15-744: Computer Networking

L-5 Fair Queuing

Announcements

- Project proposals: please e-mail!
  - Before/during/after class is fine
- Bin Fan will likely be back next week
- Change in lecture schedule:
  - Friday: Router design
  - Monday: Router algorithms
    - Another volunteer for presentation?

2

Fair Queuing

- Fair Queuing
- Core-stateless Fair queuing
- Assigned reading
  - Analysis and Simulation of a Fair Queueing Algorithm, Internetworking: Research and Experience
  - Congestion Control for High Bandwidth-Delay Product Networks (XTP - 2 sections)

3

Overview

- TCP and queues
- Queuing disciplines
- RED
- Fair-queuing
- Core-stateless FQ
- XCP

4

Example

- 10Gb/s linecard
  - Requires 300Mbytes of buffering.
  - Read and write 40 byte packet every 32ns.
- Memory technologies
  - DRAM: require 4 devices, but too slow.
  - SRAM: require 80 devices, 1kW, $2000.
- Problem gets harder at 40Gb/s
  - Hence RLDRAM, FCRAM, etc.

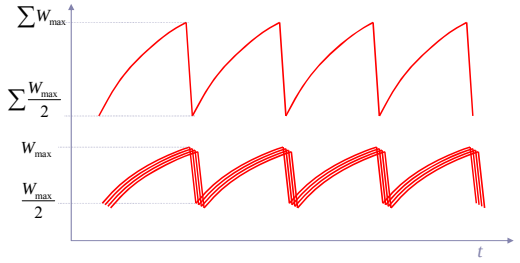
5

Rule-of-thumb

- Rule-of-thumb makes sense for one flow
- Typical backbone link has > 20,000 flows
- Does the rule-of-thumb still hold?

6

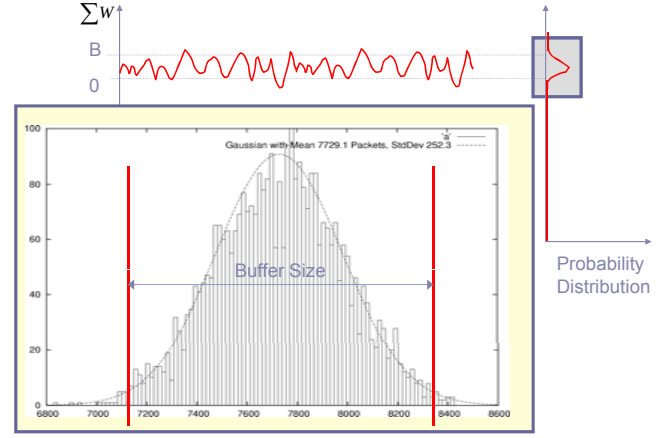
If flows are synchronized



- Aggregate window has same dynamics
- Therefore buffer occupancy has same dynamics
- Rule-of-thumb still holds.

7

If flows are not synchronized



8

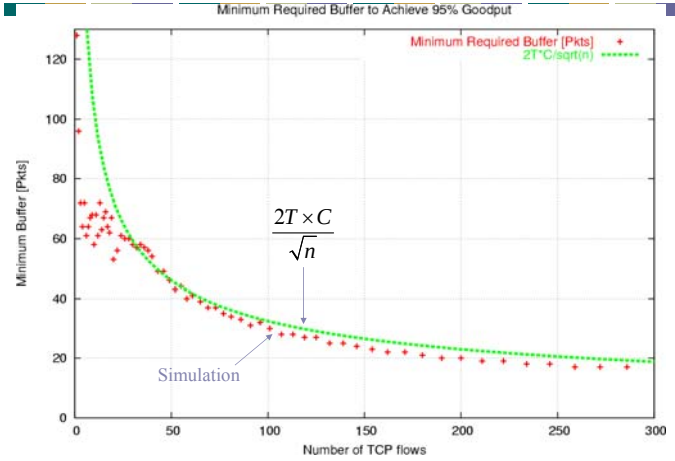
Central Limit Theorem

- CLT tells us that the more variables (Congestion Windows of Flows) we have, the narrower the Gaussian (Fluctuation of sum of windows)
  - Width of Gaussian decreases with $\frac{1}{\sqrt{n}}$
  - Buffer size should also decreases with $\frac{1}{\sqrt{n}}$

$$B \rightarrow \frac{B_{n=1}}{\sqrt{n}} = \frac{2T \times C}{\sqrt{n}}$$

9

Required buffer size



10

Overview

- TCP and queues
- Queuing disciplines
- RED
- Fair-queuing
- Core-stateless FQ
- XCP

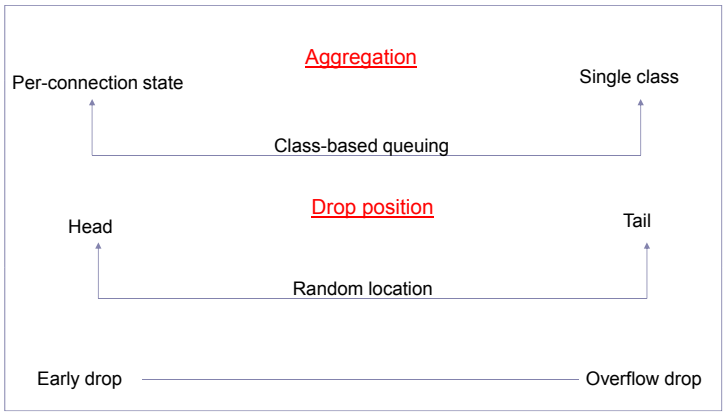
11

Queuing Disciplines

- Each router must implement some queuing discipline
- 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

12

Packet Drop Dimensions



13

Typical Internet Queuing

- FIFO + drop-tail
  - Simplest choice
  - Used widely in the Internet
- FIFO (first-in-first-out)
  - Implies single class of traffic
- Drop-tail
  - Arriving packets get dropped when queue is full regardless of flow or importance
- Important distinction:
  - FIFO: scheduling discipline
  - Drop-tail: drop policy

14

FIFO + Drop-tail Problems

- Leaves responsibility of congestion control to edges (e.g., TCP)
- Does not separate between different flows
- No policing: send more packets → get more service
- Synchronization: end hosts react to same events

15

Active Queue Management

- Design active router queue management to aid congestion control
- Why?
  - Routers can distinguish between propagation and persistent queuing delays
  - Routers can decide on transient congestion, based on workload

16

Active Queue Designs

- Modify both router and hosts
  - DECbit – congestion bit in packet header
- Modify router, hosts use TCP
  - Fair queuing
    - Per-connection buffer allocation
  - RED (Random Early Detection)
    - Drop packet or set bit in packet header as soon as congestion is starting

17

Overview

- TCP and queues
- Queuing disciplines
- RED
- Fair-queuing
- Core-stateless FQ
- XCP

18

Internet Problems

- 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
- Lock-out problem
  - Drop-tail routers treat bursty traffic poorly
  - Traffic gets synchronized easily → allows a few flows to monopolize the queue space

19

Design Objectives

- Keep throughput high and delay low
- Accommodate bursts
- Queue size should reflect ability to accept bursts rather than steady-state queuing
- Improve TCP performance with minimal hardware changes

20

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

21

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

22

Random Early Detection (RED)

- Detect incipient congestion, allow bursts
- Keep power (throughput/delay) high
  - Keep average queue size low
  - Assume hosts respond to lost packets
- Avoid window synchronization
  - Randomly mark packets
- Avoid bias against bursty traffic
- Some protection against ill-behaved users

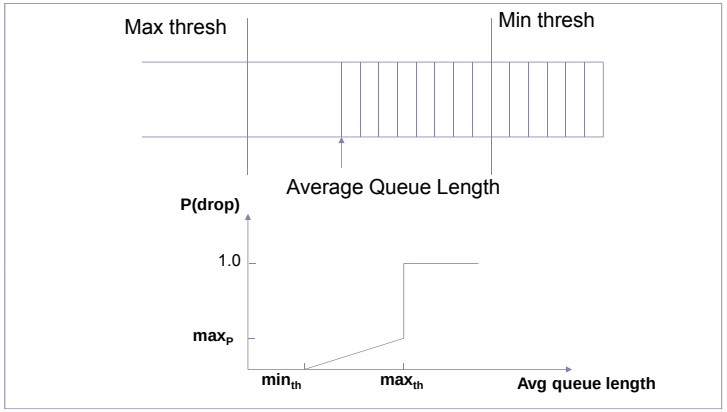
23

RED Algorithm

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

24

RED Operation



25

RED Algorithm

- Maintain running average of queue length
  - Byte mode vs. packet mode – why?
- For each packet arrival
  - Calculate average queue size (avg)
  - If $min_{th} \leq avgq < max_{th}$
    - Calculate probability P_a
    - With probability P_a
      - Mark the arriving packet
    - Else if $max_{th} \leq avg$
      - Mark the arriving packet

26

Queue Estimation

- Standard EWMA: $avgq = (1-w_q)\ avgq + w_q qlen$
  - Special fix for idle periods – why?
- Upper bound on w_q depends on min_{th}
  - Want to ignore transient congestion
  - Can calculate the queue average if a burst arrives
    - Set w_q such that certain burst size does not exceed min_{th}
- Lower bound on w_q to detect congestion relatively quickly
- Typical $w_q = 0.002$

27

Thresholds

- min_{th} determined by the utilization requirement
  - Tradeoff between queuing delay and utilization
- Relationship between max_{th} and min_{th}
  - Want to ensure that feedback has enough time to make difference in load
  - Depends on average queue increase in one RTT
  - Paper suggest ratio of 2
    - Current rule of thumb is factor of 3

28

Packet Marking

- max_p is reflective of typical loss rates
- Paper uses 0.02
  - 0.1 is more realistic value
- If network needs marking of 20-30% then need to buy a better link!
- Gentle variant of RED (recommended)
  - Vary drop rate from max_p to 1 as the avgq varies from max_{th} to 2* max_{th}
  - More robust to setting of max_{th} and max_p

29

Extending RED for Flow Isolation

- Problem: what to do with non-cooperative flows?
- Fair queuing achieves isolation using per-flow state – expensive at backbone routers
  - How can we isolate unresponsive flows without per-flow state?
- RED penalty box
  - Monitor history for packet drops, identify flows that use disproportionate bandwidth
  - Isolate and punish those flows

30

Overview

- TCP and queues
- Queuing disciplines
- RED
- Fair-queuing
- Core-stateless FQ
- XCP

31

Fairness Goals

- Allocate resources fairly
- 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

32

What is Fairness?

- At what granularity?
  - Flows, connections, domains?
- What if users have different RTTs/links/etc.
  - Should it share a link fairly or be TCP fair?
- Maximize fairness index?
  - Fairness = $(\Sigma x_i)^2/n(\Sigma x_i^2)$ 0<fairness<1
- Basically a tough question to answer – typically design mechanisms instead of policy
  - User = arbitrary granularity

Max-min Fairness

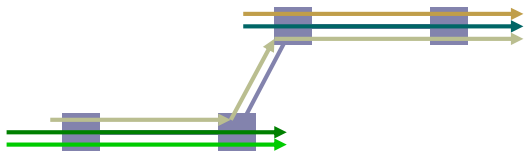
- Allocate user with "small" demand what it wants, evenly divide unused resources to "big" users
- Formally:
  - Resources allocated in terms of increasing demand
  - No source gets resource share larger than its demand
  - Sources with unsatisfied demands get equal share of resource

Max-min Fairness Example

- Assume sources 1..n, with resource demands X1..Xn in ascending order
- Assume channel capacity C.
  - Give C/n to X1; if this is more than X1 wants, divide excess (C/n - X1) to other sources: each gets C/n + (C/n - X1)/(n-1)
  - If this is larger than what X2 wants, repeat process

Max-Min Fair Sharing Example

$$r_{fair} = \frac{C - \sum_{else} r_i}{n_{here}}$$



Assume 10 Mbs links

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?

37

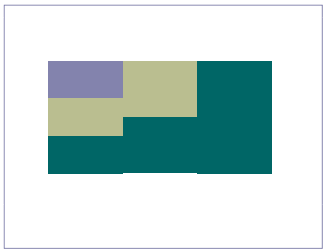
Bit-by-bit RR

- 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
    - This can be complicated

38

Bit-by-bit RR Illustration

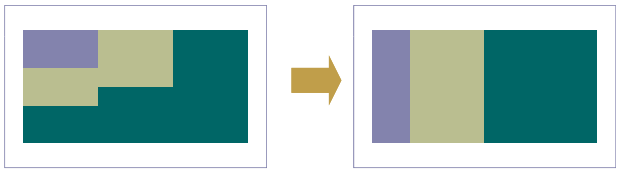
- Not feasible to interleave bits on real networks
  - FQ simulates bit-by-bit RR



39

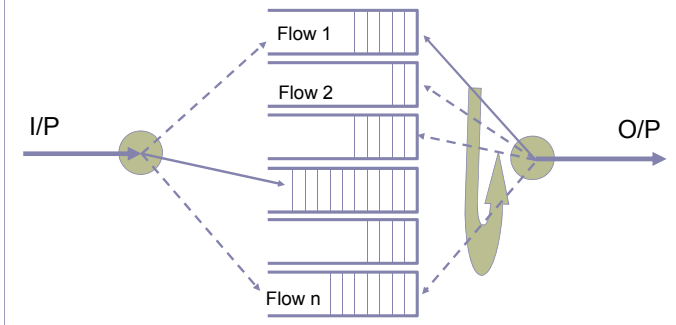
Fair Queuing

- 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?



40

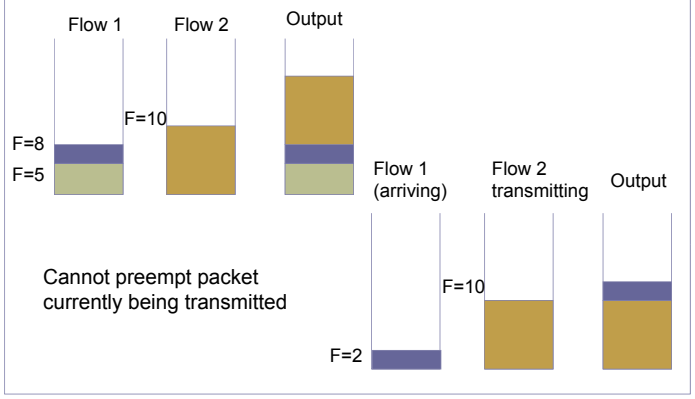
FQ Illustration



Variation: Weighted Fair Queuing (WFQ)

41

Bit-by-bit RR Example



42

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
  - Finish times change whenever the flow count changes

43

Overview

- TCP and queues
- Queuing disciplines
- RED
- Fair-queuing
- Core-stateless FQ
  - Not discussed in class – FYI only
- XCP

44

Core-Stateless Fair Queuing

- Key problem with FQ is core routers
  - Must maintain state for 1000's of flows
  - Must update state at Gbps line speeds
- CSFQ (Core-Stateless FQ) objectives
  - Edge routers should do complex tasks since they have fewer flows
  - Core routers can do simple tasks
    - No per-flow state/processing → this means that core routers can only decide on dropping packets not on order of processing
    - Can only provide max-min bandwidth fairness not delay allocation

45

Core-Stateless Fair Queuing

- Edge routers keep state about flows and do computation when packet arrives
- DPS (Dynamic Packet State)
  - Edge routers label packets with the result of state lookup and computation
- Core routers use DPS and local measurements to control processing of packets

46

Edge Router Behavior

- Monitor each flow i to measure its arrival rate (r_i)
  - EWMA of rate
  - Non-constant EWMA constant
    - $e^{-T/K}$ where T = current interarrival, K = constant
    - Helps adapt to different packet sizes and arrival patterns
- Rate is attached to each packet

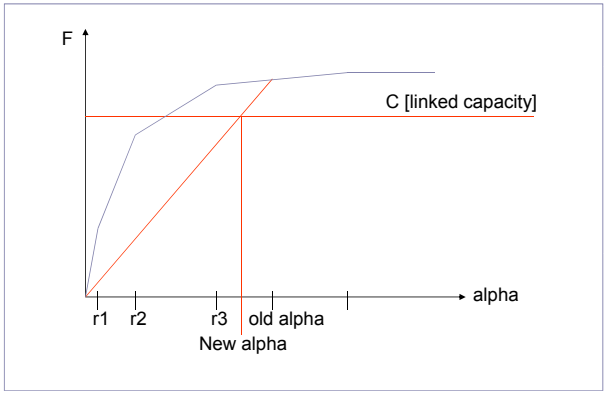
47

Core Router Behavior

- Keep track of fair share rate α
  - Increasing α does not increase load (F) by N * α
  - $F(\alpha) = \Sigma_i \min(r_i, \alpha)$ → what does this look like?
  - Periodically update α
  - Keep track of current arrival rate
    - Only update α if entire period was congested or uncongested
- Drop probability for packet = max(1- α/r, 0)

48

F vs. Alpha



49

Estimating Fair Share

- Need F(α) = capacity = C
  - Can't keep map of F(α) values → would require per flow state
  - Since F(α) is concave, piecewise-linear
    - F(0) = 0 and F(α) = current accepted rate = F_c
    - $F(\alpha) = F_c / \alpha$
    - $F(\alpha_{new}) = C \rightarrow \alpha_{new} = \alpha_{old} * C/F_c$
- What if a mistake was made?
  - Forced into dropping packets due to buffer capacity
  - When queue overflows α is decreased slightly

50

Other Issues

- Punishing fire-hoses – why?
  - Easy to keep track of in a FQ scheme
- What are the real edges in such a scheme?
  - Must trust edges to mark traffic accurately
  - Could do some statistical sampling to see if edge was marking accurately

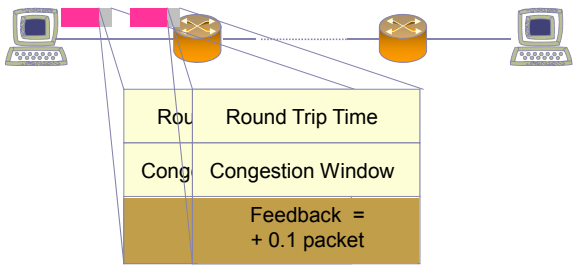
51

Overview

- TCP and queues
- Queuing disciplines
- RED
- Fair-queuing
- Core-stateless FQ
- XCP
  - See also slides by Nicolas Feltman

52

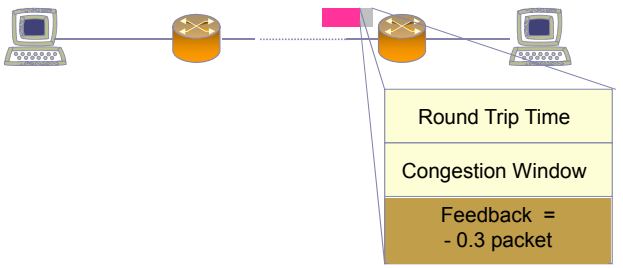
How does XCP Work?



Congestion Header

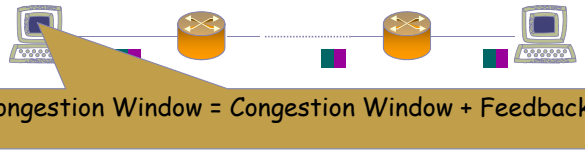
53

How does XCP Work?



54

How does XCP Work?

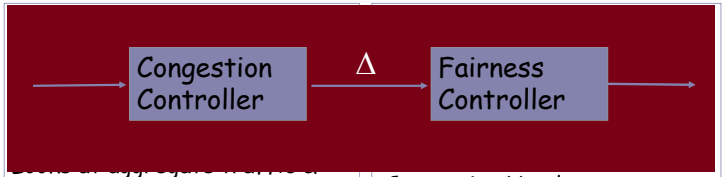


XCP extends ECN and CSFQ

Routers compute feedback without any per-flow state

55

How Does an XCP Router Compute the Feedback?



queue

MIMD

Algorithm:

Aggregate traffic changes by Δ

Δ ~ Spare Bandwidth

Δ ~ - Queue Size

So, $\Delta = \alpha\, d_{avg}\, Spare - \beta\, Queue$

Congestion He

AIMD

Algorithm:

If Δ > 0 ⇒ Divide Δ equally between flows

If Δ < 0 ⇒ Divide Δ between flows proportionally to their current rates

56

Getting the devil out of the details …

Congestion Controller

$\Delta = \alpha\, d_{avg}\, Spare - \beta\, Queue$

Theorem: System converges to optimal utilization (i.e., stable) for any link bandwidth, delay, number of sources if:

$$0 < \alpha < \frac{\pi}{4\sqrt{2}} \quad and \quad \beta = \alpha^2\sqrt{2}$$

No Parameter Tuning

Fairness Controller

Algorithm:

If $\Delta > 0 \Rightarrow$ Divide Δ equally between flows

If $\Delta < 0 \Rightarrow$ Divide Δ between flows proportionally to their current rates

Need to estimate number of flows N

$$N = \sum_{pkts\ in\ T} \frac{1}{T \times (Cwnd_{pkt} / RTT_{pkt})}$$

RTT_{pkt}: Round Trip Time in header

No Per-Flow State

57

Discussion

- RED
  - Parameter settings
- RED vs. FQ
  - How much do we need per flow tracking? At what cost?
- FQ vs. XCP/CSFQ
  - Is coarse-grained fairness sufficient?
  - Misbehaving routers/trusting the edge
  - Deployment (and incentives)
  - How painful is FQ
- XCP vs CSFQ
  - What are the key differences
- Granularity of fairness
  - Mechanism vs. policy → will see this in QoS

58

Important Lessons

- How does TCP implement AIMD?
  - Sliding window, slow start & ack clocking
  - How to maintain ack clocking during loss recovery → fast recovery
- How does TCP fully utilize a link?
  - Role of router buffers
- TCP alternatives
  - TCP being used in new/unexpected ways
  - Key changes needed

59