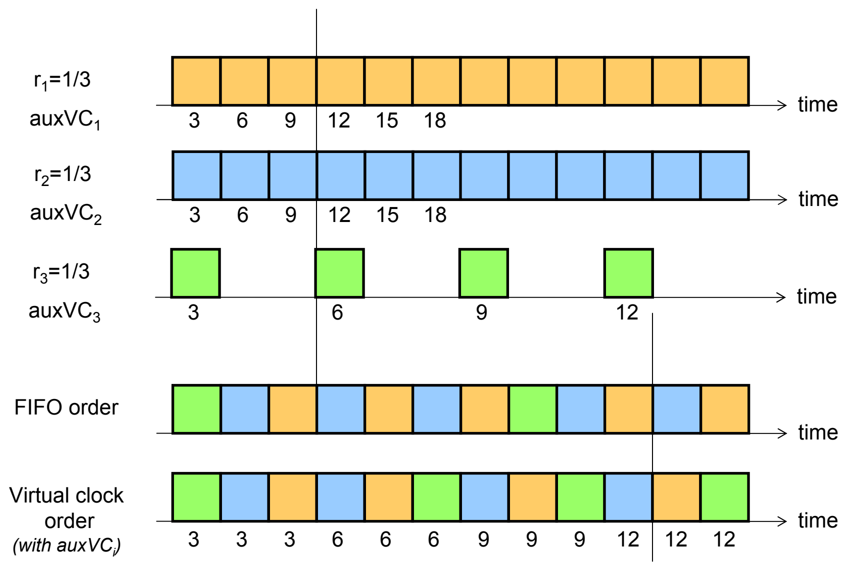


- WRR is a good approximation of GPS if
  - All flows are active
  - Over long periods of time

Virtual Clock (VC)

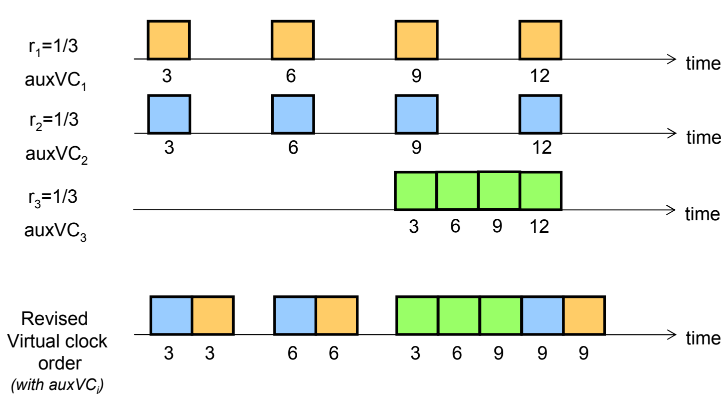
- Emulates a time-division multiplexing (TDM) system
- Two state variables for each flow j:
  - $auxVC_j$ virtual transmission time of the flow
  - r_j reserved rate
- Upon arrival of a packet from flow j *with size* L_j^k *at time* a_j^k:
  - $auxVC_j = max\ (auxVC_j, a_j^k) + L_j^k / r_j$
  - Stamp $auxVC_j$ in packet header
  - Packet are transmitted in increasing order of virtual transmission times

Example: Virtual Clock



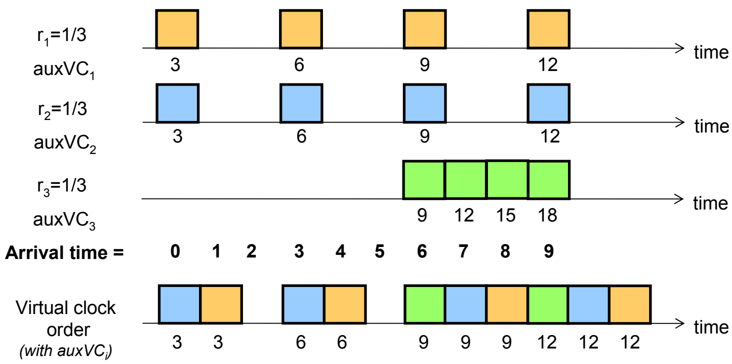
Example: Virtual Clock

- Why the $max\ (auxVC_{j\,,}\ a_j^k)$?
- Suppose: $auxVC_j = auxVC_j + L_j^k / r_j$
- Then: Flows can accumulate credits when they are not active



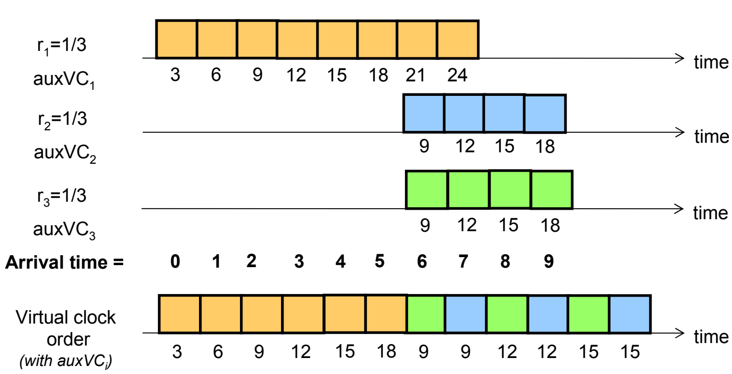
Virtual Clock

- "$auxVC_j = max\ (auxVC_{j}, a_j^k) + L_j^k / r_j$" prevents credit accumulation of idle flows



Problem with Virtual Clock

- Flow that gets more than reserved rate may be penalized in the future

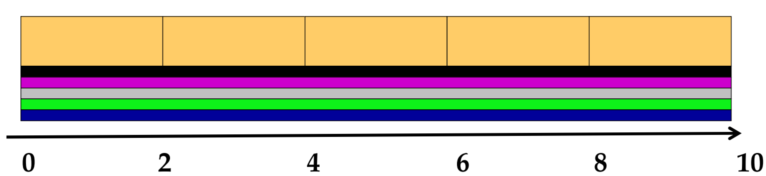


Weighted Fair Queueing (WFQ)

- Also called Packetized Generalized Processor Sharing (PGPS)
- Scheduling decision:
  Transmit the packet that will finish first in the ideal fluid GPS system
  - Uses virtual finishing times
  - Timestamps are assigned when a packet arrives
  - Orders packets in increasing order of finishing times

Weighted Fair Queueing

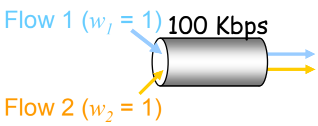
- GPS assumes that traffic behaves like a fluid flow system, that can transmit one bit at a time



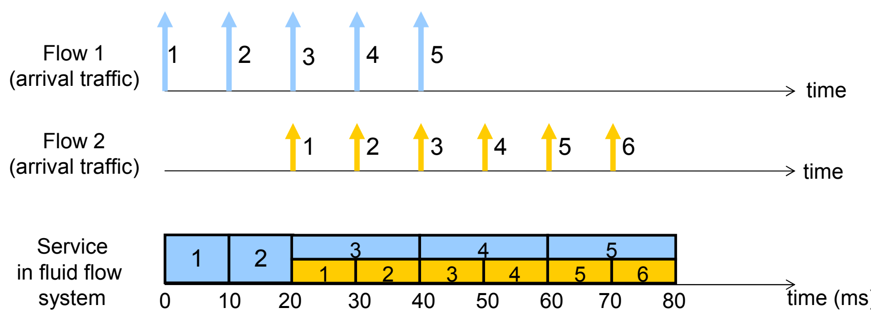
- In practice, GPS can only be approximated
- **Weighted Fair Queueing:** Select the first packet that finishes in GPS

- Can be proven: WFQ can approximate GPS very closely

WFQ scheduling



	Packet Size (bits)	Packet inter-arrival time (ms)	Rate (Kbps)
Flow 1	1000	10	100
Flow 2	500	10	50



Slide from Ion Stoica

GPS characterization

- There are N flows with weights $w_1, w_2, \ldots, w_N$
- The service given to two backlogged flows is proportional to their weights
- At any time *t* , the service rate of a backlogged flow *i* is:

$$\frac{w_i}{\sum_{j \in B(t)} w_k} C$$

where *B(t)* is the set of backlogged flows at time *t*

C is the capacity of the link

- The total rate guarantee to a flow is:

$$g_j = \frac{w_j}{\sum_k w_k} C$$

System Virtual Time

- Virtual time (V_{GPS}) – service that backlogged flow with weight = 1 would receive in GPS
- W_i is th service received by flow i

$$W_i(t, t+dt) = w_i \times \frac{W(t, t+dt)}{\sum_{j \in B(t)} w_j} \qquad \forall i \in B(t)$$

$$\frac{\partial W_i}{\partial t} = \frac{w_i}{\sum_{j \in B(t)} w_j} \times \frac{\partial W}{\partial t} \qquad \forall i \in B(t)$$

$$\frac{\partial V_{GPS}}{\partial t} = \frac{1}{\sum_{j \in B(t)} w_j} \times \frac{\partial W}{\partial t}$$

$$W_i(t_1, t_2) = w_i \times \int_{t=t_1}^{t_2} \left(\frac{1}{\sum_{j \in B(t)} w_j} \times \frac{\partial W}{\partial t} \right) dt \qquad \forall i \in B(t)$$

Service Allocation in GPS

- The service received by flow i during an interval $[t_1, t_2)$, while it is backlogged is

$$W_i(t_1, t_2) = w_i \times \int_{t=t_1}^{t_2} \frac{\partial V_{GPS}}{\partial t} dt \qquad \forall i \in B(t)$$

$$W_i(t_1, t_2) = w_i \times (V_{GPS}(t_2) - V_{GPS}(t_1)) \qquad \forall i \in B(t)$$

WFQ

- WFQ uses a **Virtual Time** that tracks the progress of GPS system
- When fewer flows are active, virtual time moves faster

$$V(0) = 0$$
$$V(t_l) = V(t_{l-1}) + \frac{\tau}{\sum_{j \in B_l} w_j} \qquad \tau \leq t_l - t_{l-1}$$

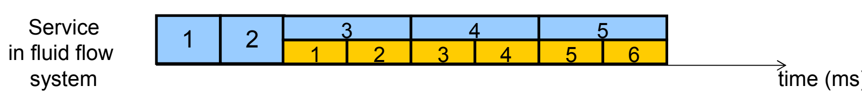
Here: B_l is the set of backlogged flows in time interval $[t_{l-1}, t_l]$

- Virtual finish time of k-th packet from flow j

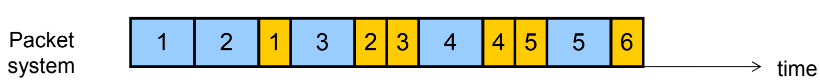
$$F_j^k = \max\{F_j^{k-1}, V(a_j^k)\} + \frac{L_j^k}{w_j}$$

- Packets are sorted and transmitted in the order of virtual finishing times

Packet System: Example 1



- Select the first packet that finishes in the fluid flow system

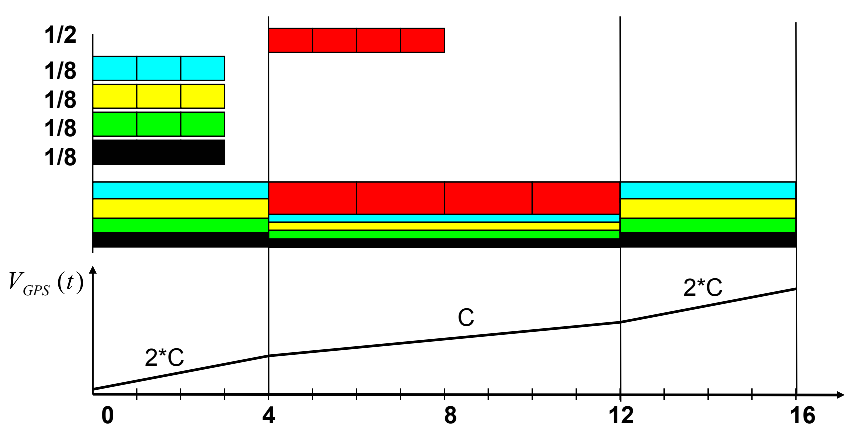


- **WFQ is not easy to implement.**
- Need to keep track of the finishing times of all packets in the GPS system
- Solution: Maintain a sorted queue that sorts packets according to finishing times in the GPS system

Virtual Time Implementation of Weighted Fair Queueing

- Need to keep per flow instead of per packet virtual start, finish time only
- System virtual time is used to reset a flow's virtual start time when a flow becomes backlogged again after being idle

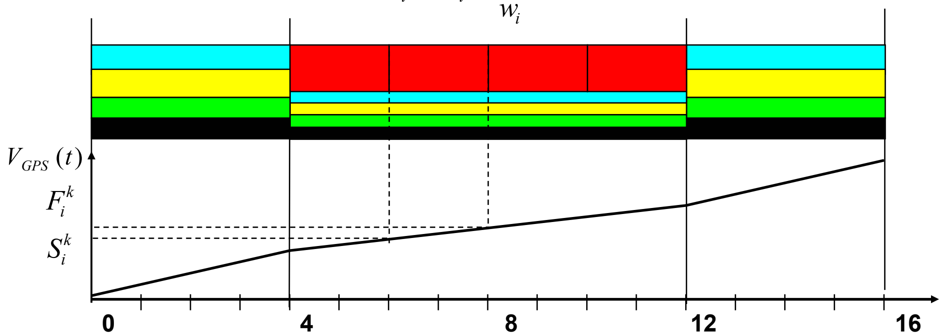
System Virtual Time in GPS



Virtual Start and Finish Times

- Utilize the time the packets would start S_i^k and finish F_i^k in a fluid system

$$F_i^k = S_i^k + \frac{L_i^k}{w_i}$$



Slide from Ion Stoica