CS 268: Computer Networking

L-5 Router Queue Management

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

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

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 = \( \frac{\sum x_i^2}{n(\sum x_i^2)^2} \), \( 0 < \text{fairness} < 1 \)
- 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

Implementing max-min Fairness

• Generalized processor sharing
  • Fluid fairness
  • Bitwise round robin among all queues
• Why not simple round robin?
  • Variable packet length $\rightarrow$ can get more service by sending bigger packets
  • Unfair instantaneous service rate
    • What if arrive just before/after packet departs?
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 $\rightarrow$ round number
  • Can calculate $F_i$ for each packet if number of flows is known at all times
    • This can be complicated

Bit-by-bit RR Illustration

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

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

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 $F_i$?
**FQ Illustration**

Variation: Weighted Fair Queuing (WFQ)

**Bit-by-bit RR Example**

Cannot preempt packet currently being transmitted
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 $\delta$ has no effect
  - If $A_i > F_{i-1}$, conversation is inactive and $\delta$ determines how much history to take into account
    - Infrequent senders do better when history is used

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
Discussion Comments

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

Overview

- Fairness
- Fair-queuing
- Core-stateless FQ
- Other FQ variants
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 \(\Rightarrow\) 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 $T = \text{current interarrival}$, $K = \text{constant}$
    - Helps adapt to different packet sizes and arrival patterns
  - Rate is attached to each packet

Core Router Behavior

- Keep track of fair share rate $\lambda$
  - Increasing $\lambda$ does not increase load ($F$) by $N * \lambda$
  - $F(\lambda) = \lambda \min(r_i, \lambda) \rightarrow$ what does this look like?
  - Periodically update $\lambda$
  - Keep track of current arrival rate
    - Only update $\lambda$ if entire period was congested or uncongested
- Drop probability for packet = $\max(1 - \frac{\lambda}{r}, 0)$
F vs. Alpha

Estimating Fair Share

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

Discussion Comments

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

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

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
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

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
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

Start-time Fair Queuing

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

Class Project

• End goal → a possibly publishable research paper
  • Collect and analyze network traffic
  • Propose a network architectural mechanism and evaluate it (usually via simulation)
  • New applications of well-developed network technology usually of interest
    • Networks with new capabilities, restrictions, impairments
    • E.g., satellite (long delay), wireless (high loss rates), sensor (power considerations), datacenter (low latency, particular traffic patterns)
  Security, architecture, other dimensions of networking
Class Project

- Group size ➔ very strong preference for 2

- Project meetings
  - Fri 9/18: Sign-ups to meet up with Randy to discuss initial ideas
  - 10-15 min meetings to discuss project ideas and get feedback

- Proposal (Tu 9/29)
  - Short project proposal presentations
    - Basic idea
    - Description of some related work
    - Rough timeline
    - Necessary/requested resources

- Checkpoint (Th 10/22)
  - Should have preliminary experiments/results for presentation to the class