Packet Scheduling by Routers

N. C. State University

CSC557 ♦ Multimedia Computing and Networking

Fall 2001

Lecture # 19

"Roadmap" for Multimedia Networking

1. Introduction
   - why QoS?
   - what are the problems?

Lecture 17

2. Basic operations
   - jitter buffers (at hosts)
   - task scheduling (at hosts)
   - packet shaping (at hosts)
   - packet dropping (at routers)
   - packet scheduling (at routers)

Lecture 16

Lecture 18

Today's lecture

3. Types of service
   - Integrated Services (IntServ) and Resource Reservation Protocol (RSVP)
   - Differentiated Services (DiffServ)

"Roadmap" (cont'd)

4. Application-level feedback and control
   - Real-time Protocol (RTP), Real-time Control Protocol (RTCP)
   - Real-time Streaming Protocol (RTSP)
5. Application signaling and device control
   - Session Announcement Protocol (SAP)
   - Session Description Protocol (SDP)
   - Session Initiation Protocol (SIP)
   - Media Gateway Control Protocol (MGCP)
6. Routing
   - Multi-protocol Label Switching (MPLS)
   - multicasting

(Reminder of Problems, + Solutions)

Less-than-ideal average delays and loss rates	• Schedule packet transmission carefully to control delays/losses
2. Variations in traffic loads in the network	• …
✓ TCP's congestion control	• Improved packet dropping policies
✓ Retransmission-based error recovery	• Use with jitter buffers, and to restore anchor frames of video
5. Simplistic routing algorithms	• …
✓ "Burstiness" or variability of a single traffic source ✓ peak rate, average rate, maximum burst size	• "Shape" traffic to reduce variability

Definitions

- *Connections* or *Flows* or *Streams*
  - the sequence of packets associated with the execution of one application
- *Packet switching* requires sophisticated *schedulers*; circuit switching doesn't
  - circuit-switched: fixed-size packets, fixed arrival times
  - packet-switched: variable-sized packets, variable arrival times
- Goals of packet scheduling at routers
  - provide performance guarantees
  - *isolate* flows from one another
- Packet *waiting times*
  - time between arrival of packet to a queue and departure from the queue

The Work Conservation Law

- *First-in, first-out* (FIFO) scheduling
  - easy to implement
  - no isolation/protection between flows
- *Work-conservation*
  - link is never unused, as long as there are packets ready to transmit
- Theorem (without proof)
  - the sum of packet waiting times, weighted by their size, is constant for all work-conserving schedulers
- Corollary
  - if one connection's mean waiting time goes down, another connection's mean waiting time must go up

Example

- Six packets to transmit on a 100 bits/s link

Packet size	FIFO Departure Time	FIFO Weighted Waiting Time	LIFO Departure Time	LIFO Weighted Waiting Time
400	0	0	14	56
100	4	4	13	13
400	5	20	9	36
600	9	54	3	18
200	15	30	1	2
100	17	17	0	0
TOTAL		125		125

Work Conservation

- Work-conserving schedulers
  - simple to implement
  - efficient (never waste bandwidth)
  - however, they increase delay variation (jitter) in the network
- Non-work-conserving schedulers
  - are more complex
  - less efficient
  - but, limit the jitter within the network

Fairness

- Everyone gets a "fair share"
  - appropriate for best-effort traffic
- Fairness bad for multimedia??
  - don't treat everybody equally!
  - some applications require higher throughput than others
  - some users willing to pay more than others
- Max-Min Fair
  1. divide available bandwidth equally among competing flows, up to the amount needed by the flow (s) with the lowest demand
  2. As long as there is still available bandwidth, repeat step #1 for the remaining ("unsatisfied") flows

Fairness (cont'd)

- Example: six sources compete for bandwidth on a 1300 bits/s link

Source	Demands	"Round 1"	"Round 2"	"Round 3"	Total
1	100	100			100
2	200	100	100		200
3	200	100	100		200
4	300	100	100	67	267
5	400	100	100	67	267
6	500	100	100	67	267
Total	1700	600	500	200	1300

Fairness (cont'd)

- *Weighted* Max-Min Fair
  - every source is weighted (== relative importance, or priority, or resource amount required)
- Change step #1 to
  1. divide available bandwidth in proportion to weight among competing flows, up to the amount needed by the flow (s) with the lowest demand/weight ratio

Fairness (cont'd)

Source	Demands	Weight	Demand / Weight	"Round 1"	"Round 2"	Total
1	100	1	100	100		100
2	200	1	200	100	100	200
3	200	2	100	200		200
4	300	1	300	100	100	200
5	400	4	100	400		400
6	500	1	500	100	100	200
Total	1700			1000	300	1300

Delay Guarantees

- Possible choices
  - average delay of all packets $\leq t_1$
  - 100% of packets have delay $\leq t_2$
  - p% of packets have delay $\leq t_3$
- Which is appropriate for multimedia?

Admission Control

- *Admission control*
  - who is allowed to send traffic into the network when there aren't sufficient resources to satisfy all QoS requirements?
  - e.g., for telephone calls: "all circuits are busy right now…"
- Desirable qualities for admission control
  1. works for many different classes of traffic
  2. simple to implement
  3. minimum statistics needed from the user/application
  4. maximum network utilization
- No such method exists?!

Jitter Regulation (Non-Work-Conserving)

- *Packet Eligibility Time*: time when a packet may be considered for scheduling at router $s = e_p(s)$
  - prior to this time, packet must be held in a buffer
  - the purpose of this buffer is to reduce jitter in the network
- *Packet times*
  - t_p = time to transmit a packet at the peak rate allocated to the flow
  - $d_p(s)$ = worst case delay of packet p at router s
  - $a_p(s)$ = arrival time of packet p at router s

Jitter Regulation (Non-Work-Conserving)

- Rate jitter regulation
  - $e_p(s) = \max(e_{p-1}(s) + t_{p-1}, a_p(s))$
- Delay jitter regulation
  - $e_p(s) = e_p(s-1) + d_p(s-1)$
  - Eligible as soon as it arrives at first switch
  - requires close synchronization between switches
  - completely eliminates jitter within the network
- Examples
  - Packets arrive at *s* at times 0, 5, 7, 12, 13
  - RJ (t_{p-1} = 3): 0, 5, 8, 12, 15
  - DJ at switch s+1 ($d_p(s-1)$ = 5): 5, 10, 12, 17, 18

Priority Scheduling

- Static (strict) priority
  - Easily protects high-priority traffic from low-priority
  - How many levels are needed?
  - *Starvation* is possible
    - a lower priority traffic class may never get transmitted
- Dynamic priority
  - Compute a "service number" (dynamically) for each packet
  - Schedule in order of this service number
  - E.g., for earliest deadline first scheduling,
    service number = deadline

GPS

- An ideal scheduler in terms of fairness
- Max-min allocation of bandwidth
  - Divide available bandwidth equally among the connections, using bit-level packet interleaving
  - If any flow has an empty queue (no packets waiting to be scheduled), divide the excess evenly among the remaining flows
  - Continue as long as there are packets waiting to be scheduled
- A *round* in GPS = time to go through complete set of queues and transmit 1 bit from each non-empty queue

GPS (cont'd)

- Can also weight the allocation to connections
  - weighted GPS (achieves weighted max-min fairness)
- Problem: not practical to implement!
  - interleaving of packets is nightmare to demultiplex

Approximation #1 to GPS: Weighted Round Robin (WRR)

- Definitions
  - Let *P'(i)* = average packet size for flow *i*
  - a *round* in WRR = time to go through complete set of queues and transmit a packet of size *P'(i)* from each non-empty queue *I*
    - round lengths can vary!
- In one "round":
  - go through the set of connections one by one
  - schedule packet of size *P'(i)* to transmit from each non-empty queue *i*
  - have to account for variable packet sizes
- Weighting
  - each flow has a weight w_i
  - *normalized weight* in packets = w_i / *P'(i)*

WRR (cont'd)

- Example
  - Flows A, B, and C have mean packet sizes 50, 500, and 1500 bytes, and weights .5 : .75 : 1.0
  - Step 1: divide weights by packet sizes to get normalized weights: .01 : .0015 : .000666
  - Step 2: normalize to get integer weights: 60 : 9 : 4
  - In one round: 60 packets*50 = 3000 bytes for A, 9 * 500 = 4500 bytes for B, 4 * 1500 = 6000 bytes for C
  - 3000 : 4500 : 6000 = .5 : .75 : 1.0
- Close approximation to GPS if
  a) packet sizes are fixed, and
  b) link speeds are high, and/or packet sizes are small
- May be unfair to some flows over the duration of a single round

Approximation #2 to GPS: Weighted Fair Queueing (WFQ)

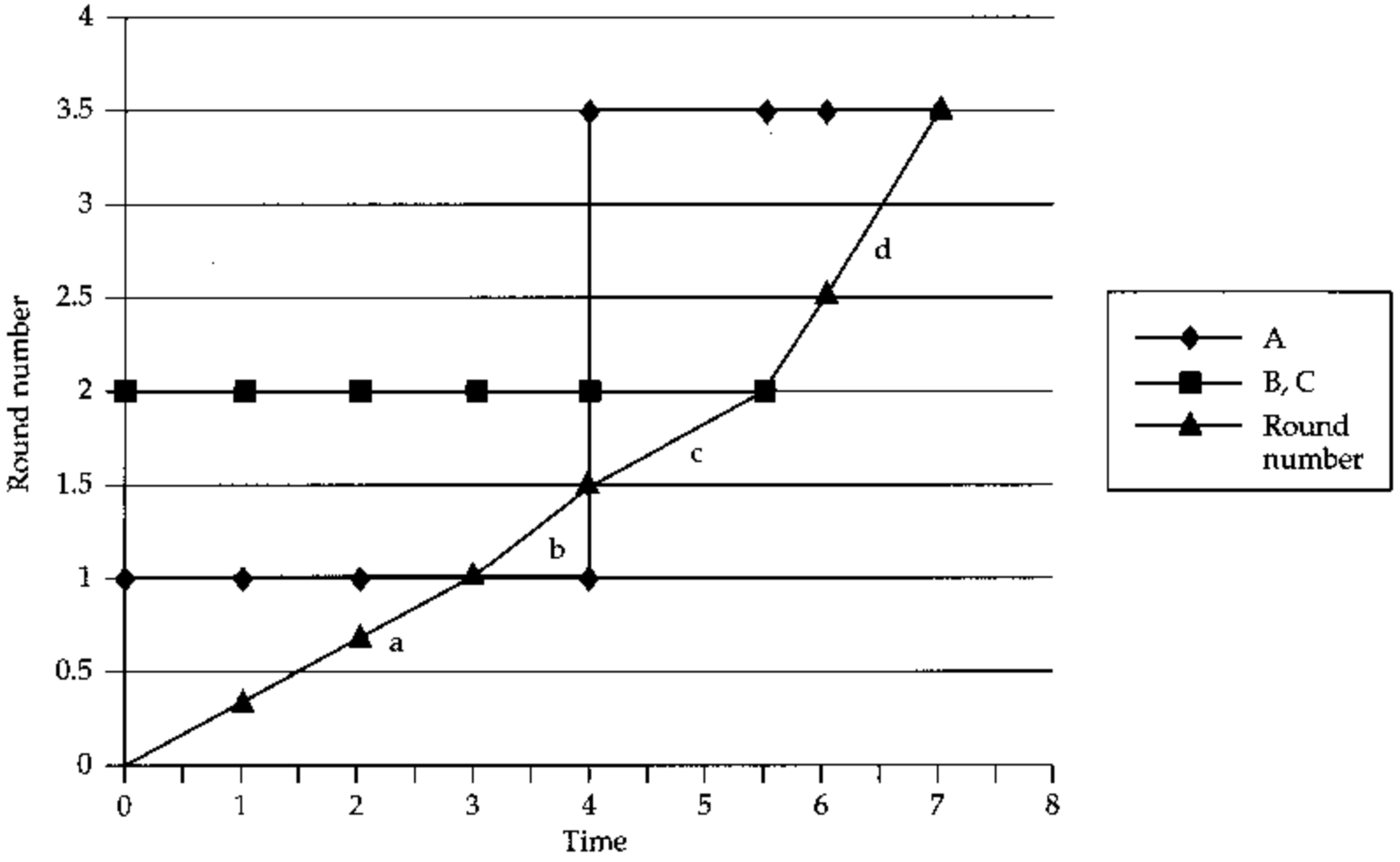
- Definitions
  - *rounds* = as in GPS
  - *expected finish number* for a packet = round in which last bit of packet would be transmitted, using GPS
  - *active connections* = flows for which the most-recently-arrived packet has a finish number greater than the current round
- Scheduling
  - for each packet at the head of a queue, compute its expected finish number (by simulating what GPS would do)
  - schedule the packets in order of increasing finish number

WFQ: Finish Number Computation For an Incoming Packet

- Finish number of inactive connection =
  current round number +
  length of incoming packet (in bits)
- Finish number of active connection =
  finish number of most-recently-arrived packet for this connection +
  length of incoming packet
- *The finish number for a connection* =
  finish number of most-recently-arrived packet for this connection
- *Finish time* = time at which packet actually leaves the queue
  - ≠ finish number

WFQ Example

- Assume
  - Packets of size 1, 2, and 2 arrive at t=0 for equally-weighted flows A, B, C
  - Packet of size 2 arrives on flow A at t=4



WFQ: Computing the "Current" Round

- Computing the current round
  - the rate of increase in the round # is updated on every packet arrival and departure
  - expensive!
- If every connection has weight w_i
  - in computation of finish number, divide packet length by the weight
  - increase the round number at a rate inversely proportional to sum of the weights of active connections, not by number of active connections

WFQ: Delay Bound

- Assumptions
  - each flow *i* is shaped by a token bucket with parameters $\rho(i)$ and $\sigma(i)$
  - path taken by *i* has *K* hops, or routers
  - every link has transmission rate *r*
  - set of flows scheduled at router *k* has aggregate weight *W(k)*
    - rate *g(i)* given to flow *i* is = minimum of (link rate *r* * w_i / *W(k))* over all routers
  - maximum packet size is P_{max}

WFQ: Delay Bound (cont'd)

1. Burst clearing time (at first router) = $\sigma(i)$ / $g(i)$
2. Packet transmission time = sum of (P_{max} / $g(i,k)$) over all K routers (except the first)
   - assumption: arrives just slightly too late to be scheduled in a round
3. Clearing time for packets of other flows = K^* (P_{max} / r)
4. End-to-end delay $\leq$
   burst clearing time
   + packet transmission time
   +clearing time for packets of other flows

Example

- Assumptions
  - $\rho(i) = 10^5$ bits/s
  - $\sigma(i) = 2*10^4$ bits
  - K = 10
  - $w_i = 2$, $W(k) = 30$ (at all routers)
  - $r = 10^6$ bits/s
  - $P_{max} = 5000$ bits
- Calculation
  - $g(i) = 10^6 * 2 / 30 = 6.7 * 10^4$ bits/s
  - Burst clearing time = $2*10^4 / 6.7 * 10^4 = .3$ s
  - Packet transmission time = $9 * 5000 / 6.7 * 10^4 = .67$ s
  - Clearing time = $10 *(5000 / 10^6) = .05$ s
  - Delay (end-to-end) $\leq .3 + .67 + .05 = 1.12$ s

Example (cont'd)

- Notes
  - independent of $\rho(i)$, for any source!
  - to decrease delay,
    - increase link rate
    - decrease burst size
    - decrease number of hops
    - increase weight relative to others

Evaluation of WFQ

- Good protection of flows
  - guaranteed throughput and delay bounds possible
- Overall rate allocated within any interval is very close to GPS
- Somewhat expensive to implement
  - per-connection monitoring, control, and state maintenance?

Sources of Info

- Recommended
  - Keshav, *An Engineering Approach to Computer Networks*, 1996 (chapter 9)
- Web sites