15-744: Computer Networking

L-5 Fair Queuing

Fair Queuing

- Fair Queuing
- Core-stateless Fair queuing
- Assigned reading
  - [DKS90] Analysis and Simulation of a Fair Queueing Algorithm, Internetworking: Research and Experience
  - [SSZ98] Core-Stateless Fair Queueing: Achieving Approximately Fair Allocations in High Speed Networks

Overview

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

Example

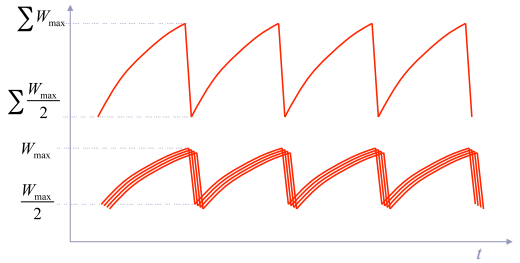
- 10Gb/s linecard
  - Requires 300Mbytes of buffering.
  - Read and write 40 byte packet every 32ns.
- Memory technologies
  - DRAM: require 4 devices, but too slow.
  - SRAM: require 80 devices, 1kW, $2000.
- Problem gets harder at 40Gb/s
  - Hence RLDRAM, FCRAM, etc.

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?

5

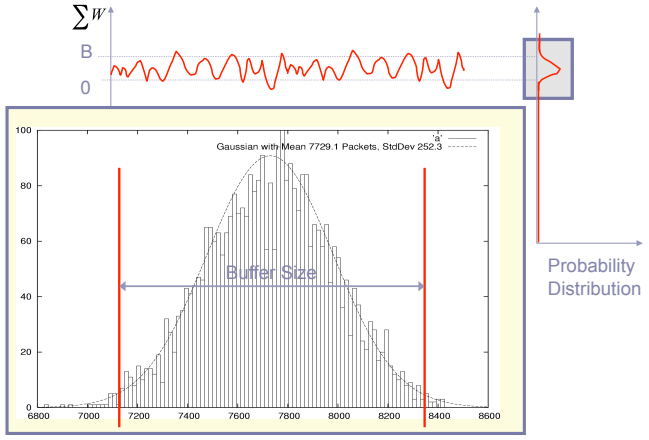
If flows are synchronized



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

6

If flows are not synchronized



7

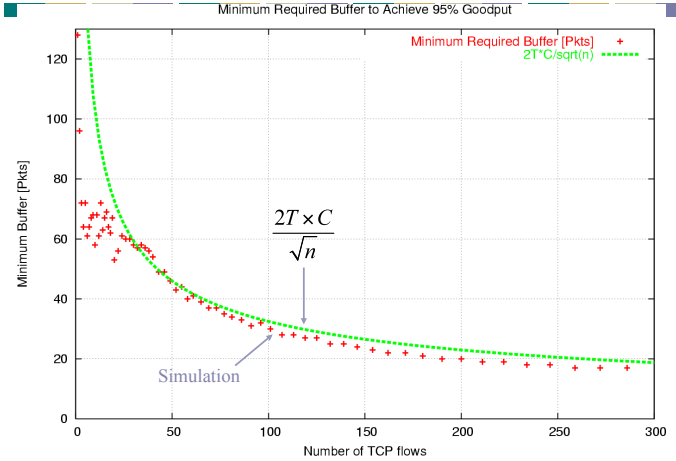
Central Limit Theorem

- 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 $\frac{1}{\sqrt{n}}$ $\frac{1}{\sqrt{n}}$
  - Buffer size should also decreases with

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

8

Required buffer size



9

Overview

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

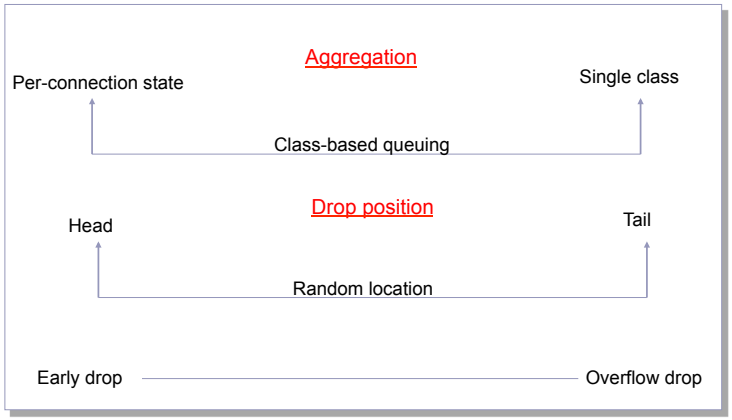
10

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

11

Packet Drop Dimensions



12

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

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

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

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

Overview

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

17

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

18

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

19

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

20

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

21

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

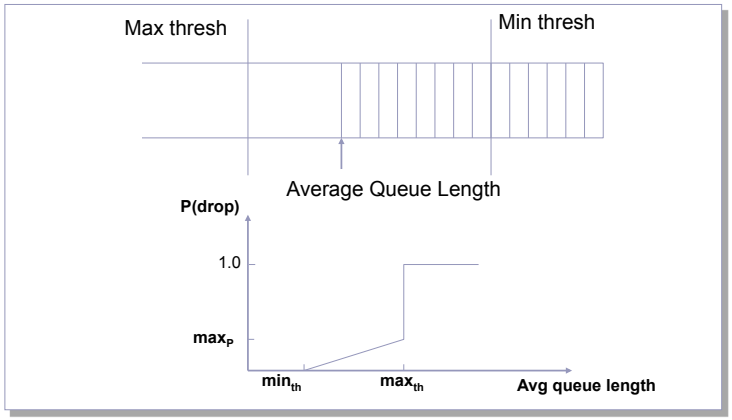
22

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

23

RED Operation



24

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

25

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 $w_q = 0.002$

26

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

27

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

28

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

29

Stochastic Fair Blue

- Same objective as RED Penalty Box
  - Identify and penalize misbehaving flows
- Create L hashes with N bins each
  - Each bin keeps track of separate marking rate (p_m)
  - Rate is updated using standard technique and a bin size
  - Flow uses minimum p_m of all L bins it belongs to
  - Non-misbehaving flows hopefully belong to at least one bin without a bad flow
    - Large numbers of bad flows may cause false positives

30

Stochastic Fair Blue

- False positives can continuously penalize same flow
- Solution: moving hash function over time
  - Bad flow no longer shares bin with same flows
  - Is history reset →does bad flow get to make trouble until detected again?
    - No, can perform hash warmup in background

31

Overview

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

32

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

33

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

34

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

35

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

36

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

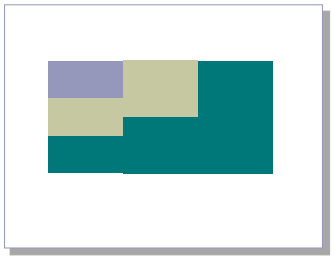
Bit-by-bit RR

- 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

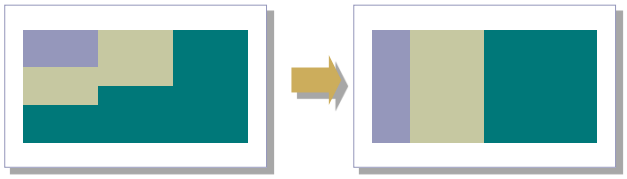
- Not feasible to interleave bits on real networks
  - FQ simulates bit-by-bit RR



39

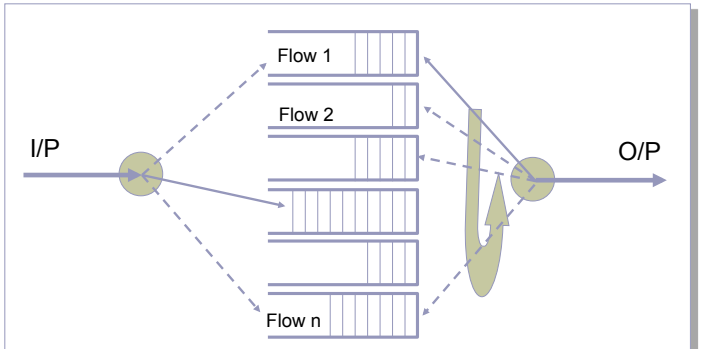
Fair Queuing

- 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

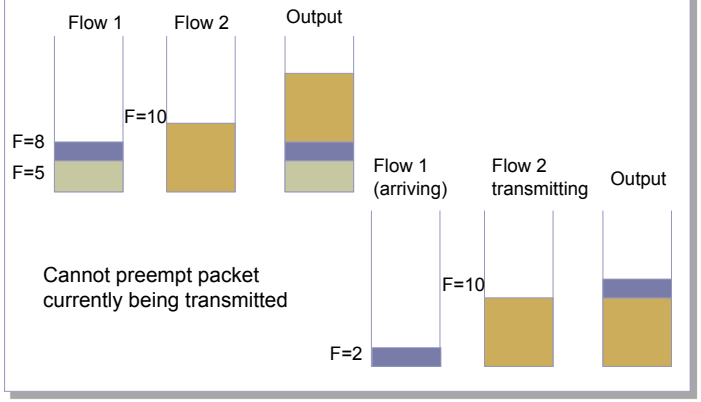
FQ Illustration



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
- 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

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

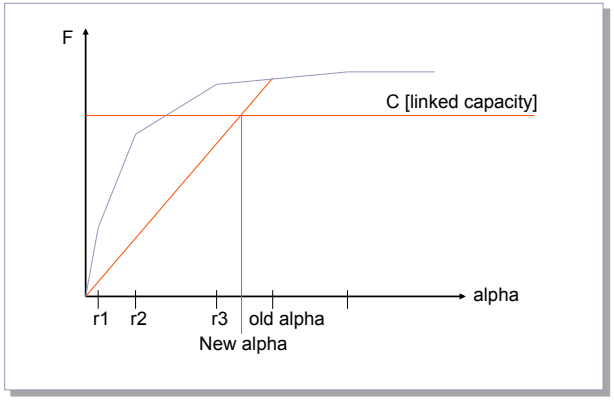
Edge Router Behavior

- 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

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)

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

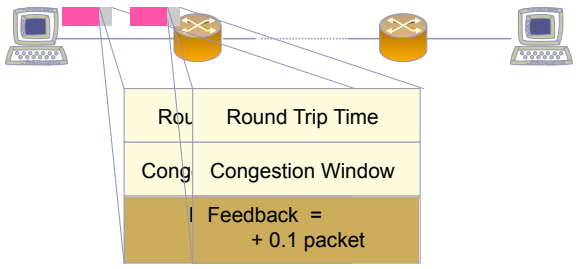
51

Overview

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

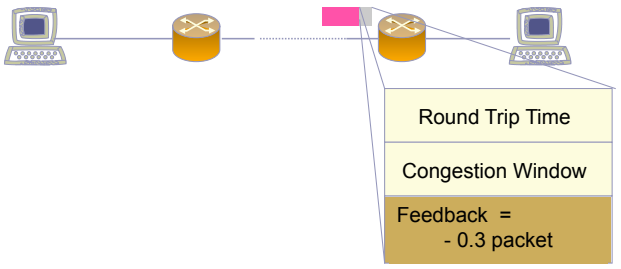
52

How does XCP Work?

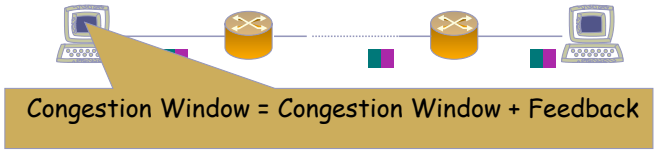


Congestion Header

How does XCP Work?



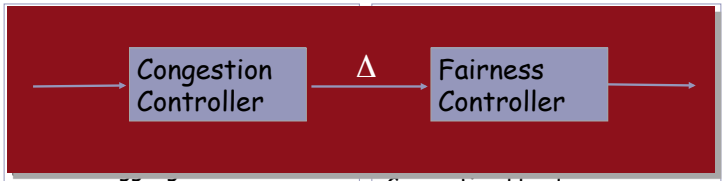
How does XCP Work?



XCP extends ECN and CSFQ

Routers compute feedback without any per-flow state

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

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

Lessons

- Fairness and isolation in routers
  - Why is this hard?
  - What does it achieve – e.g. do we still need congestion control?
- Routers
  - FIFO, drop-tail interacts poorly with TCP
  - Various schemes to desynchronize flows and control loss rate (e.g. RED)
- Fair-queuing
  - Clean resource allocation to flows
  - Complex packet classification and scheduling
- Core-stateless FQ & XCP
  - Coarse-grain fairness
  - Carrying packet state can reduce complexity

60

Next Lecture: TCP & Routers

- RED
- XCP
- Assigned reading
  - [FJ93] Random Early Detection Gateways for Congestion Avoidance
  - [KHR02] Congestion Control for High Bandwidth-Delay Product Networks

61

EXTRA SLIDES

The rest of the slides are FYI

Overview

- Fairness
- Fair-queuing
- Core-stateless FQ
- Other FQ variants

63

Delay Allocation in FQ

- Reduce delay for flows using less than fair share
  - Advance finish times for sources whose queues drain temporarily
- Schedule based on B_i instead of F_i
  - $F_i = P_i + \max(F_{i-1}, A_i) \rightarrow B_i = P_i + \max(F_{i-1}, A_i - \delta)$
  - If $A_i < F_{i-1}$, conversation is active and δ has no effect
  - If $A_i > F_{i-1}$, conversation is inactive and δ determines how much history to take into account
    - Infrequent senders do better when history is used

64

Stochastic Fair Queuing

- Compute a hash on each packet
- Instead of per-flow queue have a queue per hash bin
- An aggressive flow steals traffic from other flows in the same hash
- Queues serviced in round-robin fashion
  - Has problems with packet size unfairness
- Memory allocation across all queues
  - When no free buffers, drop packet from longest queue

65

Deficit Round Robin

- Each queue is allowed to send Q bytes per round
- If Q bytes are not sent (because packet is too large) deficit counter of queue keeps track of unused portion
- If queue is empty, deficit counter is reset to 0
- Uses hash bins like Stochastic FQ
- Similar behavior as FQ but computationally simpler

66

Self-clocked Fair Queuing

- Virtual time to make computation of finish time easier
- Problem with basic FQ
  - Need be able to know which flows are really backlogged
    - They may not have packet queued because they were serviced earlier in mapping of bit-by-bit to packet
    - This is necessary to know how bits sent map onto rounds
    - Mapping of real time to round is piecewise linear → however slope can change often

67

Self-clocked FQ

- Use the finish time of the packet being serviced as the virtual time
  - The difference in this virtual time and the real round number can be unbounded
- Amount of service to backlogged flows is bounded by factor of 2

68

Start-time Fair Queuing

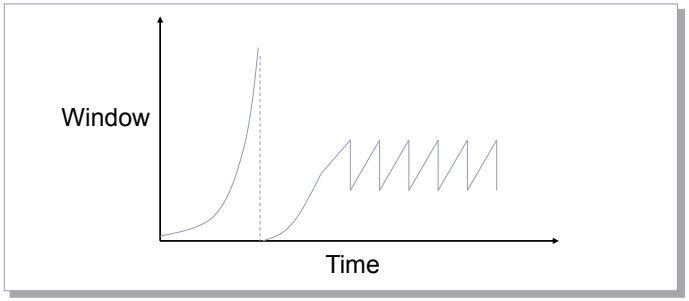
- Packets are scheduled in order of their start not finish times
- Self-clocked → virtual time = start time of packet in service
- Main advantage → can handle variable rate service better than other schemes

TCP Modeling

- Given the congestion behavior of TCP can we predict what type of performance we should get?
- What are the important factors
  - Loss rate
    - Affects how often window is reduced
  - RTT
    - Affects increase rate and relates BW to window
  - RTO
    - Affects performance during loss recovery
  - MSS
    - Affects increase rate

Overall TCP Behavior

- Let's concentrate on steady state behavior with no timeouts and perfect loss recovery



Simple TCP Model

- Some additional assumptions
  - Fixed RTT
  - No delayed ACKs
- In steady state, TCP losses packet each time window reaches W packets
  - Window drops to W/2 packets
  - Each RTT window increases by 1 packet→W/2 * RTT before next loss
  - BW = MSS * avg window/RTT =
    - MSS * (W + W/2)/(2 * RTT)
    - .75 * MSS * W / RTT

Simple Loss Model

- What was the loss rate?
  - Packets transferred between losses =
    - Avg BW * time =
    - (.75 W/RTT) * (W/2 * RTT) = $3W^2/8$
  - 1 packet lost → loss rate = p = $8/3W^2$
  - W = sqrt(8 / (3 * loss rate))
- BW = .75 * MSS * W / RTT
  - BW = MSS / (RTT * sqrt (2/3p))

73

TCP Friendliness

- What does it mean to be TCP friendly?
  - TCP is not going away
  - Any new congestion control must compete with TCP flows
    - Should not clobber TCP flows and grab bulk of link
    - Should also be able to hold its own, i.e. grab its fair share, or it will never become popular
- How is this quantified/shown?
  - Has evolved into evaluating loss/throughput behavior
  - If it shows 1/sqrt(p) behavior it is ok
  - But is this really true?

74