CS640: Introduction to Computer Networks

Aditya Akella

Lecture 20 -
Queuing and Basics of QoS

The Road Ahead

- Queuing Disciplines
- Fair Queuing
- Token Bucket

2

Queuing Disciplines

- Each router must implement some queuing discipline
  - Scheduling discipline
  - Drop policy
- Queuing allocates both bandwidth and buffer space:
  - Bandwidth: which packet to serve (transmit) next
  - Buffer space: which packet to drop next (when required)
- Queuing also affects latency
- Important for QoS; also for best effort

3

Typical Internet Queuing

- FIFO + drop-tail
  - Simplest choice
  - Used widely in the Internet
  - FIFO: scheduling discipline
  - Drop-tail: drop policy
- FIFO (first-in-first-out)
  - Implies single class of traffic, no priority
- Drop-tail
  - Arriving packets get dropped when queue is full regardless of flow or importance

4

FIFO + Drop-tail Problems

- Lock-out problem
  - Drop-tail routers treat bursty traffic poorly
  - Traffic gets synchronized easily → allows a few flows to monopolize the queue space
- Full queues
  - Routers are forced to have have large queues to maintain high utilizations
  - TCP detects congestion from loss
    - Forces network to have long standing queues in steady-state

5

FIFO + Drop-tail Problems

- No policing: send more packets → get more service
  - Lack of isolation among flows
- Synchronization: end hosts react to same events
  - Full queue → empty → Full → empty…
- Poor support for bursty traffic
  - Almost always see burst losses!

6

Active Queue Management

- Design "active" router queue management to facilitate better behavior under congestion
- Objectives: solve FIFO problems, better support for QoS
  - Keep throughput high and delay low
    - High power (throughput/delay)
  - Accommodate bursts
    - Queue size should reflect ability to accept bursts rather than steady-state queuing
  - Research focus: Improve *TCP performance* with minimal hardware changes

7

Lock-out Problem

- Random drop
  - Packet arriving when queue is full causes some random packet to be dropped
- Drop front
  - On full queue, drop packet at head of queue
- Random drop and drop front solve the lock-out problem but not the full-queues problem

8

Full Queues Problem

- Drop packets before queue becomes full (early drop)
- Intuition: notify senders of incipient congestion
  - Example: early random drop (ERD):
    - If qlen > drop level, drop each new packet with fixed probability *p*
    - Does not control misbehaving users

9

Random Early Detection (RED)

- Detect incipient congestion
- Assume hosts respond to lost packets
  - Compliant congestion control
- Avoid window synchronization
  - Randomly mark packets
- Avoid bias against bursty traffic

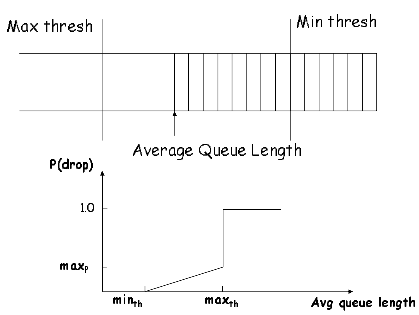
10

RED Algorithm

- Maintain running average of queue length
- If avg < min_{th} do nothing
  - Low queuing, send packets through
- If avg > max_{th}, drop packet
  - Protection from misbehaving sources
- Else mark packet in a manner proportional to queue length
  - Notify sources of incipient congestion

11

RED Operation



12

Fair Queuing: Goals

- How do you protect the most important packets?
  - How do you provide some isolation in general?
  - Simple priority queuing does not help
- Two approaches:
  - Fair Queuing
  - Leaky bucket (with other techniques which we will cover next class)
- FQ key goal: Allocate resources "fairly"
  - Keep separate queue for each flow
- Isolate ill-behaved users
  - Router does not send explicit feedback to source
  - Still needs e2e congestion control
- Still achieve statistical muxing
  - One flow can fill entire pipe if no contenders
  - Work conserving → scheduler never idles link if it has a packet

13

What is "Fairness"?

- At what granularity?
  - Flows, connections, domains?
- What if users have different RTTs/links/etc.
  - TCP is "RTT-Fair"
    - BW inversely proportional to RTT of flow
  - Should they share a link fairly or be TCP-fair?
- Maximize fairness index?
  - Fairness = $(\Sigma x_i)^2/n(\Sigma x_i^2)$ $0<\text{fairness}<1$
- Basically a tough question to answer
  - Typically design mechanisms instead of policy
    - Local notion of fairness, as we will see next
  - User = arbitrary granularity

14

Max-min Fairness

- Allocate user with "small" demand what it wants, evenly divide unused resources to "big" users
- Formally:
  - Resources allocated in terms of increasing demand
  - No source gets resource share larger than its demand
  - Sources with unsatisfied demands get equal share of resource

15

Implementing Max-min Fairness

- Generalized processor sharing
  - Fluid fairness
  - Bitwise round robin among all queues
- Why not simple round robin?
  - Variable packet length → can get more service by sending bigger packets
  - Unfair instantaneous service rate
    - What if arrive just before/after packet departs?

16

Bit-by-bit RR

- Single flow: 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
  - $F_i = S_i+P_i = \max(F_{i-1}, A_i) + P_i$
- Multiple flows: clock ticks when a bit from all active flows is transmitted → round number
  - Can calculate F_i for each packet if number of flows is know at all times
    - Why do we need to know flow count? → need to know A → This can be complicated

17

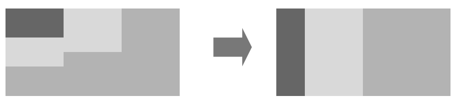
Bit-by-bit RR Illustration

- Not feasible to interleave bits on real networks
  - FQ simulates bit-by-bit RR

18

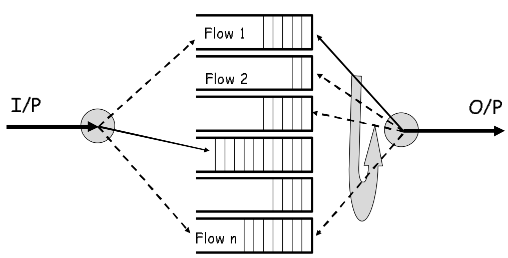
Fair Queuing

- Mapping bit-by-bit schedule onto packet transmission schedule
- Transmit packet with the lowest F_i at any given time
  - How do you compute F_i? As we saw before, this is hard.



19

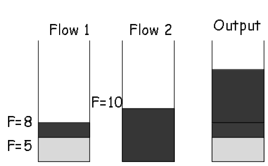
FQ Illustration

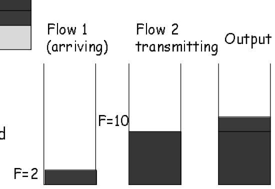


Variation: Weighted Fair Queuing (WFQ) → Flows can have weight
Key to QoS, as we will see in next class.

20

Bit-by-bit RR Example





Cannot preempt packet currently being transmitted

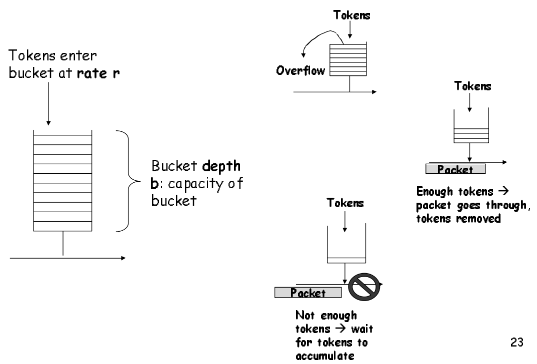
21

Fair Queuing Tradeoffs

- FQ can control congestion by monitoring flows
  - Non-adaptive flows can still be a problem – why?
- Complex state
  - Must keep queue per flow
    - Hard in routers with many flows (e.g., backbone routers)
    - Flow aggregation is a possibility (e.g. do fairness per domain)
- Complex computation
  - Classification into flows may be hard
  - Must keep queues sorted by finish times
  - Must track number of flows at fine time scales

22

Token Bucket for Traffic Policing



23

Token Bucket Characteristics

- On the long run, rate is limited to r
- On the short run, a burst of size b can be sent
- Amount of traffic entering at interval T is bounded by:
  - Traffic = b + r*T
  - Can provide a lose sense of isolation mong flows.

24