CS557: Queue Management

Christos Papadopoulos

Remixed by Lorenzo De Carli
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., virtual circuits)
    - Difficult to do allocation of distributed resources
    - Can be wasteful of resources
  - Let sources send as much data as they want
    - Then recover from congestion when it occurs
    - Easier to implement, but 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
Taxonomy

- Router-centric v.s. Host-centric
  - **router-centric**: address the problem from inside network - routers decide what to forward and what to drop
    - a variant not captured in the taxonomy: adaptive routing!
  - **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

- Reservation-based v.s. Feedback-based
  - **Reservations**: hosts ask for resources, network responds yes/no
    - implies router-centric allocation
    - E.g., RSVP protocol
  - **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

- 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

• Best-effort networks
  – Mostly host-centric, feedback, window based
  – TCP is *the* example

• Networks with flexible Quality of Service
  – Router-centric, reservation, rate-based
Queuing Disciplines

• Each router MUST implement some queuing discipline regardless of what the resource allocation mechanism is

• Queuing allocates bandwidth, buffer space, and promptness:
  – **bandwidth**: which packets get transmitted
  – **buffer space**: which packets get queued/dropped
  – **promptness**: when packets get transmitted
FIFO Queuing

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

Scheduling

Per-connection state

Class-based queuing

Drop position

Head

Random location

When to drop

Single class

FIFO

Early drop

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
But what does “fairness” mean?

• In this discussion we are going to use **max-min fairness**:
  • Suppose we have an amount $\mu$ of a resource $R$ to be shared between $N$ users
  • We define the fair share of $r$ as $\mu_f = \mu / N$
  • Every user requests a certain amount of $R$; $\rho_i$ is the amount of resource requested by the $i$-th user
  • Max-min fairness is achieved if:
    • All users requesting an amount $\rho_i \leq \mu_f$ receive the desired amount
    • The remaining amount of resource is distributed evenly among all other users requesting $\rho_i > \mu_f$
Max-min fairness example

• Suppose I have a link bandwidth of 10Kb/s to distribute across 5 users
• Fair share $\mu_f = \frac{10 \text{ Kb/s}}{5} = 2 \text{ Kb/s}$
Max-min fairness example

- Suppose I have a link bandwidth of 10 Kb/s to distribute across 5 users
- Fair share $\mu_f = \frac{10 \text{ Kb/s}}{5} = 2 \text{ Kb/s}$

**Requested:**
A: 1 Kb/s,
B: 2 Kb/s
C: 20 Kb/s
D: 1 Kb/s
E: 4 Kb/s
Max-min fairness example

• Suppose I have a link bandwidth of 10Kb/s to distribute across 5 users
• Fair share $\mu_f = \frac{10 \text{ Kb/s}}{5} = 2 \text{ Kb/s}$

<table>
<thead>
<tr>
<th>Requested:</th>
<th>Assigned:</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: 1 Kb/s,</td>
<td>A: 1 Kb/s,</td>
</tr>
<tr>
<td>B: 2 Kb/s</td>
<td>B: 2 Kb/s</td>
</tr>
<tr>
<td>C: 20 Kb/s</td>
<td>C: 3 Kb/s</td>
</tr>
<tr>
<td>D: 1 Kb/s</td>
<td>D: 1 Kb/s</td>
</tr>
<tr>
<td>E: 4 Kb/s</td>
<td>E: 3 Kb/s</td>
</tr>
</tbody>
</table>
Fair Queuing

• Demmers90
Fair Queuing

• Main idea:
  – maintain a separate queue for each flow 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
FQ Illustration

Flow 1
Flow 2
Flow n

I/P

O/P
Variation: Weighted Fair Queuing (WFQ)
Issues

• What constitutes a user?
  – Several granularities at which one can express flows
    – Source (= source IP address): restricts sources which consume high bandwidth (e.g. file storage service)
    – Receiver: (= destination IP address) expose nodes to denial of service (just send a node unwanted traffic to fill its share)
    – Process: (= source IP + TCP source port): easily sidestepped by opening multiple copies of same process
    – Source+destination (= source IP + destination IP): can consume boundless BW by sending traffic to many destinations
    – Conversation (=source IP+TCP source port+ destination IP+TCP destination port)
  – The paper assumes conversation as the scheduling granularity, but the choice is not critical for the functioning of the algorithm
Issues - II

- 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
    - But not feasible to interleave bits!
  - Fair Queuing *simulates* bit-by-bit RR
Bit-by-bit RR

- Router maintains local clock
- Single flow: suppose clock ticks when a bit is transmitted. For packet $i$:
  - $P_i$: length, $A_i =$ arrival time, $S_i$: begin transmit time, $F_i$: finish transmit time. $F_i = S_i + P_i$
  - $F_i = \max (F_{i-1}, A_i) + P_i$
- Multiple flows: clock ticks when a bit from all active flows is transmitted
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
Fair Queuing Example
Fair Queuing Example

Flow 1
F=8
F=5

Flow 2
F=10
Fair Queuing Example

Flow 1
- F=8
- F=5

Flow 2
- F=10

Output
- F=5
Fair Queuing Example

Cannot preempt packet currently being transmitted

Flow 1 (arriving)  Flow 2 (transmitting)
Fair Queuing Example

Flow 1
F=8
F=5

Flow 2
F=10

Output

Flow 1 (arriving)
F=2
Flow 2 (transmitting)
F=10
Output

Cannot preempt packet currently being transmitted

21
Delay Allocation

• Aim: give less delay to those using less than their fair share
• Adjust finish times for sources whose queues drain temporarily
• $B_i = P_i + \max (F_{i-1}, A_i - \delta)$
• Schedule earliest $B_i$ first
Allocate Promptness

- \( B_i = P_i + \max (F_{i-1}, A_i - \delta) \)
- \( \delta \) 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
  - This strategy makes the scheduling decision dependent on the previous packet finishing round, as long as it was not too long ago
  - How long ago? That depends on the choice of \( \delta \)
Notes on FQ

• FQ is a scheduling policy, not a drop policy
• Still achieves statistical muxing - one flow can fill entire pipe if no contenders – FQ is work conserving (transmission never idle if there are packets in any queue)
• WFQ is a possible variation – need to learn about weights off line. Default is one bit per flow, but sending more bits is possible
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
Congestion Avoidance

• TCP’s approach is reactive:
  – detect congestion after it happens
  – increase load trying to maximize utilization until loss occurs
  – TCP has a congestion avoidance phase, but that’s different from what we’re talking about here

• Alternatively, we can be proactive:
  – we can try to predict congestion and reduce rate before loss occurs
  – this is called congestion avoidance
Router Congestion Notification

- Routers well-positioned to detect congestion
  - Router has unified view of queuing behavior
  - Routers can distinguish between propagation and persistent queuing delays
  - Routers can decide on transient congestion, based on workload

- Hosts themselves are limited in their ability to infer these from perceived behavior
Router Mechanisms

• Congestion notification
  – the DEC-bit scheme
    • explicit congestion feedback to the source
    • Very simple mechanism
  – Random Early Detection (RED)
    • implicit congestion feedback to the source
    • well suited for TCP
  – Explicit Control Protocol (XCP)
    • Provide better congestion signal (not just binary)