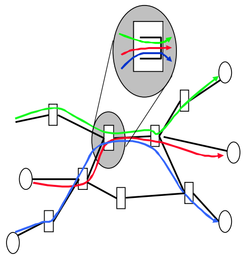
CS 268: Lecture 8 Router Support for Congestion Control

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

Router Support For Congestion Management

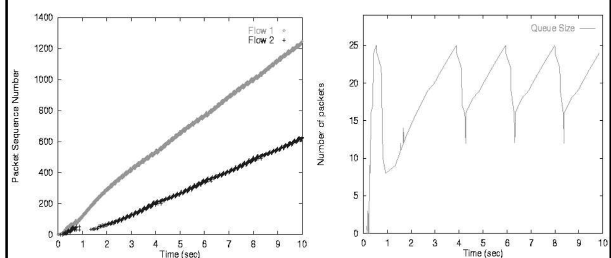
- Traditional Internet
  - Congestion control mechanisms at end-systems, mainly implemented in TCP
  - Routers play little role
- Router mechanisms affecting congestion management
  - Scheduling
  - Buffer management
- Traditional routers
  - FIFO
  - Tail drop



Drawbacks of FIFO with Tail-drop

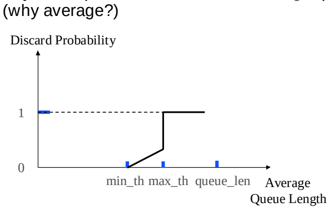
- Buffer lock out by misbehaving flows
- Synchronizing effect for multiple TCP flows
- Burst or multiple consecutive packet drops
  - Bad for TCP fast recovery

FIFO Router with Two TCP Sessions



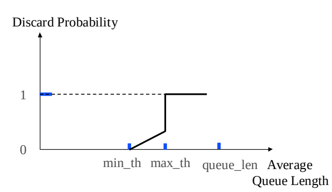
RED

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



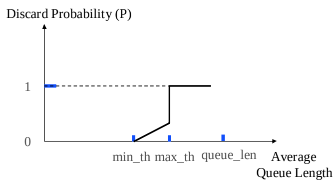
RED (cont'd)

- min_th – minimum threshold
- max_th – maximum threshold
- avg_len – average queue length
  - avg_len = (1-w)*avg_len + w*sample_len



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



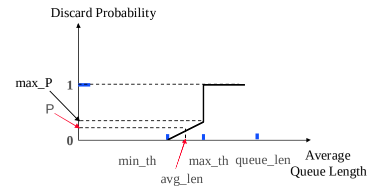
7

RED (cont'd)

- $P = max\_P*(avg\_len - min\_th)/(max\_th - min\_th)$
- Improvements to spread the drops

  $P' = P/(1 - count*P)$, where

  - count – how many packets were consecutively enqueued since last drop



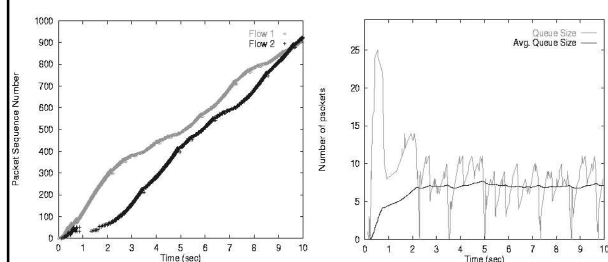
8

RED Advantages

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

9

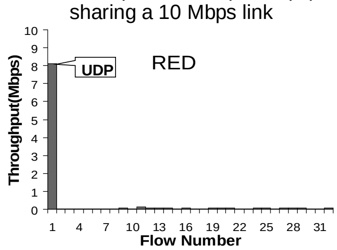
RED Router with Two TCP Sessions



10

Problems with RED

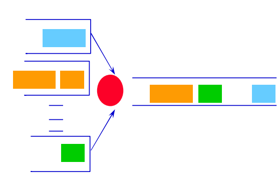
- No protection: if a flow misbehaves it will hurt the other flows
- Example: 1 UDP (10 Mbps) and 31 TCP's sharing a 10 Mbps link



11

Solution?

- Round-robin among different flows [Nagle '87]
  - One queue per flow



12

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

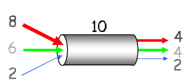
- Denote
  - C – link capacity
  - N – number of flows
  - r_i – arrival rate
- Max-min fair rate computation:
  1. compute C/N
  2. if there are flows i such that $r_i <= C/N$, update C and N

$$C = C - \sum_{i \ s.t \ r_i \le C} r_i$$

  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

- $C = 10$; $r_1 = 8$, $r_2 = 6$, $r_3 = 2$; $N = 3$
- $C/3 = 3.33 \rightarrow C = C - r3 = 8$; $N = 2$
- $C/2 = 4$; $f = 4$

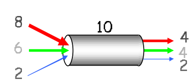


f = 4:
min(8, 4) = 4
min(6, 4) = 4
min(2, 4) = 2

Alternate Way to Compute Fair Rate

- If link congested, compute f such that

$$\sum_i \min(r_i, f) = C$$



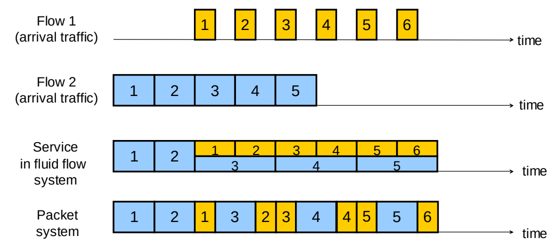
f = 4:
min(8, 4) = 4
min(6, 4) = 4
min(2, 4) = 2

Implementing Fair Queueing

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

19

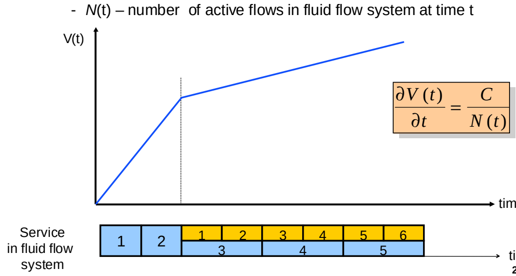
Example



20

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



$$\frac{\partial V(t)}{\partial t} = \frac{C}{N(t)}$$

21

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

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

22

"Weighted Fair Queueing" (WFQ)

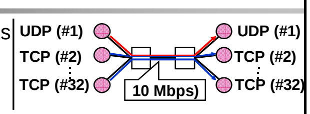
- What if we don't want exact fairness?
  - E.g.,: file servers
- Assign weight w_i to each flow *i*
- And change virtual finishing time

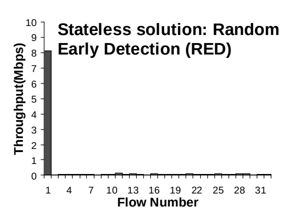
$$F_i^{k+1} = \max(\ V(a_i^k), F_i^k) + \frac{L_i^{k+1}}{w_i}$$

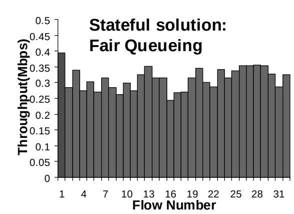
23

Simulation Example

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







24

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

25

Insight

- If each packet of a flow with arrival rate r is forwarded with probability

$$P = \min\left(1, \frac{f}{r}\right)$$

- the rate of flow's forwarded traffic r' is

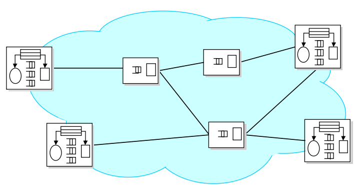
$$r' = r \times P = r \times \min\left(1, \frac{f}{r}\right) = \min(r, f)$$

- No need to maintain per-flow state if r is carried in the packet
  - Need to update rate in packet to r'

26

CSFQ

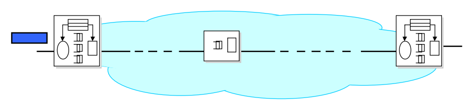
- A contiguous and trusted region of network in which
  - Edge nodes – perform per flow operations
  - Core nodes – do not perform any per flow operations



27

Algorithm Outline

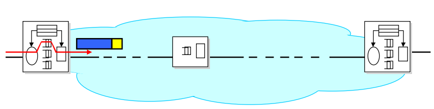
- Ingress nodes: estimate rate r for each flow and insert it in the packets' headers



28

Algorithm Outline

- Ingress nodes: estimate rate r for each flow and insert it in the packets' headers



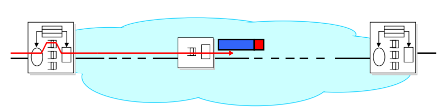
29

Algorithm Outline

- Core node:
  - Compute fair rate f on the output link
  - Enqueue packet with probability

$$P = \min(1, f / r)$$

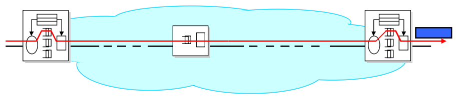
  - Update packet label to $r = \min(r, f)$



30

Algorithm Outline

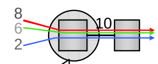
- Egress node: remove state from packet's header

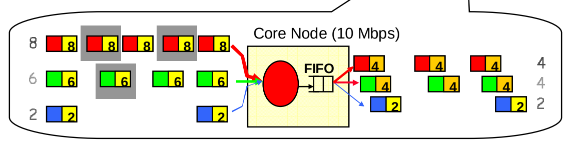


31

Example: CSFQ

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

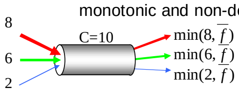




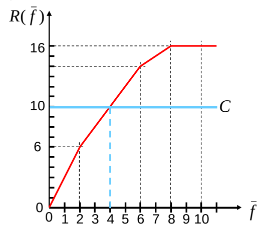
32

Fair Rate Estimation

- **Observation**: rate of aggregate forwarded rate (R) is a monotonic and non-decreasing function of estimated rate ($\bar{f}$)

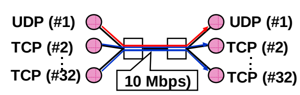


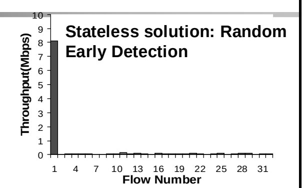
$$R(\bar{f}) = \min(8, \bar{f}) + \min(6, \bar{f}) + \min(2, \bar{f})$$

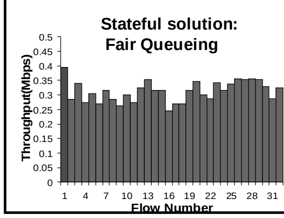


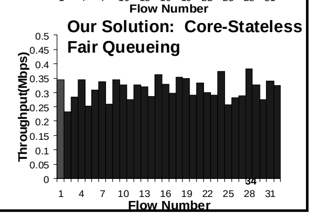
33

Simulation Example









34

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

35

Announcements

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

36