Router Support For Congestion Control

- 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

Router Packet Processing

- Scheduling:
  - Decide when to send a packet
  - Decide which packet to transmit
- Buffer management:
  - Decide when to drop a packet
  - Decide which packet to drop

FIFO: First-In First-Out

- Maintain a queue to store all packets
- When to send?
  - When there is at least one packet in the queue
- Which packet to send?
  - The packet at the head of the queue

Tail-drop Buffer Management

- When to drop?
  - When buffer full
- Which packet to drop
  - The arriving packet

Drawbacks of FIFO with Tail-drop

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

- When buffer overflows many/all flows experience losses
- Flows which experience losses will scale down their transmission (by reducing the window size)
- This will cause the link to be underutilized
- … the process repeats
- Net result: inefficient link utilization

Buffer Lock Out by Misbehaving Flows

- “Aggressive” flows can squeeze out TCP flows
- Example: one 10 Mbps link shared by
  - 1 UDP flow that sends at 10 Mbps, and does not implement any congestion control
  - 31 TCP flows

Burst of Packet Losses

- A TCP flow sends a burst of packets that arrive at a router when buffer is full
- All packets in the burst are lost
- This may cause RTO to trigger: cwnd → 1
- For example, with TCP Reno if there are three packet losses in a window, this will almost sure trigger an RTO

Random Early Detection (RED) Routers (Floyd & Fall 93)

- Goals
  - Avoid synchronizing multiple TCP flows
  - Avoid burst of packet drops

RED

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

- min_th – minimum threshold
- max_th – maximum threshold
- avg_len – average queue length
  - \( \text{avg}_\text{len} = (1-w) \times \text{avg}_\text{len} + w \times \text{sample}_\text{len} \)

**Discard Probability**

- If \( \text{avg}_\text{len} < \text{min}_\text{th} \) → enqueue packet
- If \( \text{avg}_\text{len} > \text{max}_\text{th} \) → drop packet
- If \( \text{avg}_\text{len} \geq \text{min}_\text{th} \) and \( \text{avg}_\text{len} < \text{max}_\text{th} \) → enqueue packet with probability \( P \)

**RED Advantages**

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

**RED Router with Two TCP Sessions**

- High network utilization with low delays
- Average queue length small, but capable of absorbing large bursts
  - Many refinements to basic algorithm make it more adaptive (requires less tuning)
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

Explicit Congestion Notification

- Goal: avoid burst of packet losses
- Rather than drop packets to signal congestion, router can send an explicit signal
- Explicit congestion notification (ECN):
  - instead of optionally dropping packet, router sets a bit in the packet header
  - If data packet has bit set, then ACK has ECN bit set
- Backward compatibility:
  - bit in header indicates if host implements ECN
  - note that not all routers need to implement ECN

Picture

ECN Advantages

- No need for retransmitting optionally dropped packets
- No confusion between congestion losses and corruption losses

Remaining Problem: Misbehaving Flows

- Internet vulnerable to CC cheaters!
- Single CC standard can’t satisfy all applications
- Goal:
  - Make Internet invulnerable to cheaters
  - Allow end users to use whatever congestion control they want
- How?

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
    - Misbehaving flow – a flow that does not implement any congestion control
  - 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 \( r_i \leq C/N \), update \( C \) and \( N \)
    \[
    C = C - \sum_{i \in S} r_i \cap C/N
    \]
  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

- \( C = 10; r_1 = 8, r_2 = 6, r_3 = 2; N = 3 \)
- \( C/3 \rightarrow C = C - r_3 = 8; N = 2 \)
- \( C/2 = 4 \rightarrow f = 4 \)

Implementing Fair Queueing

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

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 $t$

Fair Queueing Implementation

- Define
  - $F^k$: finishing time of packet $i$ of flow $i$ (in system virtual time reference system)
  - $a_i^k$: arrival time of packet $i$ of flow $i$
  - $L_i^k$: length of packet $i$ of flow $i$

- The finishing time of packet $k+1$ of flow $i$ is
  \[ F_{i,k+1} = \max(V(a_i^{k+1}), F_i^k) + L_i^{k+1} \]

FQ Advantages

- FQ protect well-behaved flows from ill-behaved flows
- Example: 1 UDP (10 Mbps) and 31 TCP’s sharing a 10 Mbps link

Big Picture

- 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