CS551
Router
Queue Management
Bill Cheng

http://merlot.usc.edu/cs551-f12
Congestion Control vs. Resource Allocation

Network’s key role is to allocate its transmission resources to users or applications.

Two sides of the same coin
- Let network do resource allocation (e.g., VCs)
  - difficult to do allocation of distributed resources
  - can be wasteful of resources
- Let sources send as much data as they want
  - recover from congestion when it occurs
  - easier to implement, may lose packets
Connectionless Flows

How can a connectionless network allocate anything to a user?
- It doesn’t know about users or applications

Flow:
- A sequence of packets between same source - destination pair, following the same route

Flow is visible to routers - it is not a channel, which is an end-to-end abstraction

Routers may maintain soft-state for a flow

Flow can be implicitly defined or explicitly established (similar to VC)
- Different from VC in that routing is not fixed
Goals

- **Fairness**
  - Fair Queueing [Demers89a]

- **Efficiency**
  - RED [Floyd93a]

- **Stability**
  - XCP [Katabi02a]

- **Service Differentiation**
  - RIO [Clark98a]
Design Dimensions

- How quickly do you provide feedback
- What kind of fairness do you provide
  - Fair to whom (flows, users, etc.)
  - How fair (probabilistic, guarantee, etc.)
  - Definition of fair (equal size, max-min)
- How efficient you are (router go idle?)
- How much state you must keep
  - constant amount, for some flows, for each flow
- How do you signal congestion
  - dropping packets vs. explicit feedback (DECbit, ECN)
Queueing Policies

Many policies have been considered

- FIFO ("drop tail")
  - also drop head
- Round robin (per flow)
- Weighted round robin
- Fair queueing
- Token bucket
- Virtual clock
- Class-based queueing (per class of traffic)
- Stochastic fair queueing (statistical)
Router-centric v.s. Host-centric

- **Router-centric:** address problem from inside network - routers decide what to forward and what to drop
  - variant: only at edge-routers
- **Host centric:** address problem at the edges - hosts observe network conditions and adjust behavior
- Not always a clear separation: hosts and routers may collaborate, e.g., routers advise hosts
Taxonomy (Cont...)

Reservation-based v.s. Feedback-based

- **Reservations**: hosts ask for resources, network responds yes/no
  - implies router-centric allocation
- **Feedback**: hosts send with no reservation, adjust according to feedback
  - either router or host centric: explicit (e.g., ICMP source quench) or implicit (e.g., loss) feedback
Taxonomy (Cont...) 

- Window-based v.s. Rate-based
  - Both tell sender how much data to transmit
  - **Window**: TCP flow/congestion control
    - Flow control: advertised window
    - Congestion control: cwnd
  - **Rate**: still an open area of research
    - May be logical choice for reservation-based system
Service Models

- In practice, fewer than eight choices
- Best-effort networks
  - Mostly host-centric, feedback, window based
  - TCP as an example
- Networks with flexible Quality of Service
  - Router-centric, reservation, rate-based
Queueing Disciplines

Each router *must* implement some queuing discipline regardless of what the resource allocation mechanism is.

Queueing discipline allocates:
- *bandwidth*: which packets get transmitted
- *buffer space*: which packets get dropped
- *promptness*: when packets get transmitted
FIFO Queuing

- FIFO: first-in-first-out (or FCFS: first-come-first-served)
- Arriving packets get dropped when queue is full regardless of flow or importance - implies *droptail*
- Important distinction:
  - *FIFO*: scheduling discipline (which packet to serve next)
  - *Drop-tail*: drop policy (which packet to drop next)
Dimensions

Per-connection state

Scheduling

Single class

Class-based queuing

Head

Drop position

Tail

FIFO

Early drop

Random location

Overflow drop
FIFO

- FIFO + drop-tail is the simplest queuing algorithm
  - Used widely in the Internet

- Leaves responsibility of congestion control to edges (e.g., TCP)

- FIFO lets large user get more data through but shares congestion with others
  - Does not provide *isolation* between different flows
  - No policing
Fair Queueing
[Demers89a]

Bill Cheng

http://merlot.usc.edu/cs551-f12
Fair Queuing

Fair Queueing (FQ) [Nagle85, Nagle87]

Main idea:
- Maintain a separate queue for each flow currently flowing through router
- Router services queues in *Round-Robin* fashion

Changes interaction between packets from different flows
- Provides isolation between flows
- Ill-behaved flows cannot starve well-behaved flows
- Allocates buffer space and bandwidth fairly
Variation: Weighted Fair Queueing (WFQ)
Some Issues

What constitutes a user?
- Several granularities at which one can express flows
- For now, assume at the granularity of source-destination pair, but this assumption is not critical

Packets are of different length
- Source sending longer packets can still grab more than their share of resources
- We really need *bit-by-bit round-robin*
- Fair Queuing *simulates* bit-by-bit round-robin
  - not feasible to interleave bits!
Bit-by-bit Round-robin

- **Router maintains a logical clock**

- **Single flow:** suppose clock ticks when a bit is transmitted.
  For packet $i$:
  - $P_i$: length, $A_i = \text{arrival time}, S_i = \text{begin transmit (start time)}$
  - $F_i$: finish time
  - $S_i = \max(F_{i-1}, A_i)$
  - $F_i = S_i + P_i$
  - $F_i = \max(F_{i-1}, A_i) + P_i$

- **Multiple flows:** logical clock ticks when a bit from all active flows is transmitted
  - Logical clock = number of *rounds* served
  - Logical clock advances more slowly when there are more flows

*Computer Communications - CSCI 551*
Fair Queuing

While we cannot actually perform bit-by-bit interleaving, can *compute* (for each packet) $F_i$. Then, use $F_i$ to schedule packets

- Transmit earliest $F_i$ first

Still not completely fair

- But difference now bounded by the size of the largest packet
- Compare with previous approach
Cannot preempt packet currently being transmitted
Max-min Fairness

Max-min Fairness: a fair service maximizes the service of the customer receiving the poorest service

Max-min Fairness criterion:
1) no user receives more than its request
2) no other allocation scheme satisfying condition 1 has a high minimum allocation
3) condition 2 remains recursively true as we remove the minimal user and reduce total resource accordingly
Max-min Fairness Example

Total capacity $C$ divided among $N$ flows:
- $x_i$ is the request of flow $i$
- sort flows based on $x_i$
- initially, assign $C/N$ to each flow

1. Sort flows based on $x_i$.
2. Initially, assign $C/N$ to each flow.
3. Satisfy $x_1$, redistribute the remaining capacity evenly.
4. Recursion.

Mathematically:
- $C/N$ for each flow.
- $(C-x_1)/(N-1)$ for the remaining capacity.

Diagram:
- Bars representing flows $x_1, x_2, x_3, \ldots, x_N$.
- Initial allocation $C/N$.
- Redistribution of remaining capacity $(C-x_1)/(N-1)$.

Fair Queuing Example

All packets are of size 1 ($P_i=1$, $\forall i$), real arrive times are in *real time*

- e.g., three queues, $X$, $Y$, and $Z$, pack $X1$ arrive at queue $X$ at real time 1, pack $X3$ arrive at queue $X$ at real time 3, etc.
  - $X1$, $X3$
  - $Y2$, $Y5$
  - $Z1$, $Z4$

$$F_i = \max(F_{i-1}, A_i) + P_i$$

- $F_i = \max(F_{i-1}, A_i) + 1$
- what are the logical arrival times for the 6 packets?
  - arrival times, finish times are all *logical times*
  - how do you map *real time* to *logical time*?
  - $A_{X1} = 1$
  - $A_{Z1} = 1$
Fair Queuing Example (Cont...)

- Packets:
  - X1, X3
  - Y2, Y5
  - Z1, Z4

- Arrival/finish times:
  - \( A_{X1} = 1 \)
  - \( A_{Z1} = 1 \)
  - \( F_{X1} = F_{Z1} = 2 \)
  - 2 flows,
    slope = 1/2
  - what’s next?
How to decide?

Think event driven simulation...
- \( A_{X_1} = 1 \)
- \( A_{Z_1} = 1 \)
- \( F_{X_1} = F_{Z_1} = 2 \)
- next arrival is \( Y_2 \) (arrives at real time 2)
- arrival wins here!
How To Calculate Next Event

Current coordinate is \((x_0,y_0)\) and slope is \(r\)

- need to find the next event on the X-axis and the Y-axis
  - next event on the X-axis is the next pack arrival
  - next event on the Y-axis is the next packet departure

1) If next event will be an arrival event at \textit{real} time \(x_1\)
   - next event will occur at \((x_1,y_1)\) where \((y_1-y_0)/(x_1-x_0)=r\)
   - solve for \(y_1\), the \textit{logical} arrival time of this arriving packet
     - from logical arrival time, you can easily calculate the \textit{logical} finish time using the bit-by-bit RR equation

2) If next event will be a departure event at \textit{logical} time \(y_1\)
   - next event will occur at \((x_1,y_1)\) where \((y_1-y_0)/(x_1-x_0)=r\)
   - solve for \(x_1\), to make sure that there is no arrival between \textit{real} time \(x_0\) and \(x_1\)
     - verify that \(y_1\) is the \textit{logical} finish time of the departing packet
Packets:
- X1, X3
- Y2, Y5
- Z1, Z4

Arrival/finish times:
- $A_{X1} = 1$
- $A_{Z1} = 1$
- $F_{X1} = F_{Z1} = 2$
- 2 flows, slope = 1/2
- $A_{Y2} = 1.5$
- $F_{Y2} = 2.5$
- $A_{X3} = 1.833$
- $F_{X3} = 3$
Fair Queuing Example (Cont...)

Arrival/finish times:
- \( A_{X1} = 1 \)
- \( F_{X1} = 2 \)
- \( A_{Z1} = 1 \)
- \( F_{Z1} = 2 \)
- \( A_{Y2} = 1.5 \)
- \( F_{Y2} = 2.5 \)
- \( A_{X3} = 1.833 \)
- \( F_{X3} = 3 \)
- \( A_{Z4} = 2.25 \)
- \( F_{Z4} = 3.25 \)
- \( A_{Y5} = 2.625 \)
- \( F_{Y5} = 3.625 \)
Fair Queuing Example (Cont...)

Arrival/finish times:
- \( A_X1 = 1 \)
- \( F_X1 = 2 \)
- \( A_Z1 = 1 \)
- \( F_Z1 = 2 \)
- \( A_Y2 = 1.5 \)
- \( F_Y2 = 2.5 \)
- \( A_X3 = 1.833 \)
- \( F_X3 = 3 \)
- \( A_Z4 = 2.25 \)
- \( F_Z4 = 3.25 \)
- \( A_Y5 = 2.625 \)
- \( F_Y5 = 3.625 \)

Output:
- \( F_X1 = 2 \)
- \( F_Z1 = 2 \)
- \( F_Y2 = 2.5 \)
- \( F_X3 = 3 \)
- \( F_Z4 = 3.25 \)
- \( F_Y5 = 3.625 \)

---

Arrival/finish times:
- \( A_X1 = 1 \)
- \( F_X1 = 2 \)
- \( A_Z1 = 1 \)
- \( F_Z1 = 2 \)
- \( A_Y2 = 1.5 \)
- \( F_Y2 = 2.5 \)
- \( A_X3 = 1.833 \)
- \( F_X3 = 3 \)
- \( A_Z4 = 2.25 \)
- \( F_Z4 = 3.25 \)
- \( A_Y5 = 2.625 \)
- \( F_Y5 = 3.625 \)

Output:
- \( F_X1 = 2 \)
- \( F_Z1 = 2 \)
- \( F_Y2 = 2.5 \)
- \( F_X3 = 3 \)
- \( F_Z4 = 3.25 \)
- \( F_Y5 = 3.625 \)

---

Arrival/finish times:
- \( A_X1 = 1 \)
- \( F_X1 = 2 \)
- \( A_Z1 = 1 \)
- \( F_Z1 = 2 \)
- \( A_Y2 = 1.5 \)
- \( F_Y2 = 2.5 \)
- \( A_X3 = 1.833 \)
- \( F_X3 = 3 \)
- \( A_Z4 = 2.25 \)
- \( F_Z4 = 3.25 \)
- \( A_Y5 = 2.625 \)
- \( F_Y5 = 3.625 \)

Output:
- \( F_X1 = 2 \)
- \( F_Z1 = 2 \)
- \( F_Y2 = 2.5 \)
- \( F_X3 = 3 \)
- \( F_Z4 = 3.25 \)
- \( F_Y5 = 3.625 \)
Delay Allocation

Aim: give less delay to those using less than their fair share

Advance finish times for sources whose queues drain temporarily

\[ B_i = P_i + \max(F_{i-1}, A_i - \delta) \]

Schedule earliest \( B_i \) first

\( \delta \) gives added promptness:
- If \( A_i < F_{i-1} \), conversation is active and \( \delta \) does not affect it: \( F_i = P_i + F_{i-1} \)
- If \( A_i > F_{i-1} \), conversation is inactive and \( \delta \) determines how much history to take into account
Notes on FQ

- FQ is a scheduling policy, not a drop policy
- Still achieves statistical multiplexing - one flow can fill entire pipe if no contenders - FQ is *work conserving*
- WFQ is a possible variation - need to learn about weights offline. Default is one bit per flow, but sending more bits is possible
Weighted Fair Queuing Example

- Weights for X, Y, and Z are 1, 2, and 1, respectively
  - X1, X3
  - Y2, Y5
  - Z1, Z4

- Shrink packet size of Y2 and Y5 by half

- Need to count Y twice when queue Y is not empty

- \( F_i = \max(F_{i-1}, A_i) + P_i \)
  - \( A_{X1} = 1 \)
  - \( A_{Z1} = 1 \)
  - ... (proceed as before)
More Notes on FQ

- Router does not send explicit feedback to source - still needs e2e congestion control
  - FQ isolates ill-behaved users by forcing users to share overload with themselves
  - User: flow, transport protocol, etc

- Optimal behavior at source is to keep one packet in the queue

- But, maintaining *per flow state* can be expensive
  - Flow aggregation is a possibility