15-744: Computer Networking

L-5 TCP & Routers

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

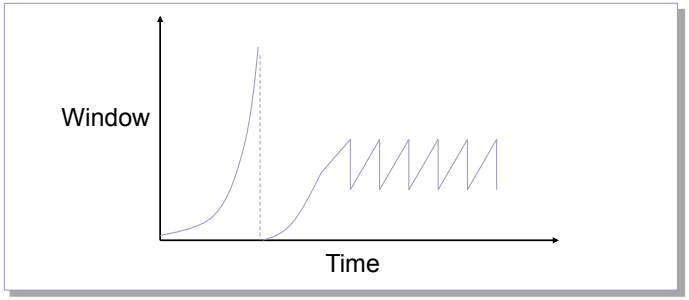
- TCP modeling
- Fairness
- Fair-queuing
- Core-stateless FQ

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



5

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

6

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))

7

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?

8

TCP Performance

- Can TCP saturate a link?
- Congestion control
  - Increase utilization until… link becomes congested
  - React by decreasing window by 50%
  - Window is proportional to rate * RTT
- Doesn't this mean that the network oscillates between 50 and 100% utilization?
  - Average utilization = 75%??
  - No…this is *not* right!

9

TCP Congestion Control

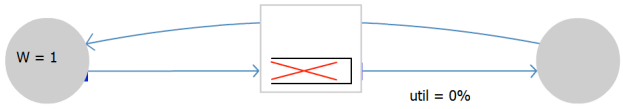
Rule for adjusting W
- If an ACK is received: W ← W+1/W
- If a packet is lost: W ← W/2

Only W packets may be outstanding

10

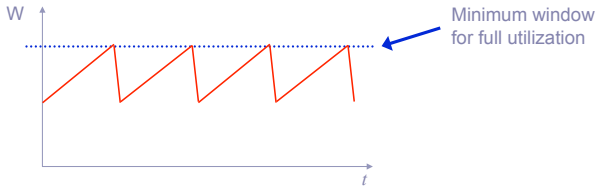
Single TCP Flow

Router without buffers



11

Summary Unbuffered Link



- The router can't fully utilize the link
  - If the window is too small, link is not full
  - If the link is full, next window increase causes drop
  - With no buffer it still achieves 75% utilization

12

TCP Performance

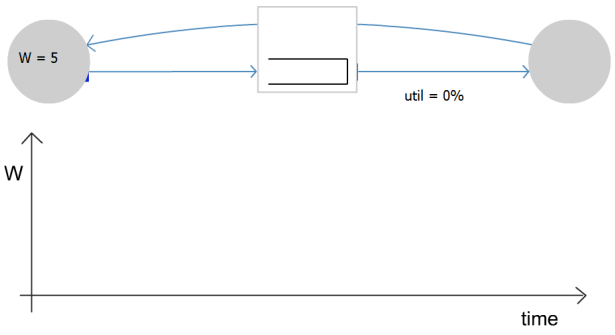
- In the real world, router queues play important role
  - Window is proportional to rate * RTT
    - But, RTT changes as well the window
  - Window to fill links = propagation RTT * bottleneck bandwidth
    - If window is larger, packets sit in queue on bottleneck link

TCP Performance

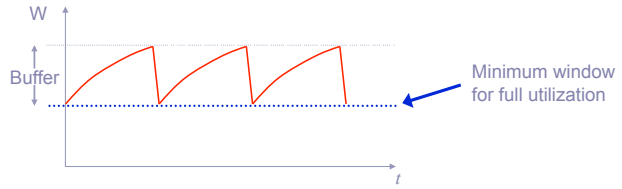
- If we have a large router queue → can get 100% utilization
  - But, router queues can cause large delays
- How big does the queue need to be?
  - Windows vary from W → W/2
    - Must make sure that link is always full
    - W/2 > RTT * BW
    - W = RTT * BW + Qsize
    - Therefore, Qsize > RTT * BW
  - Ensures 100% utilization
  - Delay?
    - Varies between RTT and 2 * RTT

Single TCP Flow

Router with large enough buffers for full link utilization



Summary Buffered Link



- With sufficient buffering we achieve full link utilization
  - The window is always above the critical threshold
  - Buffer absorbs changes in window size
    - Buffer Size = Height of TCP Sawtooth
    - Minimum buffer size needed is 2T*C
  - This is the origin of the rule-of-thumb

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.

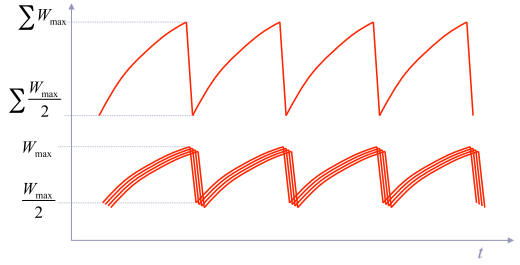
17

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?

18

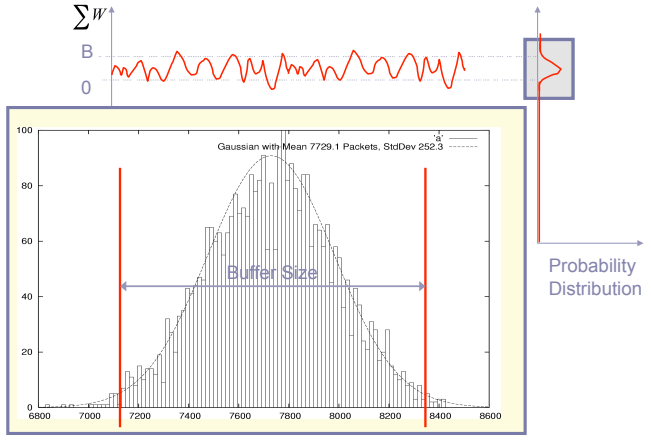
If flows are synchronized



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

19

If flows are not synchronized



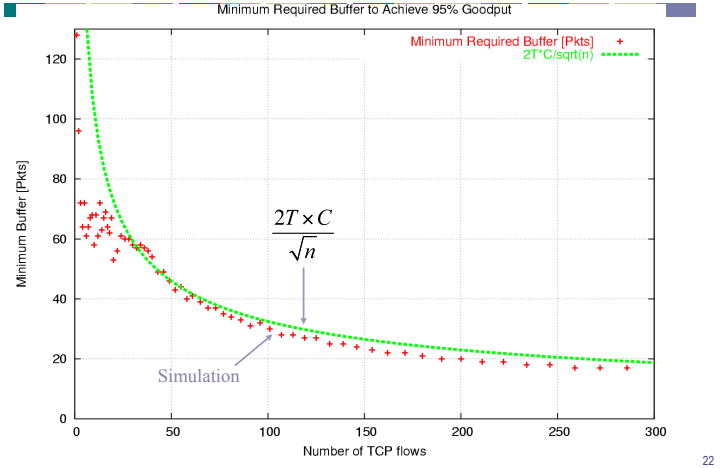
20

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 $\frac{1}{\sqrt{n}}$

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

Required buffer size



Overview

- TCP modeling
- Fairness
- Fair-queuing
- Core-stateless FQ

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

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

25

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

26

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

27

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?

28

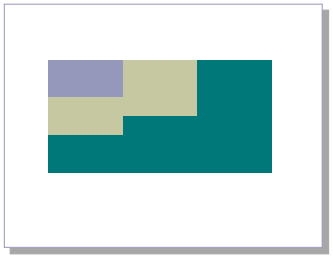
Bit-by-bit RR

- 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

29

Bit-by-bit RR Illustration

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



30

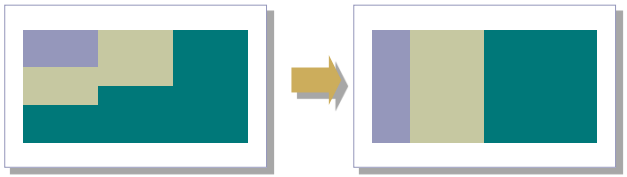
Overview

- TCP modeling
- Fairness
- Fair-queuing
- Core-stateless FQ

31

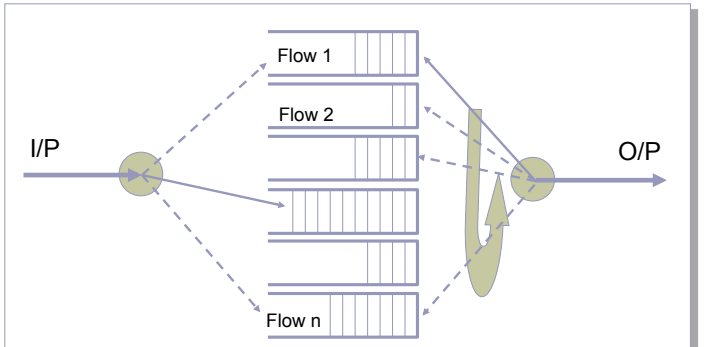
Fair Queuing

- 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?



32

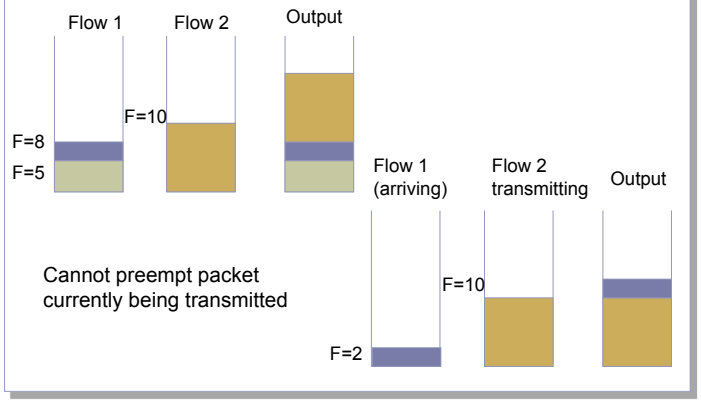
FQ Illustration



Variation: Weighted Fair Queuing (WFQ)

33

Bit-by-bit RR Example



34

Delay Allocation

- 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

35

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

36

Discussion Comments

- Granularity of fairness
  - Mechanism vs. policy → will see this in QoS
- Hard to understand
- Complexity – how bad is it?

Overview

- TCP modeling
- Fairness
- Fair-queuing
- Core-stateless FQ

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

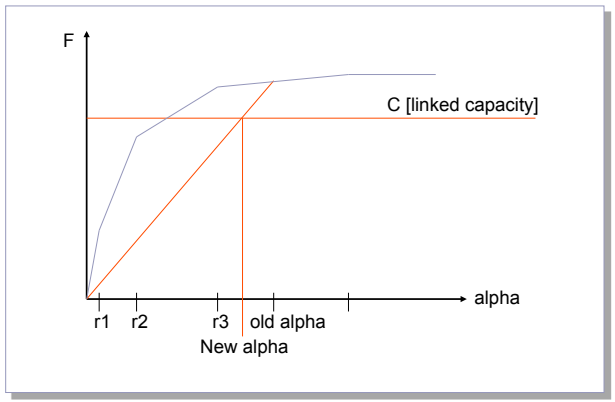
41

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)

42

F vs. Alpha



43

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

44

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

45

Discussion Comments

- Exponential averaging
- Latency properties
- Hand-wavy numbers
- Trusting the edge

46

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
- Fairness and isolation in routers
  - Why is this hard?
  - What does it achieve – e.g. do we still need congestion control?

47

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

48

EXTRA SLIDES

The rest of the slides are FYI

Overview

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

50

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

51

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

52

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

53

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

54

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

55