COMP/ELEC 429/556
Introduction to Computer Networks

Weighted Fair Queuing

Some slides used with permissions from Edward W. Knightly, T. S. Eugene Ng, Ion Stoica, Hui Zhang

Critical Features of TCP

- Increase rate until packet loss
  - What's the problem?
- Use loss as indication of congestion
  - What's the problem?
- Slow start to probe for initial rate
  - What's the problem?
- AIMD mechanism oscillates around proper rate
  - What's the problem?
- Relies on AIMD behavior of end hosts
  - What's the problem?

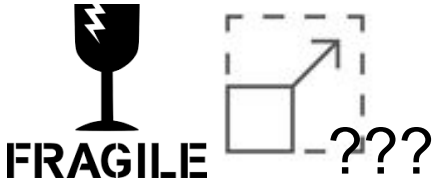
Some Answers

- Increase rate until packet loss
  - Drives network into congestion
  - High queuing delay, inefficient
- Use loss as indication of congestion
  - Cannot distinguish congestion from packet corruption
- Slow start to probe for initial rate
  - Bad for short lived flows (e.g. most Web transfers, a lot of Internet traffic is web transfer)
- AIMD mechanism oscillates around proper rate
  - Rate is not smooth
    - Bad for streaming applications (e.g. video)
  - Inefficient utilization
- Relies on AIMD behavior of end hosts for fairness
  - People can cheat (not use AIMD)
  - People can open many parallel connections

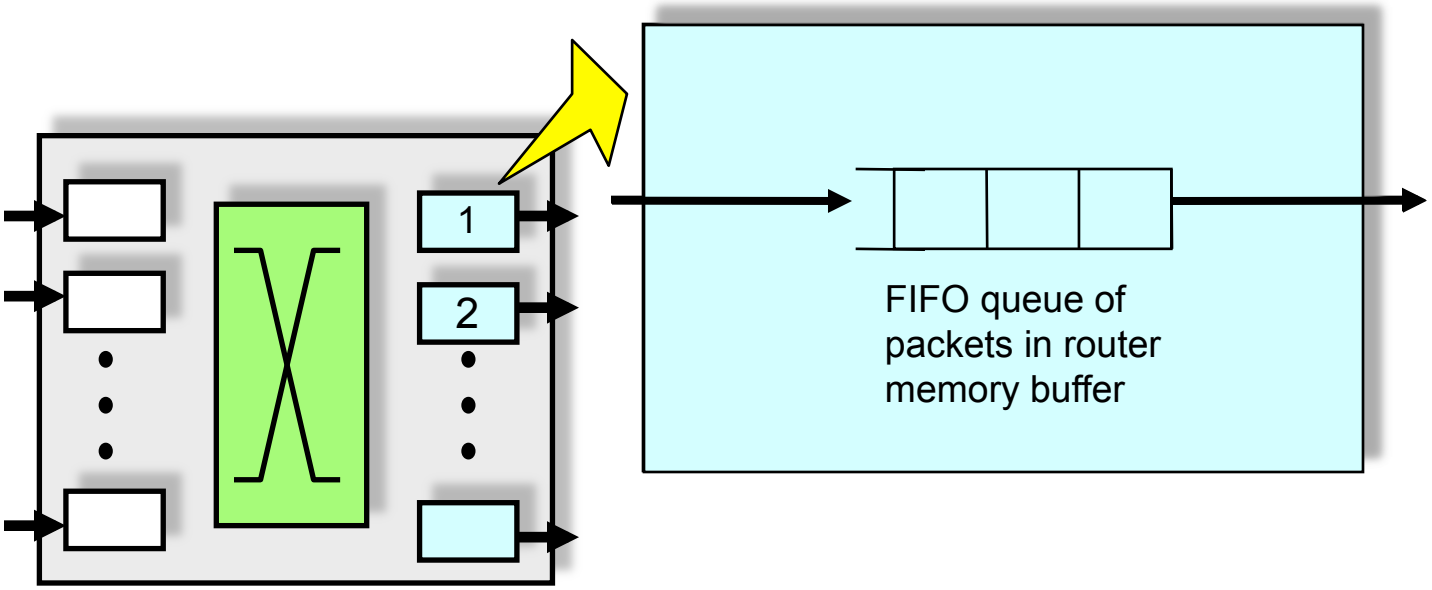
Can Routers Provide Bandwidth Guarantee to a Traffic Flow?

- If so, then sender can request for a bandwidth guarantee and knows exactly what rate to send at
  - No packet loss, no congestion
  - No need for probing bandwidth (slow start)
  - No AIMD oscillation
  - No cheating

- The answer is yes, but it'll add complexity to routers



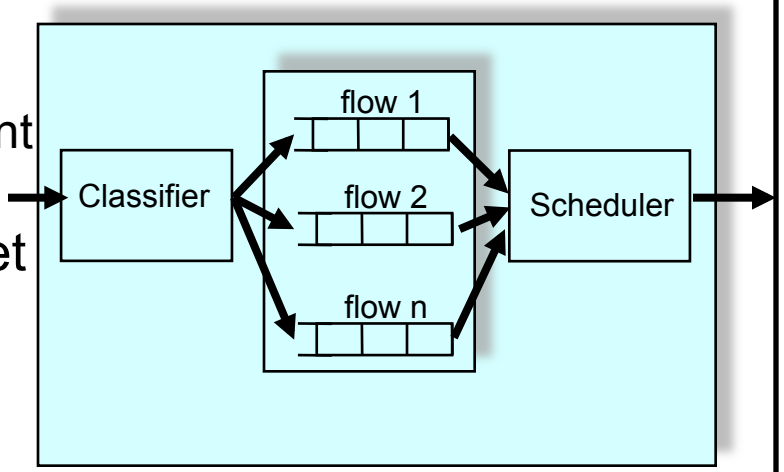
Guaranteeing Performance Requires Flow Isolation



Classifier

- A “flow” is a sequence of packets that are related
- Classifier takes a packet and matches it against flow definitions to decide which flow it belongs

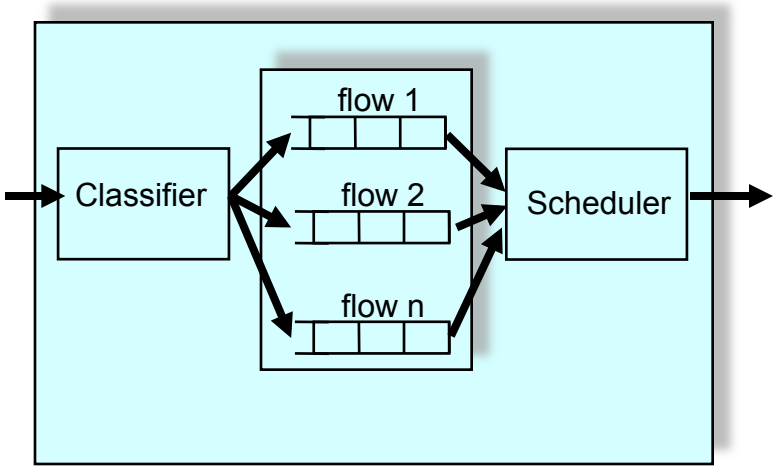
- Examples:
  - All TCP packets from Eugene’s web browser on machine A to web server on machine B
  - All packets from Rice
  - All packets between Rice and CMU
  - All UDP packets from Rice ECE department



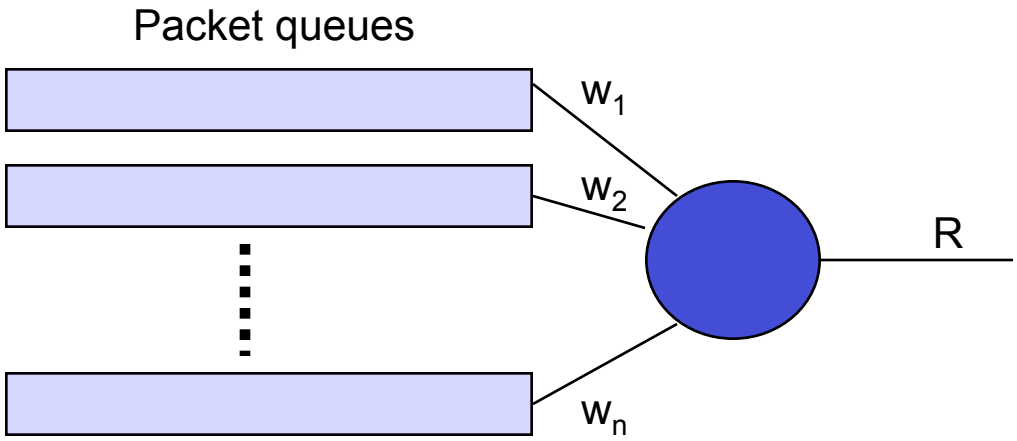
- A flow may be defined by bits in the packet
  - source/destination IP address (32 bits)
    - or address prefix
  - source/destination port number (16 bits)
  - protocol type (8 bits)
  - type of service (4 bits)
  - even bits beyond TCP and IP headers could be used

Scheduler

- Decides how the output link capacity is shared by flows
- A chance to be smart: Transmission of packets held in queues can be *scheduled*
  - Which stored packet goes out next? Which is more "important"?
  - Impacts quality of service



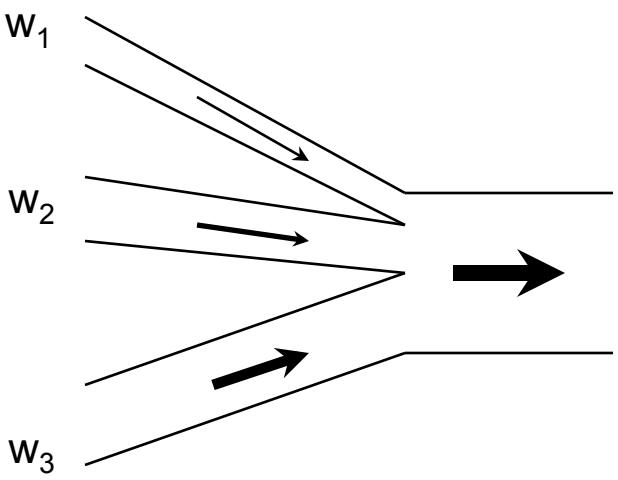
What is Weighted Fair Queuing?



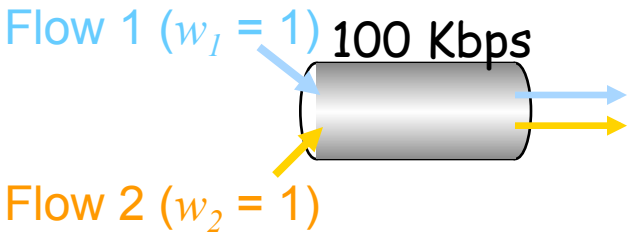
- A mathematical model for flow scheduling
- Each flow i given a weight (importance) w_i
- WFQ guarantees a minimum service rate to flow i
  - $r_i = R * w_i / (w_1 + w_2 + ... + w_n)$
  - Implies isolation among flows (one cannot mess up another)

What is the Intuition? Fluid Flow

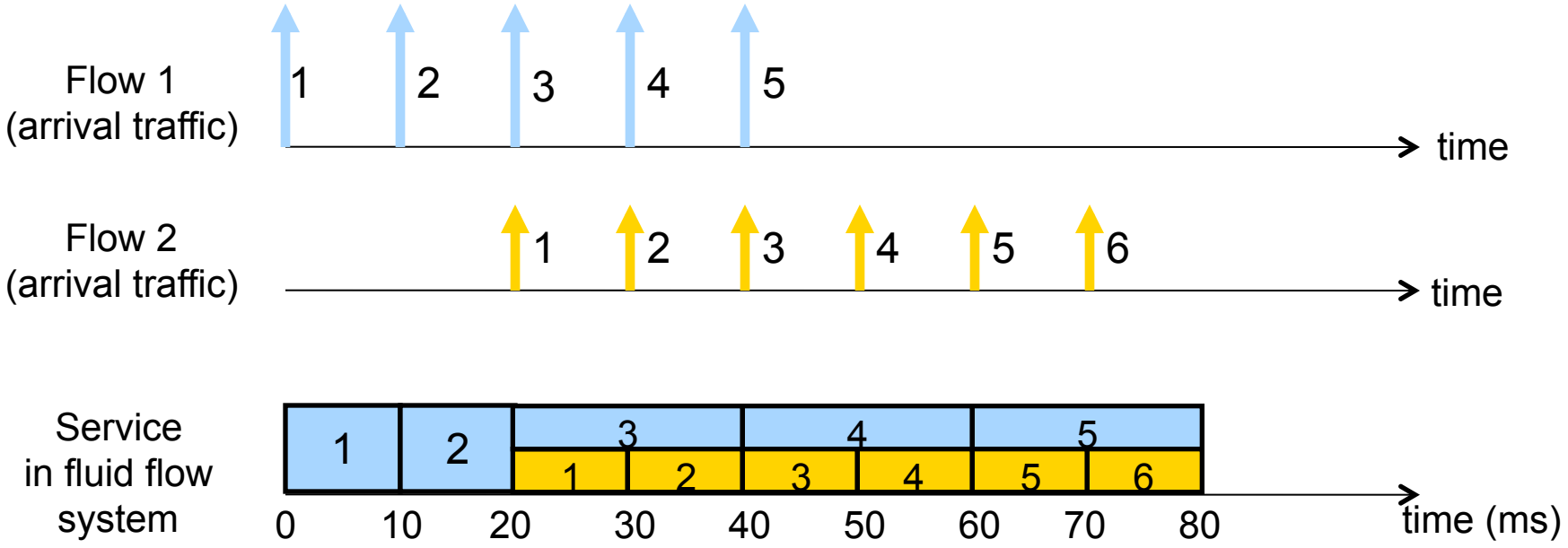
water pipes



Fluid Flow System: Example 1

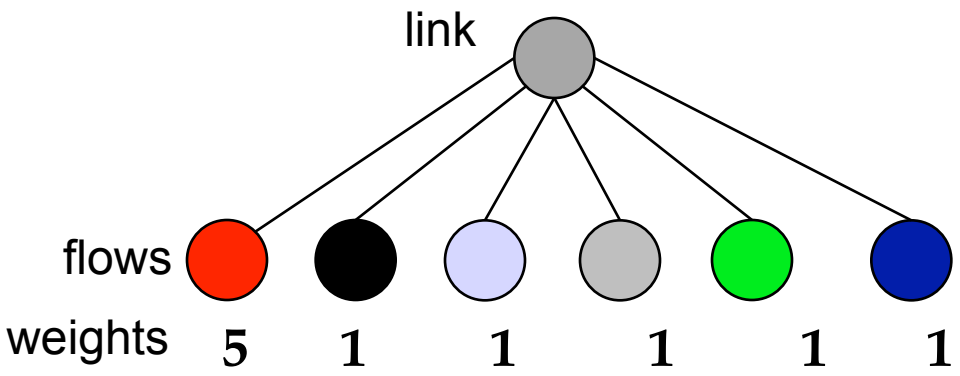


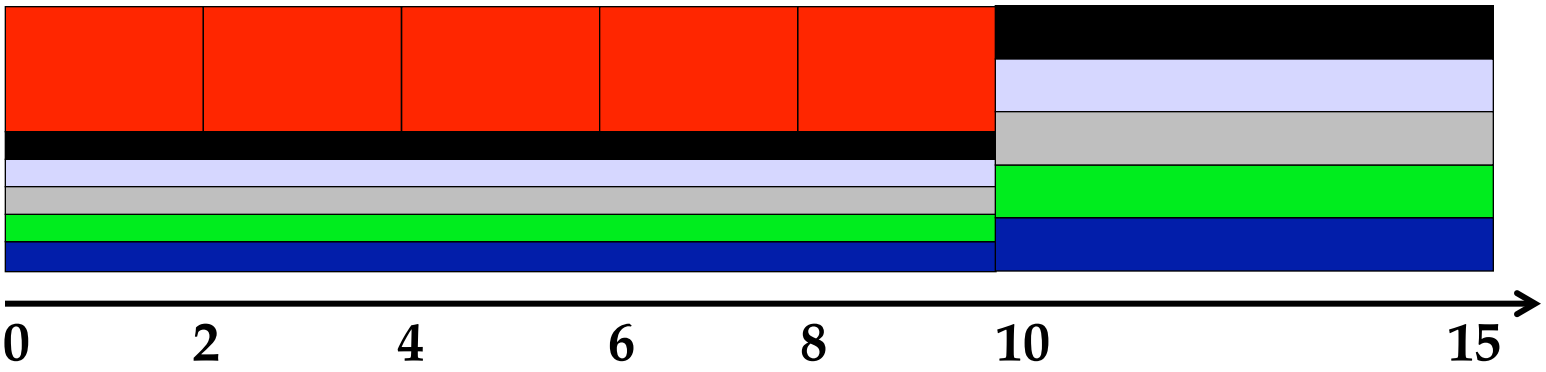
	Packet Size (bits)	Packet inter-arrival time (ms)	Arrival Rate (Kbps)
Flow 1	1000	10	100
Flow 2	500	10	50



Fluid Flow System: Example 2

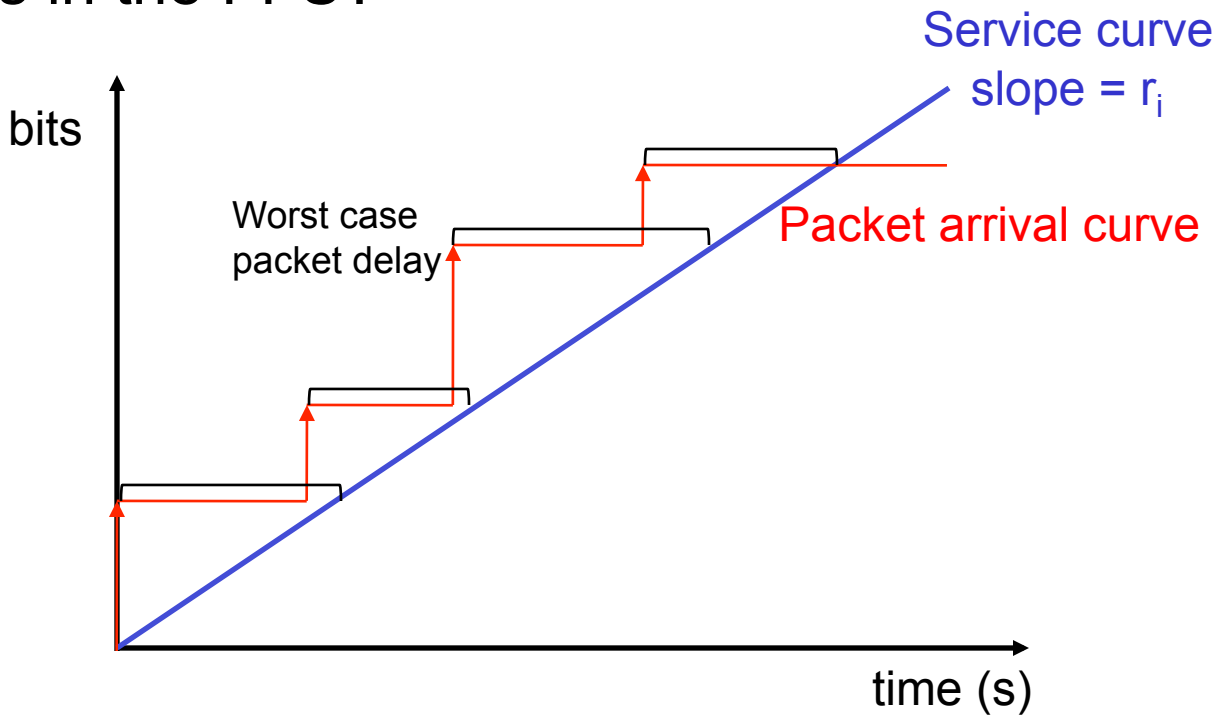
- Red flow has packets backlogged between time 0 and 10
  - Backlogged flow → flow's queue not empty
- Other flows have packets continuously backlogged
- All packets have the same size





Service Rate and Packet Delay in Fluid Flow System

- WFQ guarantees a minimum service rate to flow i
  - $r_i = R * w_i / (w_1 + w_2 + ... + w_n)$
- What worst case delays are experienced by individual packets in the FFS?

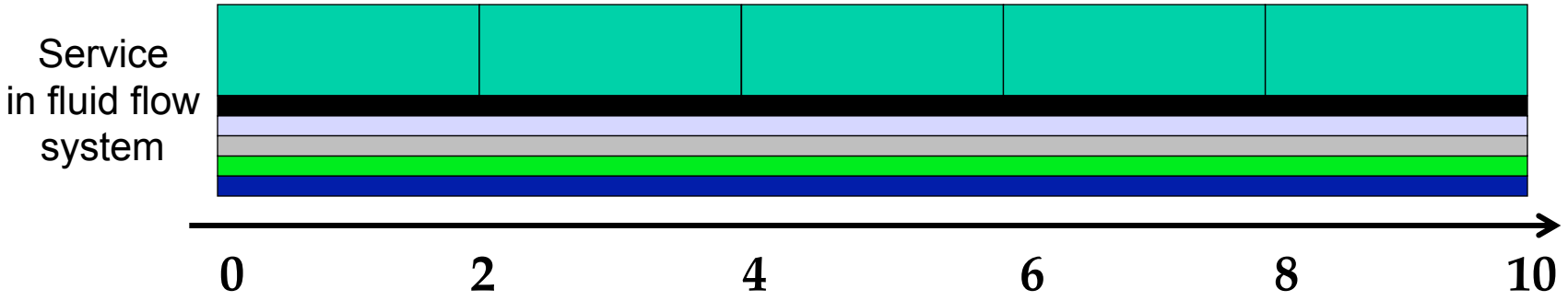


Implementation in Packet System

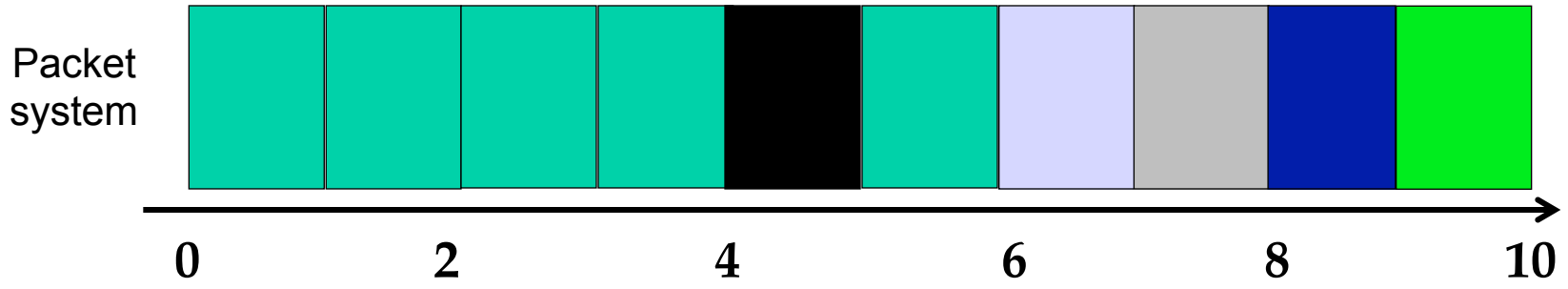
- Packet (Real) system: packet transmission cannot be preempted. Why?
- Solution: serve packets in the order in which they would have finished being transmitted in the fluid flow system

Packet System: Example 1

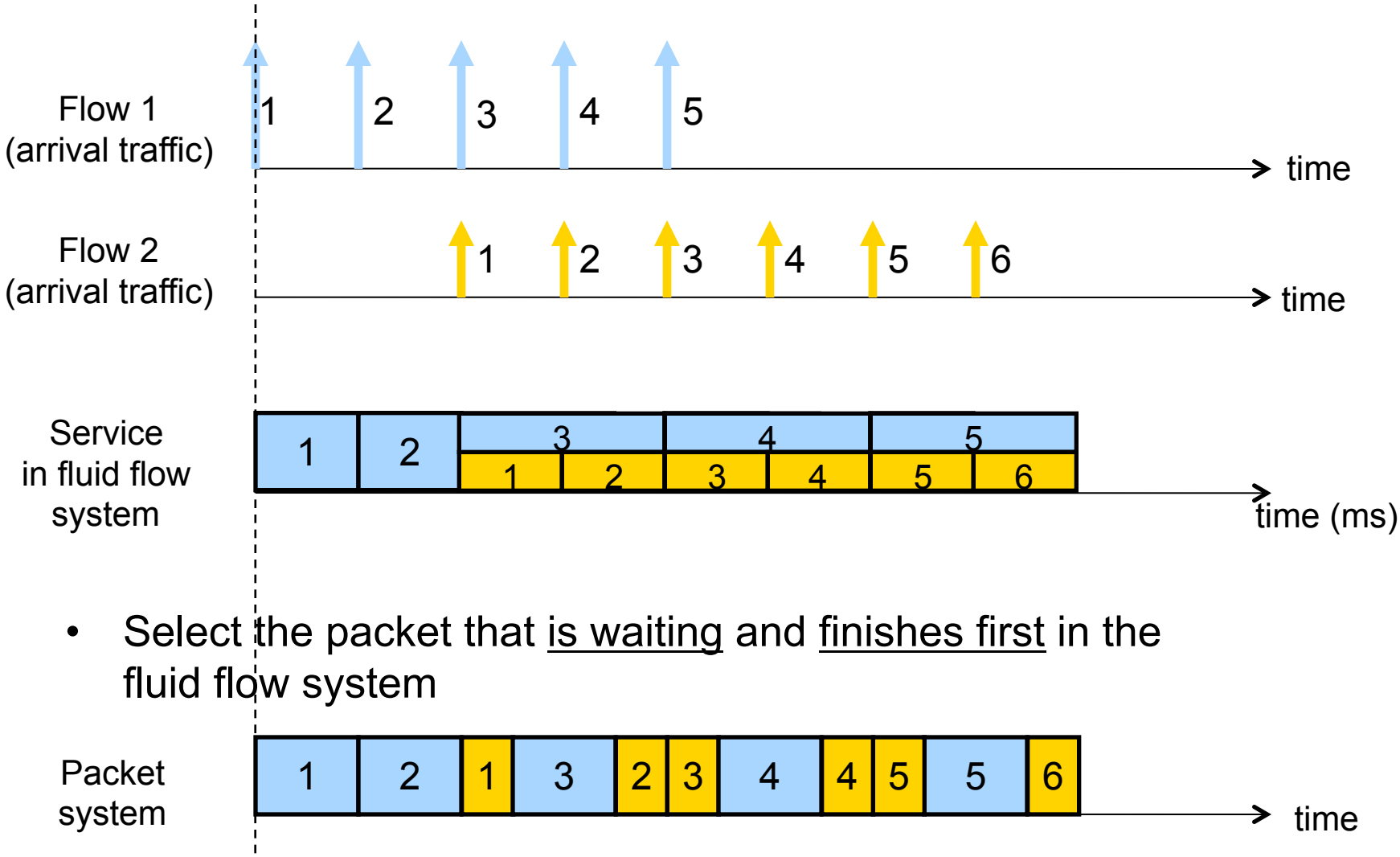
Assume for this example all packets are waiting from time 0



- Select the packet that is waiting and finishes first in the fluid flow system



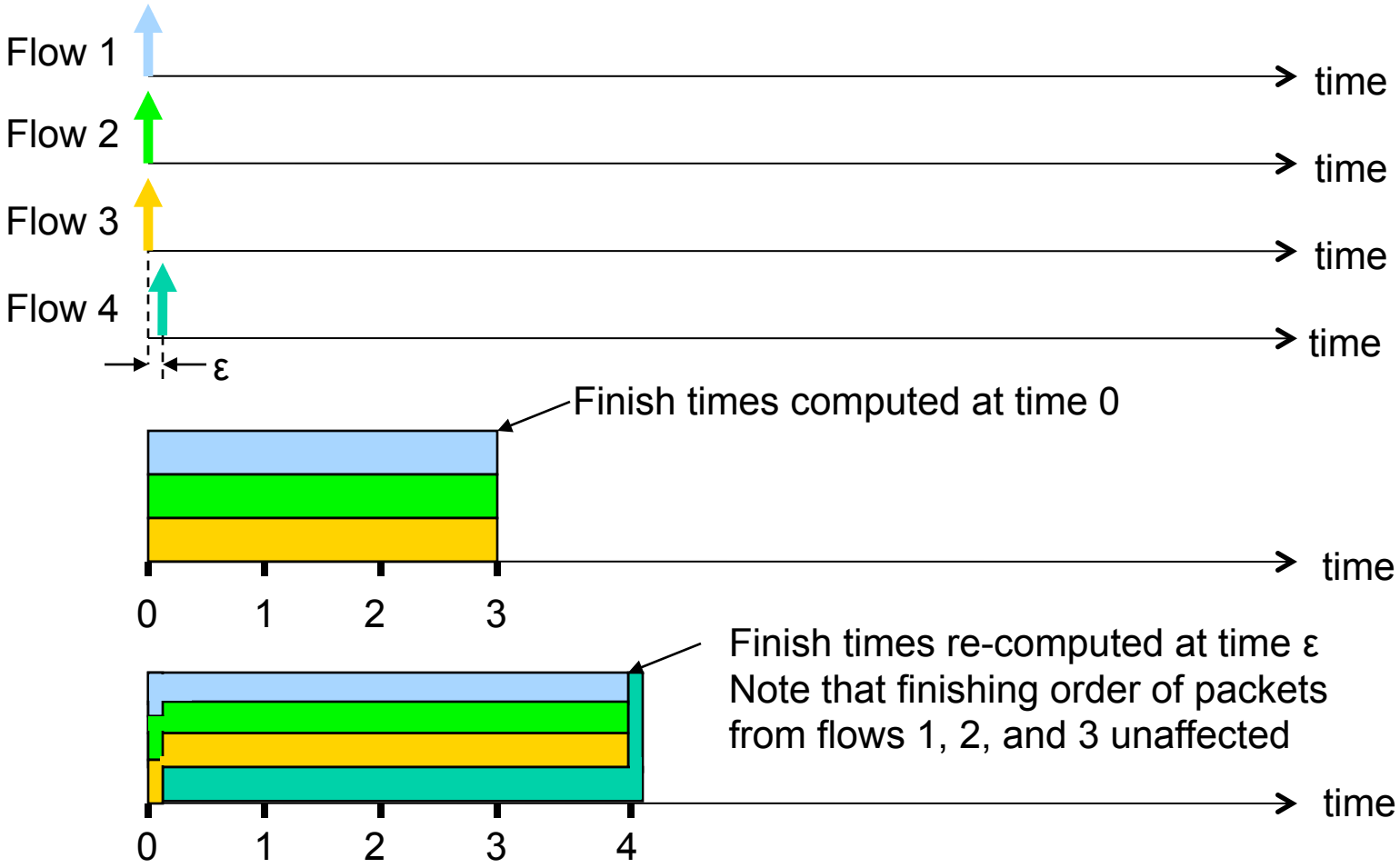
Packet System: Example 2



- Select the packet that is waiting and finishes first in the fluid flow system

Implementation Challenge

- Four flows, each with weight 1

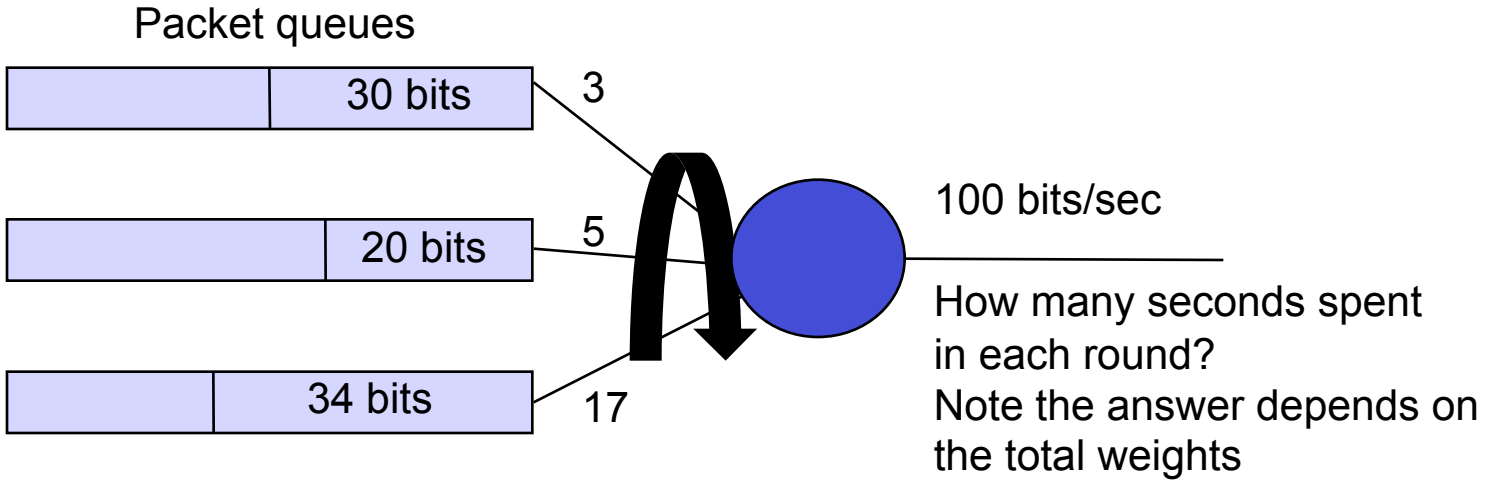


Implementation Challenge

- Need to compute the finish time of a packet in the fluid flow system…
- … but the finish time may change as new flows arrive!
- Need to update the finish times of all packets that are in service in the fluid flow system when a new flow arrives
  - But this is very expensive; a high speed router may need to handle hundred of thousands of flows!
- The new finish times don't affect ordering of packets that are in service in the fluid flow system (a lot of work for nothing)
- Can we capture ordering without using real world finish times??

Bit-by-Bit Round Robin Insight

- If flows can be served one bit at a time, the fluid flow system can be approximated by bit-by-bit weighted round robin
  - During each round from each flow that has data to send, send a number of bits equal to the flow's weight
- Each packet therefore requires a fixed number of rounds to finish (depends only on weight and packet size); the finishing round number does not change even when new flows arrive



Solution: Virtual Time

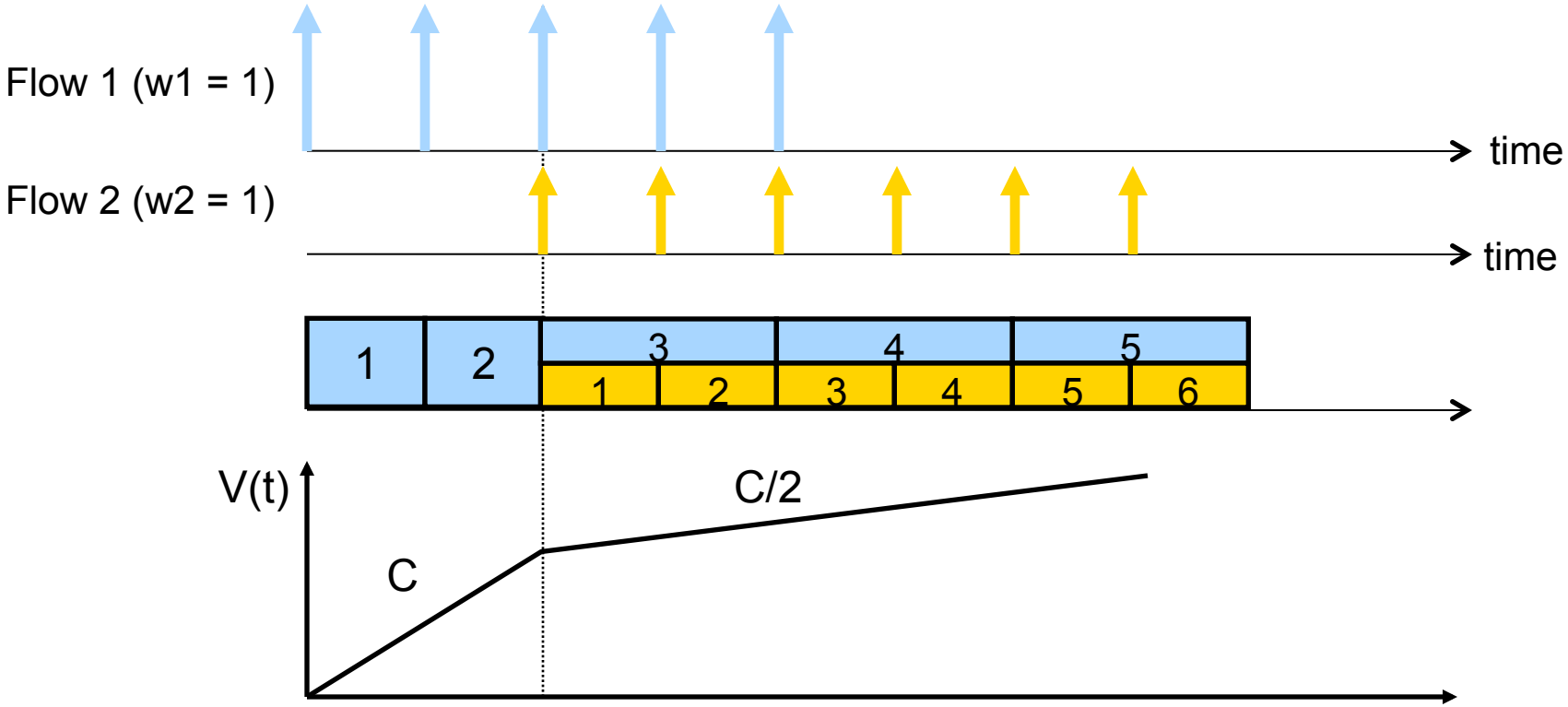
- Solution: instead of the packet finish time, maintain the round # when a packet finishes (virtual finishing time)
  - Virtual finishing time doesn't change when a new flow arrives
  - Packet ordering based on virtual finishing time is the same as that based on real finishing time.
- Need to compute *system virtual time function* V(t)
  - index of the round in the bit-by-bit round robin scheme at time t

Example

- All flow weight = 1
- Suppose each packet is 1000 bits, so takes 1000 rounds to finish. So, first packets of F1, F2, F3 finish at virtual time 1000
- When packet F4 arrives at virtual time 1 (after one round), the virtual finish time of packet F4 is 1001
- But the virtual finish time of packet F1,2,3 remains 1000
- Finishing order is preserved

Computing System Virtual Time (Round #): V(t)

- V(t) increases inversely proportionally to the sum of the weights of the backlogged flows
- Since round # increases slower when there are more flows to visit each round.



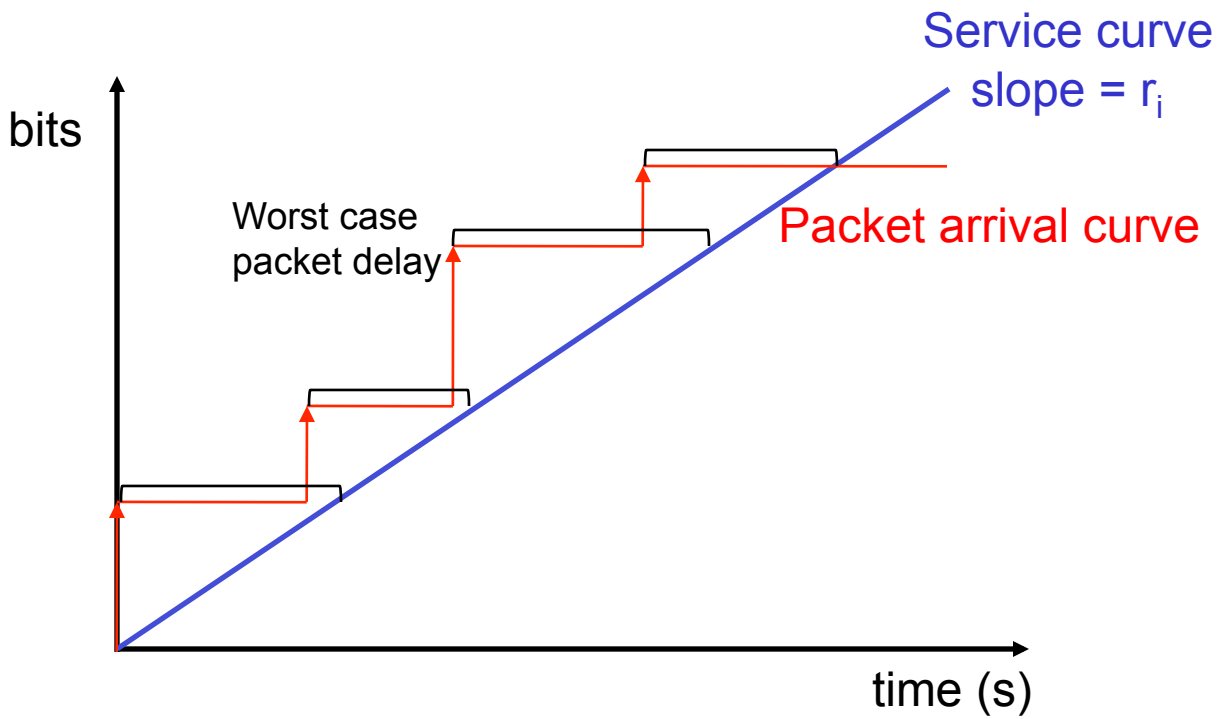
Weighted Fair Queuing Implementation

- Define
  - F_i^k virtual finishing time of packet k of flow i
  - a_i^k arrival time of packet k of flow i
  - L_i^k length of packet k of flow i
  - w_i weight of flow i
- The virtual finishing time of packet $k+1$ of flow i is

$$F_i^{k+1} = \max(V(a_i^{k+1}), F_i^k) + L_i^{k+1} / w_i$$

- Smallest virtual finishing time first scheduling policy

Recall in the Fluid Flow System



Properties of WFQ Implementation

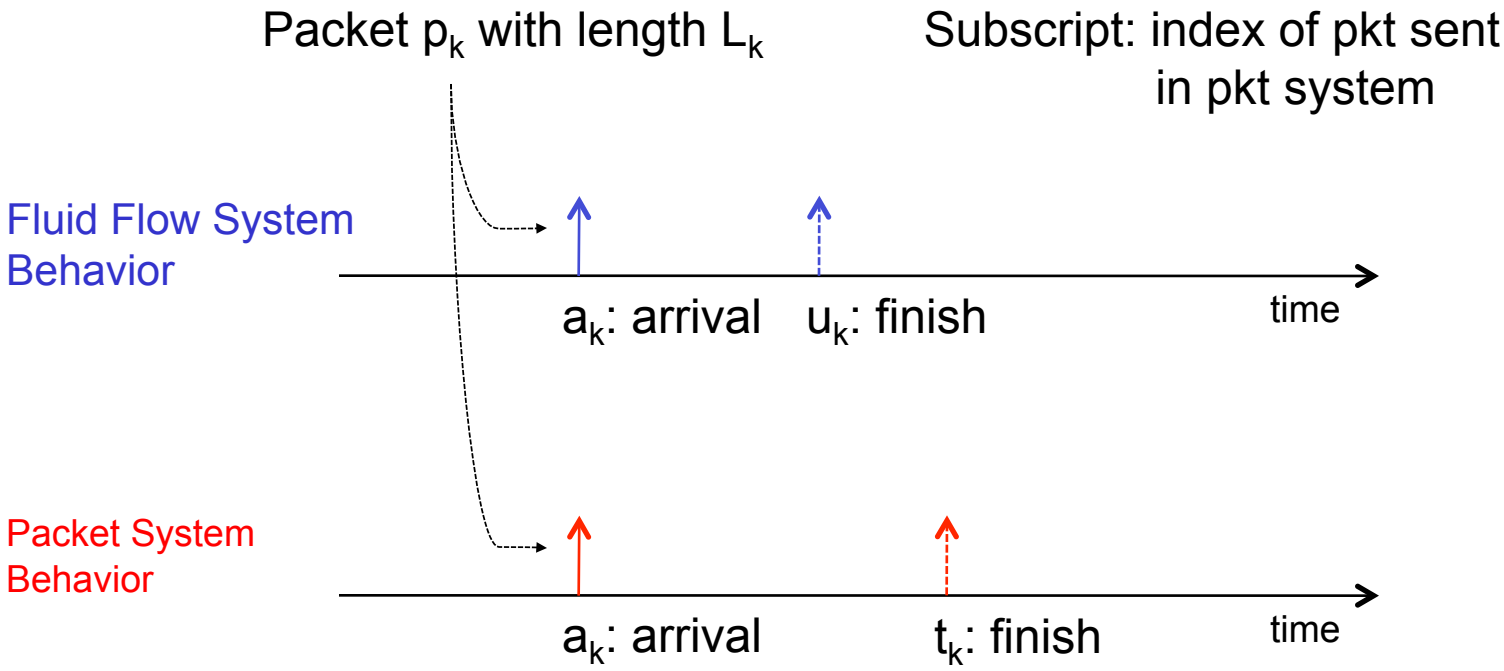
- Theorem: WFQ guarantees that any packet is finished within *max_packet_length/link_rate* of its finish time in the fluid flow system
  - Thus WFQ can guarantee bandwidth and delay
    - Weights assigned appropriately and admission control is performed
    - Packets paced by sender according to guaranteed bandwidth
  - Another way to use WFQ is to assign all flows the same weight and perform no admission control
    - Every flow gets a "max-min" fair share, provides performance isolation among flows
- See proof at the end of the slide deck

Internet Today

- FIFO queues are used for most network traffic
  - No classifier, no scheduler, best-effort
- Sophisticated mechanisms tend to be more common near the “edge” of the network
  - E.g. At campus routers
  - Use classifier to pick out BitTorrent packets
  - Use scheduler to limit bandwidth consumed by BitTorrent traffic

Remainder of this set of slides are for your reference only. They will not be needed for assignments.

Proof (1)



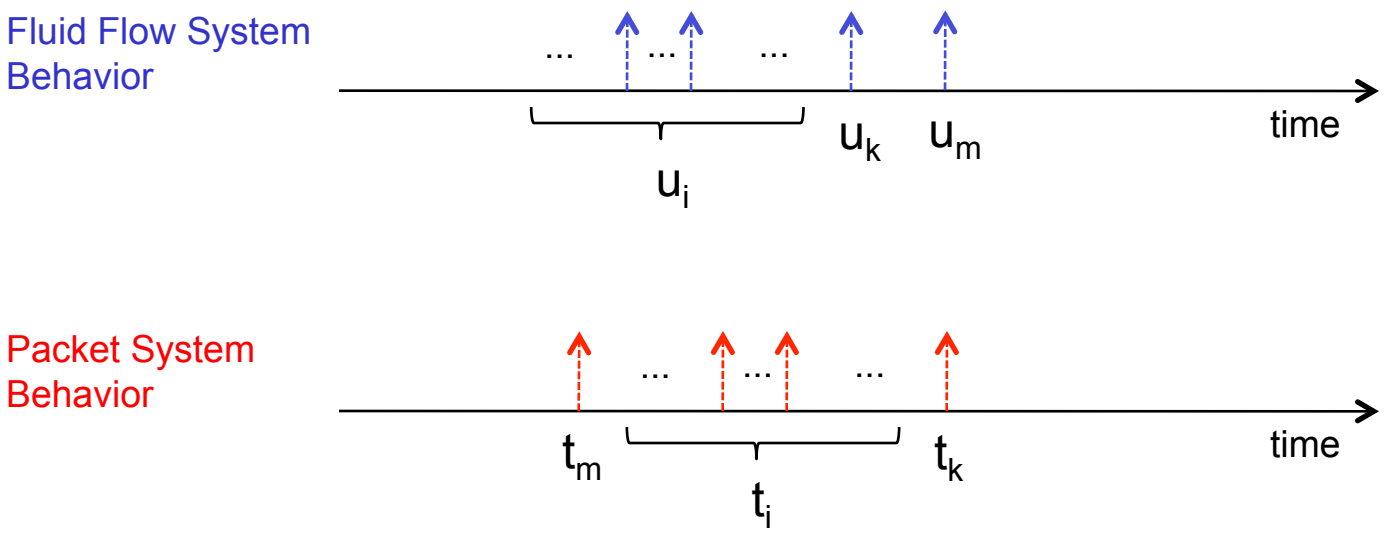
Want to prove: $t_k \leq u_k + \frac{L_{\max}}{r}$

L_{max}: largest allowed pkt
r: network link rate

Proof (2)

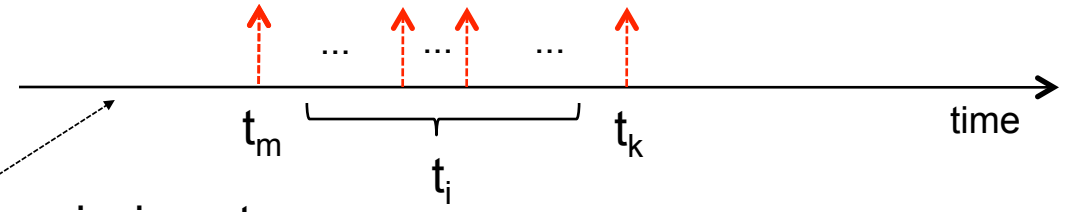
Let m be the largest packet index such that $u_m > u_k$ but $t_m < t_k$

More generally, we have $u_m > u_k \geq u_i$ and $t_m < t_i < t_k$



Proof (3)

Packet System Behavior



p_m started transmission at

$$t_m - \frac{L_m}{r}$$

$$u_m > u_k \geq u_i \Rightarrow a_i, a_k > t_m - \frac{L_m}{r}$$

$$u_k \geq (\ldots + \frac{L_i}{r} + \ldots + \frac{L_k}{r}) + t_m - \frac{L_m}{r} = t_k - \frac{L_m}{r}$$

Thus $t_k \leq u_k + \frac{L_{\max}}{r}$ Q.E.D.

Another Insight

Recall m is the largest packet index such that $u_m > u_k$ but $t_m < t_k$

If no such p_m exists, then simply $t_k \leq u_k$

That is, the packet system can be ahead of the fluid flow system by an unbounded amount