

Quality of Service for Packet Networks

Real-Time and Embedded Systems (M)

Lecture 17

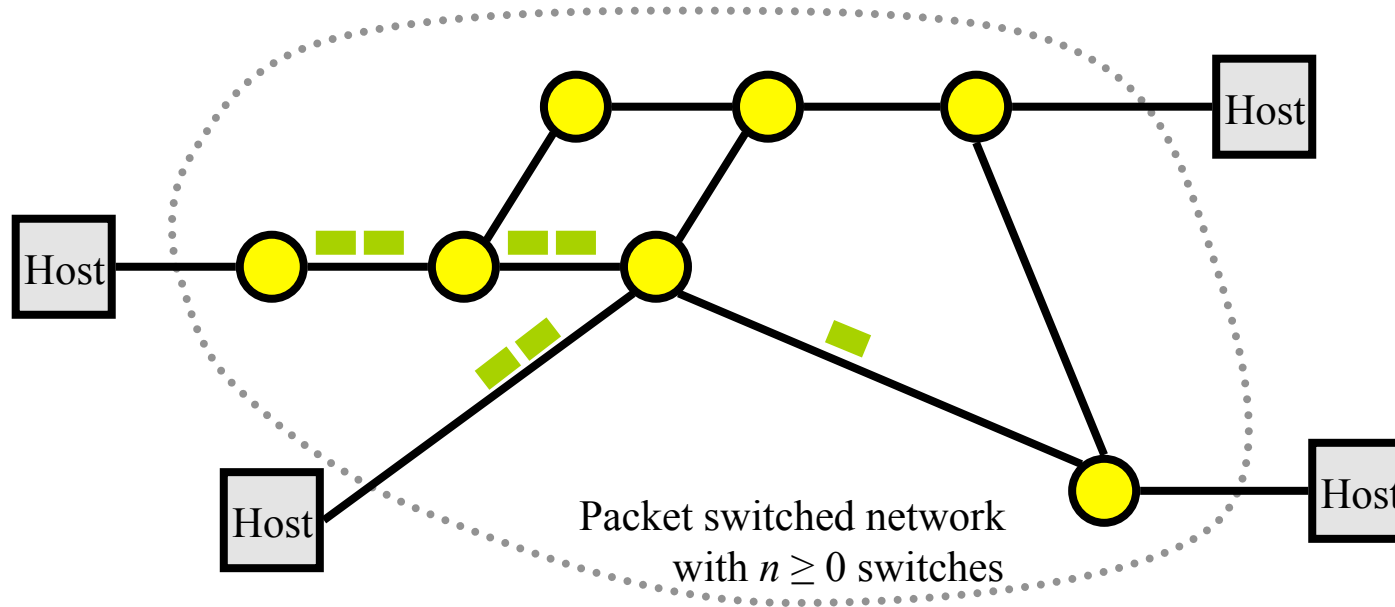
UNIVERSITY
of
GLASGOW



Lecture Outline

- Best effort versus enhanced services
 - Queuing disciplines
 - Weighted fair queuing and variants
 - Weighted round robin
 - Resource reservation protocols
 - RSVP
-
- Material corresponds to parts of chapters 7 and 11 of Liu's book

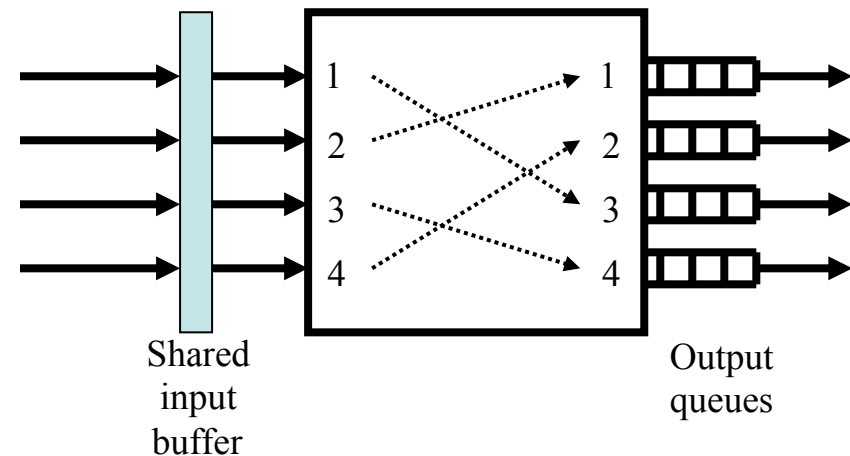
Model of Packet Switched Networks



Links have constant *propagation delay*

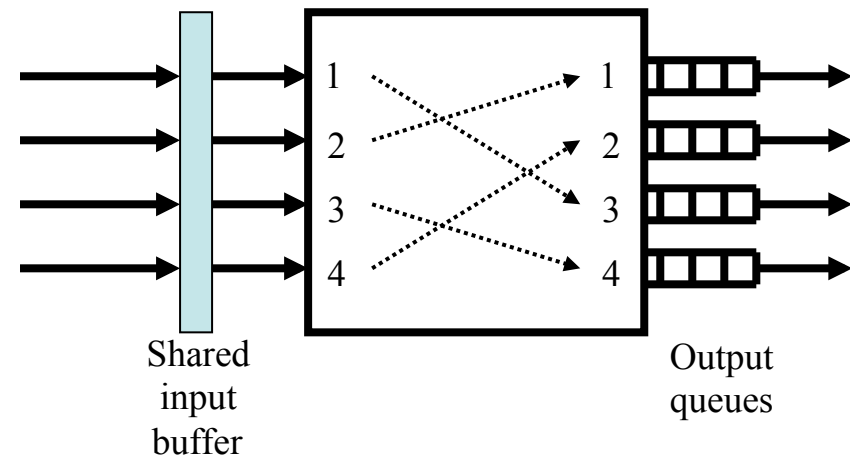
Switches are *output buffered* with packets queued for transmission if output link busy

Choice of *job scheduling algorithm* on the output link is critical for real time traffic



Best Effort versus Enhanced Service

- Best effort networks use a single output queue for each link
 - FIFO with drop tail
 - FIFO with REDand don't control the output queuing
- Uncontrolled best effort networks are inexpensive, but don't provide rate guarantees or control the jitter
- Enhanced service packet networks provide this control, and are better suited to real-time traffic
 - Packets in the output queues are scheduled for transmission to affect some policy, rather than in FIFO order



How to Implement Enhanced Service?

- To schedule packets according to some policy, the policy must be communicated to the network, and the network must perform admission control to ensure that policy constraints can be met
- Implies the network must implement:
 - A packet scheduling algorithm
 - To prioritise certain classes of traffic
 - To manage the output queues
 - Admission control
 - To determine if the signalled flows can be supported
 - A signalling protocol
 - To communicate the stream characteristics to the network
 - Flow specification
 - Required performance

Service Disciplines for Enhanced Services

- The combination of scheduling algorithm and acceptance test is a *service discipline*
- Used to control jitter and packet rate
 - Ensure flows receive their *proportional fair share* of capacity
 - Rates controlled to allocate capacity to flows proportionally, according to policy
 - Flows serviced regularly, to avoid starvation
 - Ensure *timing isolation* between flows
 - Partly as a side-effect of rate control
 - Some algorithms perform explicit jitter control, preserving the traffic pattern – inter-packet spacing – when forwarding traffic
- Algorithms can be *rate allocating* or *rate controlled*
 - Rate controlled algorithms give each flow an allocated rate, and never let flows exceed their rate
 - Rate allocating algorithms give each flow an allocated rate, but let flows exceed their rate if there is spare capacity

Priority Queuing Algorithms

- Two priority queuing algorithms are widely implemented:
 - Weighted fair queuing (WFQ)
 - Frame based WFQ
 - Weighted round robin (WRR)

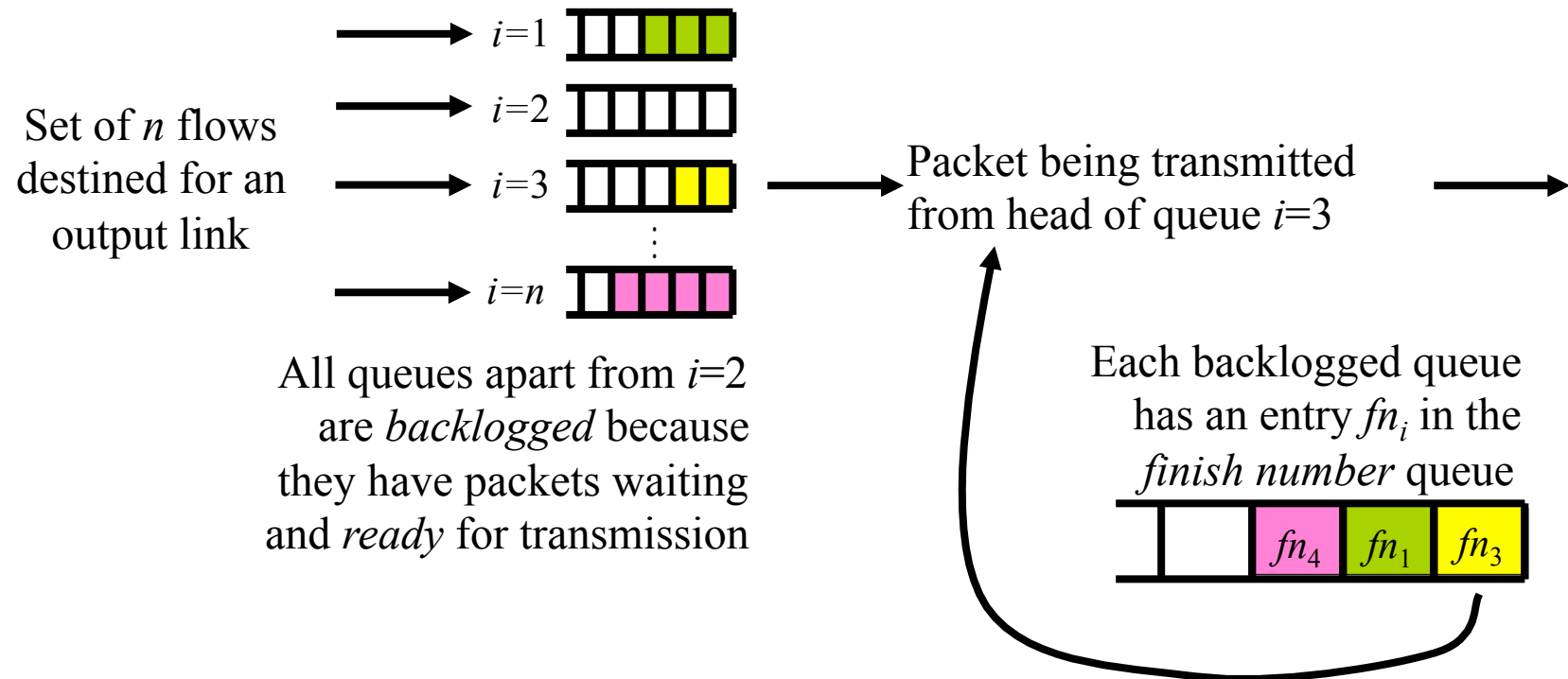
[See also lecture 8]

Weighted Fair Queuing

- “Packet-by-packet generalized processor-sharing algorithm”
 - A rate allocating service discipline; provides each flow with at least its proportional fair share of link capacity; isolates timing between flows
- Definitions:
 - A packet switch has several inputs, feeding to an output link shared by n established flows
 - Each flow, i , is allocated a fraction \tilde{u}_i of the link
 - Total bandwidth allocated to all n connections is $U = \sum_{i=1}^n \tilde{u}_i$ where $U \leq 1$
- Assumed that an acceptance test rejects connections that would cause requested bandwidth to exceed the available bandwidth
- Define the “finish number”, fn_i , to represent job completion times

Weighted Fair Queuing: Packet Scheduling

- Each link of the packet switch is output buffered
- Output buffers conceptually comprise two sets of queues:
 - A set of FIFO queues for each of the n flows
 - A priority ordered shortest finish number (SFN) queue
- Entry at head of SFN queue indicates the FIFO queue to service



Weighted Fair Queuing: Packet Scheduling

- As a packet becomes ready on a FIFO queue, its finish number is calculated, and the SFN queue is updated
 - Currently transmitting packet never pre-empted, even if the finish number of the newly ready packet would place it at the head of the SFN queue
- When a packet completes transmission, it is removed from the head of the FIFO and SFN queues
 - If the FIFO queue is still backlogged, the SFN queue is updated with the finish number of the newly ready packet
 - The packet from the queue referenced by the entry at the head of the SFN queue begins transmission
- Key is the calculation of the finish number for each packet as it becomes ready on a backlogged queue

Weighted Fair Queuing: Finish Numbers

- Define:
 - The total bandwidth of all backlogged flows, U_b
 - The finish number of the link, FN
 - The current time, t , and the previous time, t_{-1} , when FN and U_b updated
- Computing the finish number when the link becomes active:
 - The link is idle: $FN=0$, $U_b=0$, $t_{-1}=0$ and all $fn_i = 0$
 - A packet of length e arrives and starts a link busy interval on link i
 - Compute $U_b = U_b + \tilde{u}_i$ and $fn_i = fn_i + e/\tilde{u}_i$
 - Set $t_{-1} = t$
 - Insert entry (fn_i, i) in the SFN queue
 - Intuition: finish number of the first packet set to transmission delay for the job, adjusted by the fraction of the link used

Weighted Fair Queuing: Finish Numbers

- Computing subsequent finish numbers during link busy interval
 - If a packet arrives on a previously idle flow, i
 - Increment FN by $(t-t_{-1})/U_b$
 - Compute $fn_i = \max(FN, fn_i) + e/\tilde{u}_i$
 - Insert entry (fn_i, i) in the SFN queue
 - Set $t_{-1} = t$ and increment $U_b = U_b + \tilde{u}_i$
 - When the transmission of a packet on flow i completes
 - If the connection remains backlogged
 - Compute $fn_i = fn_i + e/\tilde{u}_i$ where e is the length of the newly ready packet
 - Insert entry (fn_i, i) in the SFN queue
 - If the connection becomes idle
 - Increment FN by $(t-t_{-1})/U_b$
 - Set $t_{-1} = t$ and decrement $U_b = U_b - \tilde{u}_i$
 - Intuition: finish number fn_i represents deadline when a packet on flow i will be transmitted

Weighted Fair Queuing: Properties

- Complex algorithm to calculate finish number, and determine the transmission order of packet – what is the benefit?
- Can control latency and jitter, can isolate traffic flows
 - Bounds on per-hop and end-to-end latency for traffic
 - Guaranteed network capacity

Weighted Fair Queuing: Per-Hop Latency

- Delay between time a packet becomes ready (when it reaches the head of the FIFO queue of its connection) and when transmission completes is *latency*, L_i
 - Blocking time due to the WFQ algorithm itself, ignoring queuing delay
- It has been proved that $L_i < e_i/\tilde{u}_i + 1$
where: e_i is the normalised maximum packet length,
 \tilde{u}_i is the fraction of the link assigned to this flow
 - First term: time taken to transmit largest packet
 - Second term: blocking due to non pre-emptive schedule
- Because of the rate control behaviour of WFQ, this bound is independent of other traffic on the output link

Weighted Fair Queuing: Total Per-Hop Delay

- Total per-hop delay, $W_i(1)$, for a packet of length e is equal to the sum of latency, calculated previously, and queuing delay
- To predict queuing delay, you need to know arrival pattern
 - From queuing theory, it can be shown that the queuing delay can be unbounded even if allocated bandwidth, \tilde{u}_i , equals the actual bandwidth of the flow, u_i , if there is no constraint on the arrival pattern

- Assume arrivals fit a (u_i, E_i) leaky bucket constraint and the flow is allocated a sufficient fraction $\tilde{u}_i \geq u_i$ of the link
 - Matches periodic, and many sporadic, isochronous flows

Capacity

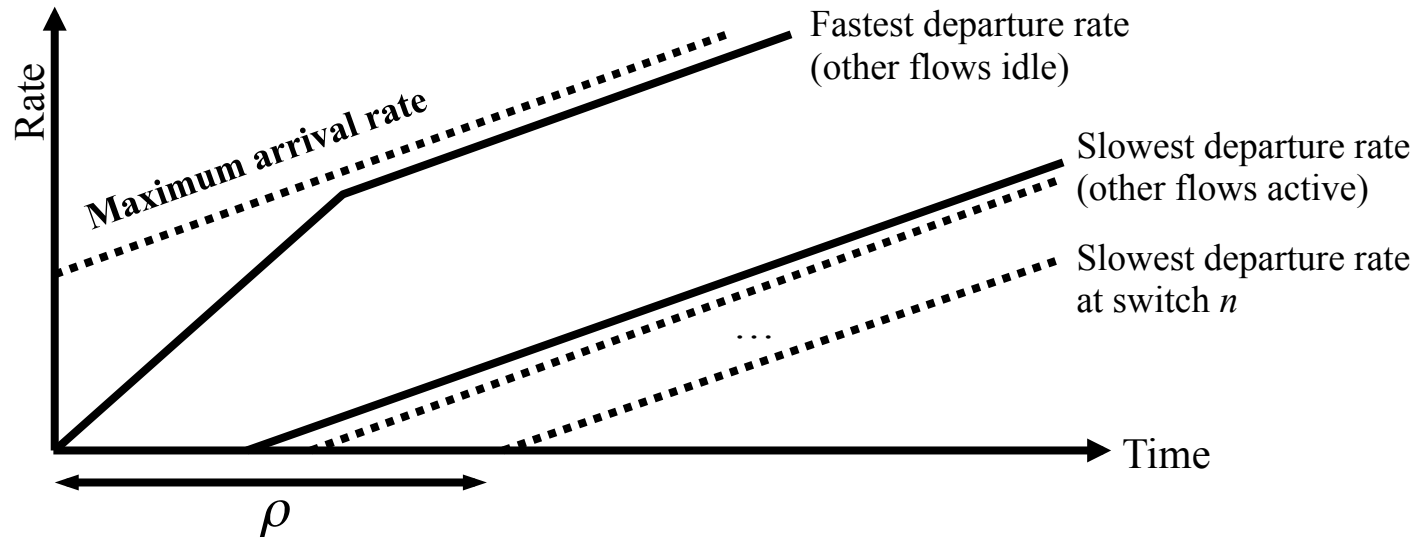
Leak rate

- Can be proven that $W_i(1) = (E_i + e) / \tilde{u}_i + 1$
 - Latency term + E_i to represent queuing delay

End-to-End Delay of WFQ

- If we know per-hop delay, can we model end-to-end delay?
- Assume a homogeneous network:
 - A connection i with rate u_i traverses ρ switches
 - Traffic is initially shaped to match a (u_i, E_i) leaky bucket
 - Intermediate switches perform WFQ, but no traffic shaping
 - All links have the same capacity, and the connection is allocated the same fraction $\tilde{u}_i = u_i$ of bandwidth

End-to-End Delay of WFQ



- Making worst case assumptions, maximum arrival rate at switch n is slowest departure rate at switch $n-1$; 1 unit of delay added due to non pre-emption at each hop
- Can derive $W_i(\rho) = \frac{E_i + \rho e}{\tilde{u}_i} + \rho$ when $\tilde{u}_i = u_i$
- Same as per-hop delay, but adjusted for the number of hops ρ

End-to-End Delay of WFQ

- Generalise: output links may have different transmission rates
- For switch j traversed by flow i
 - Assume flow i satisfies a leaky bucket (λ_i, E_i) at the first hop
 - Assume flow i is allocated $u_i = \lambda_i$
 - Let $e_{\max}(i, j)$ denote time taken to transmit largest packet of all flows sharing the output link with connection i

- Can show that $W_i(\rho) = \frac{E_i + \rho e}{\lambda_i} + \sum_{j=1}^{\rho} e_{\max}(i, j)$
 - As before, but adjusted for non pre-emption delays of variable rate/size packets at each hop

- Can also show that, maximum jitter is $\frac{E_i}{\lambda_i} + \sum_{j=i}^{\rho} e_{\max}(i, j)$

Weighted Fair Queuing

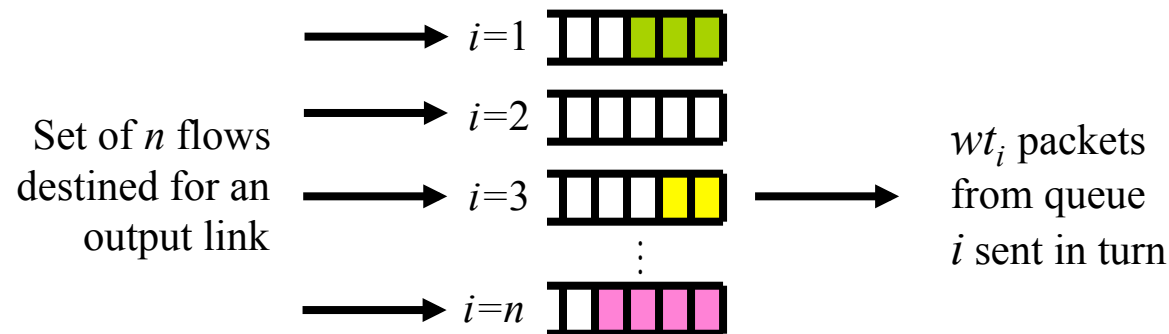
- A dynamic priority scheduling algorithms to ensure:
 - Each flow i gets at least a fraction u_i of the link bandwidth
 - Packets are scheduled fairly, and starvation is avoided
- Per-hop delay and end-to-end delay for a flow can be bounded, if the traffic pattern of the flow is known
 - Independent of the other flows in the network
- Compared to an uncontrolled packet network WFQ is complex, but can guarantee throughput, latency and jitter
 - Simplifies applications running on a WFQ network, since they can predict timing of message delivery

Priority Scheduling: Frame-based WFQ

- Weighted fair queuing can require $O(n)$ scheduling events per packet, where n is the number of flows
 - One calculation for each packet arrival and departure in the worst case
- Significant impediment to high performance implementation
- Would prefer $O(1)$ calculations per packet
- Frame-based WFQ approximates WFQ
 - Instead of recalculating the link finish number on each packet arrival and departure, use an approximation updated at most once per packet departure
 - The frame size is the interval between recalculations
- Trade-off some fairness for reduced complexity
- Book explains the algorithm...

Weighted Round Robin Scheduling

- In *round robin* scheduling, jobs are placed in a FIFO queue
 - The job at the head of the queue executes for one time slice
 - If it doesn't complete within the time slice, it is pre-empted and put at the back of the queue
 - There are n jobs in the queue, each job gets one slice every n time slots (that is, every *round*)
- A *weighted round robin* schedule extends this, to give each job i a weight wt_i
 - A job with weight wt_i executes for wt_i time slices each round
 - Length of the round equals $\sum_{i=1}^n wt_i$



WRR Scheduling: Throughput Guarantees

- Assume constant bit rate, periodic, flows: $M_i = (p_i, e_i, D_i)$
 - Minimum inter-arrival time of messages p_i
 - Size of each message e_i
 - Maximum acceptable end-to-end delay D_i
- Each round, if more than wt_i packets are backlogged on queue i , then wt_i packets are transmitted
 - Each flow is guaranteed wt_i slots each round
 - Rate allocating: may send more, if nothing else to transmit
- A design parameter is RL the maximum number of slots per round
 - At all times $\sum_{i=1}^n wt_i \leq RL$
 - Each flow is guaranteed a share wt_i/RL of the link capacity
 - Provided that:
 - $RL < p_{\min}$
 - $wt_i \geq e_i/(p_i/RL)$ (with appropriate rounding)

WRR Scheduling: End-to-End Delay Bounds

- Messages take at most e_i/wt_i rounds to complete
- Implies delay through first switch = $(e_i/wt_i)RL$
- At each subsequent switch, each round of packets arriving is sent in the next round
 - Implies one round delay at each hop
- Therefore, end-to-end delay for connection i passing through r switches can be bounded by:

$$W_i \leq (e_i/wt_i + r - 1)RL$$

- Can also be shown that jitter can be bounded by

$$\text{jitter} < p_i - e_i + (r - 1)(RL - 1)$$

for messages of size e_i with inter-arrival time of p_i

WRR Scheduling: Connection Establishment

- Why use a fixed round length RL ?
- Too costly to change the round length each time a new flow is established
 - Would require adjusting weights for all pre-existing flows
- With a fixed RL , connection establishment becomes:
 - Pass parameters (p_i, e_i, D_i) to each hop router
 - At each hop, the scheduler computes the weight, wt_i , required to support the new flow
 - If the sum of existing weights $< RL - wt_i$ the flow is accepted at that hop
 - If all hops accept, the flow is established

Weighted Round Robin Scheduling

- Flows are guaranteed capacity
- WRR scheduling is efficient to implement, since the scheduling decision is $O(1)$
 - Simply pick wt_i packets from the next queue
- End-to-end delay can be bounded
- Since the scheduler is rate allocating, jitter is not controlled but can be bounded

Resource Reservation Protocols

- Throughout the discussion of queuing algorithms, we have assumed that the required rate allocation, \tilde{u}_i , is known at each switch
- In a real packet network, hosts must inform the routers of the flow characteristics and required rate
- Implies a *resource reservation protocol* is needed
- Several issues to consider:
 - Scalability and router state
 - Multicast communication
 - Heterogeneity of destinations
 - Dynamic membership
 - Relation to routing and admission control

Case Study: RSVP

- A standard resource reservation protocol in the Internet is RSVP
- Basic operation:
 - Sources send periodic *path* messages, describing the flow
 - Create path state in intermediate routers
 - Receivers send *reservation* messages back towards the source
 - Cause intermediate routers to perform acceptance test and setup a resource reservation for the flow described by the path messages
 - May send a reject message to the receiver, if acceptance test fails
 - Reservations refreshed periodically by receivers
- Characteristics:
 - Soft state, for graceful failure
 - Receiver driven reservations support multicast
- Widely supported and available if you control the network, but not widely used in the public Internet

Summary

- Why enhanced service is needed
 - What is needed to support enhanced services
 - Queue discipline
 - Acceptance test
 - Signalling protocol
 - Two approaches to implementing priority queuing
 - WFQ
 - WRR
 - Brief pointer to RSVP
-
- No lectures next week - time to work on programming assignment
 - Resume with a tutorial on 15th March, on real-time networking