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

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
  - avg_len = \((1-w) \cdot \text{avg\_len} + w \cdot \text{sample\_len})\)
### RED (cont’d)

- 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 \( P \)

\[
P = \frac{\text{max}_\text{P} \cdot (\text{avg}_\text{len} - \text{min}_\text{th})}{(\text{max}_\text{th} - \text{min}_\text{th})}
\]

Improvements to spread the drops:

\[P' = \frac{P}{1 - \text{count} \cdot P}, \text{ where}
\]

- \( \text{count} \) – how many packets were consecutively enqueued since last drop

### RED Advantages

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

### 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
     \[ C = C - \sum_{i \in f} r_i \]
  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.

Example

Flow 1 (arrival traffic)

Flow 2 (arrival traffic)

Service in fluid flow system

Packet 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 \( t \).

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

Fair Queueing Implementation

- Define
  - \( F_{i+1}^k \) finishing time of packet \( k \) of flow \( i \) (in system virtual time reference system).
  - \( a_{i+1}^k \) – 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

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

“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

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

Simulation Example

- 1 UDP (10 Mbps) and 31 TCPs sharing a 10 Mbps link.
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
  $$P = \min\left(1, \frac{r}{T}\right)$$
- the rate of flow’s forwarded traffic $r'$ is
  $$r' = r \times P = r \times \min\left(1, \frac{r}{T}\right) = \min(r, f)$$
- 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

- 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
  
  \[
  R(T) = \min(8,T) + \min(6,T) + \min(2,T)
  \]

Summary

- FQ does not eliminate congestion \( \rightarrow \) 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/iseolate
- 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, \( r = 8 \gg P = \min(1, 4/8) = 0.5 \)
  - expected rate of forwarded traffic \( 8 \cdot P = 4 \)
  - flow 2, \( r = 6 \gg P = \min(1, 4/6) = 0.67 \)
  - expected rate of forwarded traffic \( 6 \cdot P = 4 \)
  - flow 3, \( r = 2 \gg P = \min(1, 4/2) = 1 \)
  - expected rate of forwarded traffic 2

Simulation Example

Stateless solution: Random Early Detection

Our Solution: Core-Stateless Fair Queueing

Announcements

- Project feedback
  - Tuesday, Feb 14, 12:30-2pm
  - Wednesday, Feb 15, 11:30-1pm