CS 268: Lecture 7
Router Support for Congestion Control

Scott Shenker and Ion Stoica
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720-1776

1

Router Support For Congestion Management

- Traditional Internet
  - Congestion control mechanisms at end-systems, mainly implemented in TCP
  - Routers play little role
- Router mechanisms affecting congestion management
  - Scheduling
  - Buffer management
- Traditional routers
  - FIFO
  - Tail drop

2

Drawbacks of FIFO with Tail-drop

- Buffer lock out by misbehaving flows
- Synchronizing effect for multiple TCP flows
- Burst or multiple consecutive packet drops
  - Bad for TCP fast recovery

3

FIFO Router with Two TCP Sessions

4

RED

- FIFO scheduling
- Buffer management:
  - Probabilistically discard packets
  - Probability is computed as a function of average queue length (why average?)

5

RED (cont’d)

- min_th – minimum threshold
- max_th – maximum threshold
- avg_len – average queue length
  - $\text{avg_len} = (1-w)\text{avg_len} + w\text{sample_len}$

6
RED (cont’d)

- If (avg_len < min_th) → enqueue packet
- If (avg_len > max_th) → drop packet
- If (avg_len >= min_th and avg_len < max_th) → enqueue packet with probability $P$

$P = \max_P(\text{avg_len} - \text{min_th})/ (\text{max_th} - \text{min_th})$

Improvements to spread the drops
$P' = P/(1 - \text{count*}P)$, where
- count – how many packets were consecutively enqueued since last drop

RED Advantages

- Absorb burst better
- Avoids synchronization
- Signal end systems earlier

RED Router with Two TCP Sessions

Problems with RED

- No protection: if a flow misbehaves it will hurt the other flows
- Example: 1 UDP (10 Mbps) and 31 TCP’s sharing a 10 Mbps link

Solution?

- Round-robin among different flows [Nagle ’87]
  - One queue per flow
Round-Robin Discussion

- Advantages: protection among flows
  - Misbehaving flows will not affect the performance of well-behaving flows
  - FIFO does not have such a property
- Disadvantages:
  - More complex than FIFO: per flow queue/state
  - Biased toward large packets – a flow receives service proportional to the number of packets (When is this bad?)

Solution?

- Bit-by-bit round robin
- Can you do this in practice?
- No, packets cannot be preempted (why?)
- ...we can only approximate it

Fair Queueing (FQ) [DKS’89]

- Define a fluid flow system: a system in which flows are served bit-by-bit
- Then serve packets in the increasing order of their deadlines
- Advantages
  - Each flow will receive exactly its fair rate
- Note:
  - FQ achieves max-min fairness

Max-Min Fairness

- Denote
  - $C$ – link capacity
  - $N$ – number of flows
  - $r_i$ – arrival rate
- Max-min fair rate computation:
  1. compute $C/N$
  2. if there are flows $i$ such that $r_i < C/N$, update $C$ and $N$
     \[ C = C - \sum_{i,j \neq i} r_i \]
  3. if no, $f = C/N$; terminate
  4. go to 1
- A flow can receive at most the fair rate, i.e., $\min(f, r_i)$

Example

- $C = 10$; $r_1 = 8$, $r_2 = 6$, $r_3 = 2$; $N = 3$
- $C/3 = 3.33 \Rightarrow C = C - r_3 = 8$; $N = 2$
- $C/2 = 4$; $f = 4$

Alternate Way to Compute Fair Rate

- If link congested, compute $f$ such that
  \[ \sum_{i} \min(r_i, f) = C \]
Implementing Fair Queueing

- Idea: serve packets in the order in which they would have finished transmission in the fluid flow system

System Virtual Time: \( V(t) \)

- Measure service, instead of time
- \( V(t) \) slope – rate at which every active flow receives service
  - \( C \) – link capacity
  - \( N(t) \) – number of active flows in fluid flow system at time \( t \)

\[
\frac{dV(t)}{dt} = \frac{C}{N(t)}
\]

“Weighted Fair Queueing” (WFQ)

- What if we don’t want exact fairness?
  - E.g.: file servers
- Assign weight \( w_i \) to each flow \( i \)
- And change virtual finishing time

\[
F_{i}^{k+1} = \max( V(a_i^k), F_i^k ) + \frac{L_i^{k+1}}{w_i}
\]

Example

- Flow 1 (arrival traffic)
  - Time
- Flow 2 (arrival traffic)
  - Time
- Service in fluid flow system
  - Time
- Packet system
  - Time

Fair Queueing Implementation

- Define
  - \( F_i^k \) finishing time of packet \( k \) of flow \( i \) (in system virtual time reference system)
  - \( a_i^k \) – arrival time of packet \( k \) of flow \( i \)
  - \( L_i^k \) – length of packet \( k \) of flow \( i \)
- The finishing time of packet \( k+1 \) of flow \( i \) is

\[
F_i^{k+1} = \max( V(a_i^k), F_i^k ) + L_i^{k+1}
\]

Simulation Example

- 1 UDP (10 Mbps) and 31 TCPs sharing a 10 Mbps link

Stateless solution: Random Early Detection (RED)

Stateful solution: Fair Queuing
Core-Stateless Fair Queueing (CSFQ)

- Fair Queueing requires per flow state in routers
  - Maybe impractical for very high speed routers
- Core Stateless Fair Queueing eliminates the state at core routers ...
- … but only approximates FQ’s behavior

Insight

- If each packet of a flow with arrival rate \( r \) is forwarded with probability
  \[ P = \min \left( 1, \frac{L}{r} \right) \]
- the rate of flow’s forwarded traffic \( r' \) is
  \[ r' = r \times P = r \times \min \left( 1, \frac{L}{r} \right) = \min(r, f) \]
- No need to maintain per-flow state if \( r \) is carried in the packet
  - Need to update rate in packet to \( r' \)

CSFQ

- A contiguous and trusted region of network in which
  - Edge nodes – perform per-flow operations
  - Core nodes – do not perform any per-flow operations

Algorithm Outline

- Ingress nodes: estimate rate \( r \) for each flow and insert it in the packets’ headers

Algorithm Outline

- Ingress nodes: estimate rate \( r \) for each flow and insert it in the packets’ headers

- Core node:
  - Compute fair rate \( f \) on the output link
  - Enqueue packet with probability
    \[ P = \min(1, f/r) \]
  - Update packet label to \( r = \min(r, f) \)
Algorithm Outline
- Egress node: remove state from packet's header

Fair Rate Estimation
- Observation: rate of aggregate forwarded rate (R) is a monotonic and non-decreasing function of estimated rate
  \[ R(\bar{T}) = \min(8, \bar{T}) + \min(6, \bar{T}) + \min(2, \bar{T}) \]

Summary
- FQ does not eliminate congestion → it just manages the congestion
- You need both end-host congestion control and router support for congestion control
  - End-host congestion control to adapt
  - Router congestion control to protect/isolate
- Don’t forget buffer management: you still need to drop in case of congestion. Which packet’s would you drop in FQ?
  - One possibility: packet from the longest queue

Example: CSFQ
- Assume estimated fair rate \( f = 4 \)
  - flow 1, \( r = 8 \) ⇒ \( P = \min(1, 4/8) = 0.5 \)
  - expected rate of forwarded traffic \( 8/4P = 4 \)
  - flow 2, \( r = 6 \) ⇒ \( P = \min(1, 4/6) = 0.67 \)
  - expected rate of forwarded traffic \( 6/4P = 4 \)
  - flow 3, \( r = 2 \) ⇒ \( P = \min(1, 4/2) = 1 \)
  - expected rate of forwarded traffic 2

Simulation Example
- Stateless solution: Random Early Detection
- Stateful solution: Fair Queueing
- Our Solution: Core-Stateless Fair Queueing

Flow Number

Throughput

Flow Number