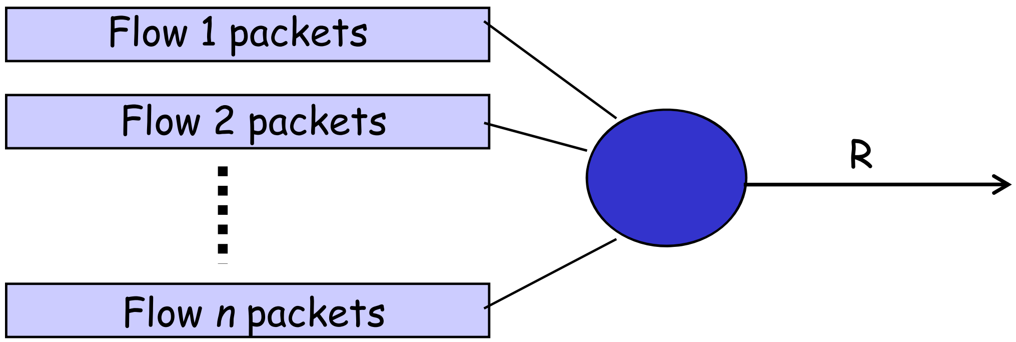
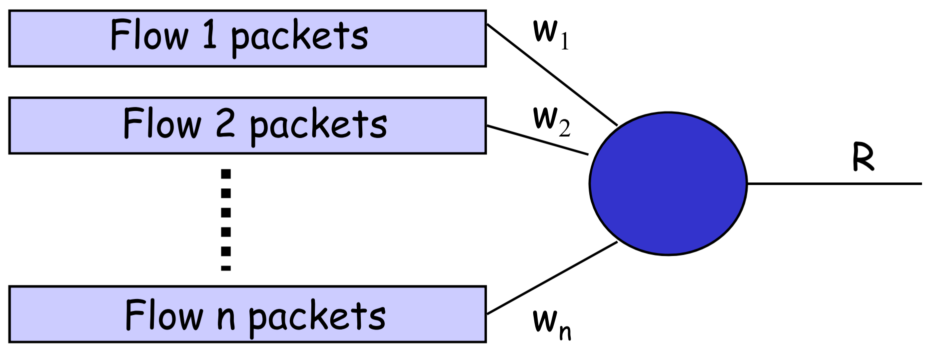
# Packet Scheduling: Weighted Fair Queueing (WFQ) and Virtual Clock (VC)

# Fair Queueing server




1. Round robin service
   - Packet-by-packet round robin
2. Processor sharing
   - Each flow receives a service rate of *R/n*, where *R* is channel rate in bps and *n* is the number of flows with non-empty queue

# Weighted Fair Queueing (WFQ) server




- Each flow $i$ is given a weight $w_i$
- Service rate received by flow $i$ is

  $$r_i = R * w_i / (w_1 + w_2 + ... + w_n)$$

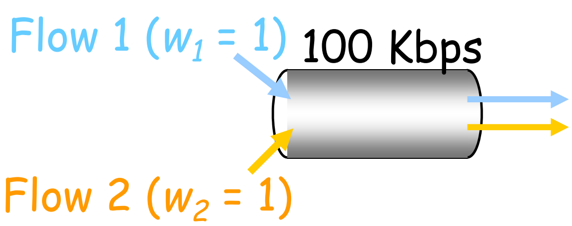
  where $R$ is channel rate in bps

# Fluid Flow System (Processor Sharing)

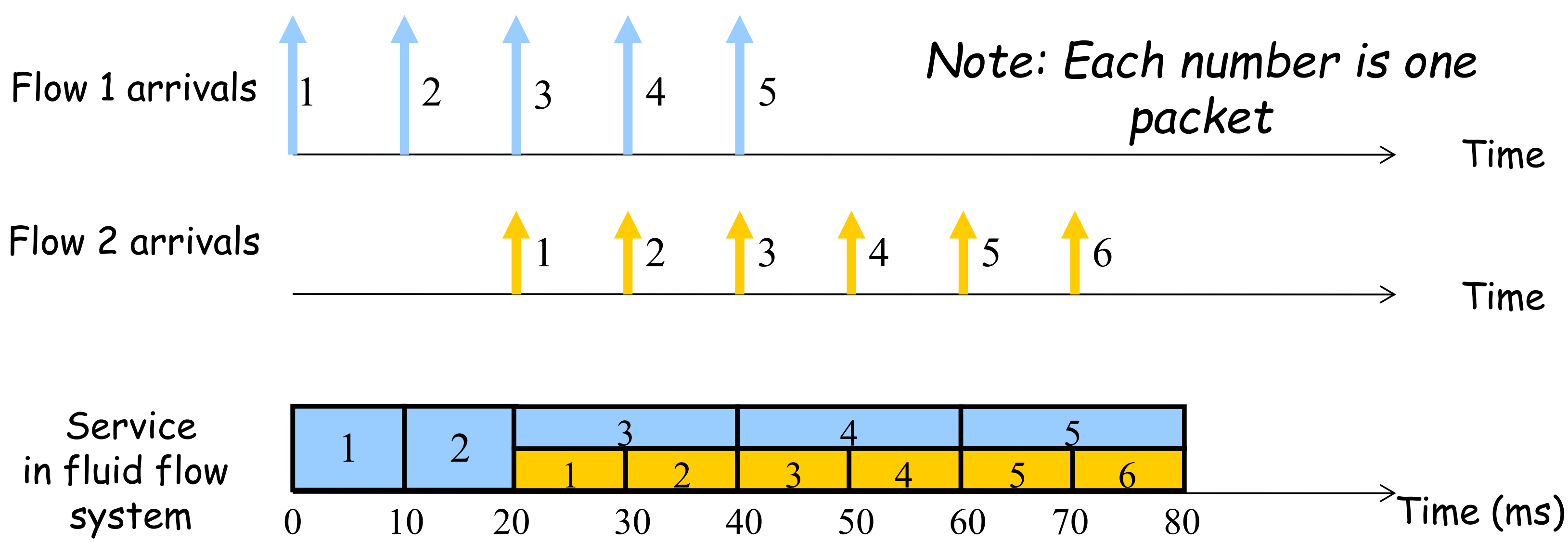
- Assume work-conserving server with no scheduling overhead

- Processor sharing can be implemented (conceptually) using *bit-by-bit* weighted round robin
  - During each round, each flow with data sends a number of bits proportional to the flow's weight

# Fluid Flow Example

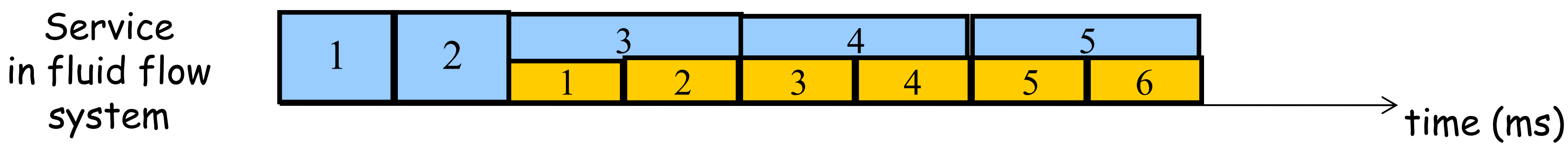



| | Packet Size (bits) | Packet inter-arrival time (ms) | Arrival Rate (Kbps) |
|---|---|---|---|
| **Flow 1** | **1000** | **10** | **100** |
| **Flow 2** | **500** | **10** | **50** |

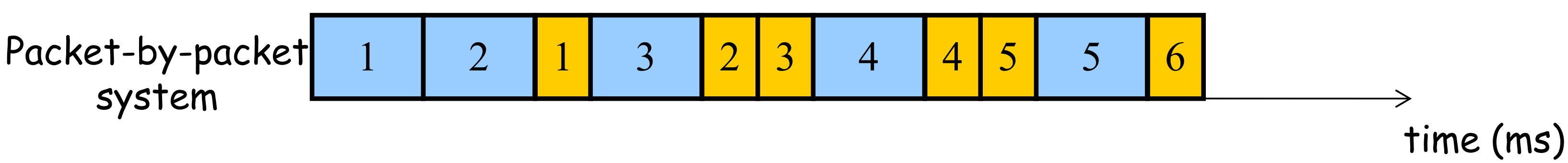


# Packet-by-packet system

- bit-by-bit round robin is not practical




- Idea: Use finishing time of packet in fluid system as priority for choosing the next packet for service




*Note: Each number is one packet*

# Packet finishing time

- Define
  - $L^f(j)$ finishing time of packet $j$ in flow $f$
  - $A^f(j)$ arrival time of packet $j$ in flow $f$
  - $s^f(j)$ length (in bits) of packet $j$ in flow $f$
- Suppose $r^f$ is a constant service rate (bits/sec) allocated to flow $f$
- Then the finishing time of packet ***j*+1** in flow $f$ would be

$$L^f(j+1) = \max\{L^f(j), A^f(j+1)\} + \frac{s^f(j+1)}{r^f}$$

## However

In WFQ, the service rate received by each flow changes whenever a new flow arrives

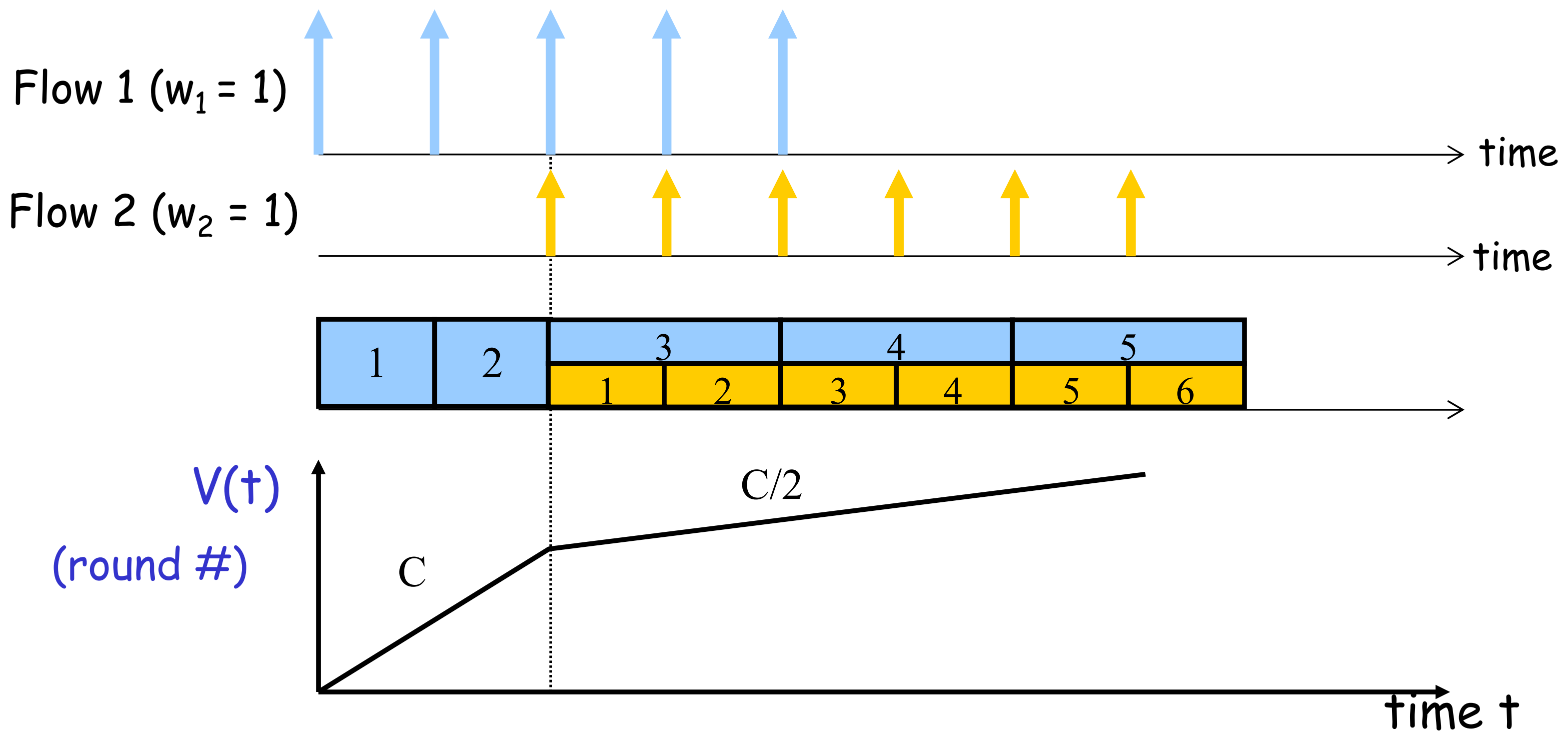
- When this happens, the finishing times of packets change and will need to be recomputed

# Solution: Virtual Time, V(t)

- *V(t) is the **round number** in the bit-by-bit round robin system*
- Observation: When a packet of a new flow arrives, an existing packet's finishing time in round number does not change
  - thus, finish order of existing packets does not change
- Instead of a packet's finish time, compute the round # when a packet will finish (virtual time finishing time)

# Virtual Time, V(t) (cont.)

- V(t) increases inversely proportionally to the sum of the weights of the backlogged flows



# WFQ/PGPS scheduling

- Define
  - $P^f(j)$ virtual time finishing time (in round number) of packet $j$ in flow $f$
  - $A^f(j)$ arrival time of packet $j$ in flow $f$
  - $s^f(j)$ length (in bits) of packet $j$ in flow $f$
  - $w^f$ weight (in bits) of flow $f$
- The priority of packet *j+1* in flow *f* is its *virtual time finishing time*

$$P^f(j+1) = \max\{P^f(j),\ V(A^f(j+1))\} + \frac{s^f(j+1)}{w^f}$$

- Select the packet with smallest priority for service

# PGPS delay relative to GPS

Let $L_{PGPS}(i)$ denote the departure time of packet $i$ for packet-by-packet WFQ service

Let $L_{GPS}(i)$ denote the departure time of packet $i$ for bit-by-bit WFQ service

Parekh and Gallager (1993) proved

$$L_{PGPS}(i) \leq L_{GPS}(i) + v_{\max}$$

where $v_{\max}$ is maximum packet service time

No notion of a reserved rate, nor admission control that bounds the number of flows

# Virtual Clock scheduling [L. Zhang 1990]

- A flow *f* is allocated a **reserved service rate**, $r^f$, similar to TDM
  - but unlike TDM, if a flow idles, its reserved rate can be used by other flows
- Like TDM, however, there are "firewalls" between individual packet flows, i.e.
  - A flow source that generates packets at a rate much higher than its reserved rate, may take up idle capacity
  - but it cannot affect the throughput rates guaranteed to other flows

# Virtual Clock server

- Each flow *f* has a "virtual clock", *priority*(*f*), which is zero initially and updated whenever a new packet in flow *f* arrives

- Let *p* denote a packet in flow *f*, with length *l(p)* bits and arrival time, *A(p)* ( ≥ 0). Upon its arrival,

$$priority(f) \leftarrow \max\{priority(f), A(p)\} + \frac{l(p)}{r^f}$$

The new value of *priority*(*f*) is assigned to packet *p* as its virtual clock value, denoted by *P(p)*

# Virtual Clock server (cont.)

- *priority (f)* holds the virtual clock value of the most recent packet arrival of flow *f*
  - Virtual clock values of packets are determined by the sequence of packet arrival times and their service times
- Whenever the server is ready for another packet, the packet among all flows with the smallest virtual clock value is selected
  - *FCFS within a flow*
  - non-preemptive

# Properties of VC server

- Each flow *f* is allocated a reserved service rate, $r^f$
  - *The number of flows is limited* (admission control)
- A misbehaving flow source that generates packets at a rate higher than its reserved rate, may take up idle capacity
  - but it cannot affect the throughput rates guaranteed to other flows

**No consideration of delay guarantee or bound in the original paper**

# References


- A. Demers, S. Keshav, S. Shenker, "Analysis and simulation of a fair queueing algorithm" *Proceedings of ACM SIGCOMM*, 1989.
  - Paper is about fair queueing and flow control. Weights are not mentioned until the last page in a concluding remark.
- A. Parekh and R. Gallager, "A generalized processor-sharing approach to flow control in integrated services networks: the single node case" *IEEE/ACM Trans. on Networking*, June 1993.
- Lixia Zhang, "Virtual clock: a new traffic control algorithm for packet-switching networks," *Proc. of ACM SIGCOMM*, 1990.

## The end