Electrical Engineering E6761
Computer Communication Networks
Lecture 10
Active Queue Mgmt
Fairness
Inference

Professor Dan Rubenstein
Tues 4:10-6:40, Mudd 1127
Course URL:
http://www.cs.columbia.edu/~danr/EE6761

1

Announcements

- Course Evaluations
  - Please fill out (starting Dec. 1st)
  - Less than 1/3 of you filled out mid-term evals
- Project
  - Report due 12/15, 5pm
  - Also submit supporting work (e.g., simulation code)
  - For groups: include breakdown of who did what
  - It's 50% of your grade, so do a good job!

2

Overview

- Active Queue Management
  - RED, ECN
- Fairness
  - Review TCP-fairness
  - Max-min fairness
  - Proportional Fairness
- Inference
  - Bottleneck bandwidth
  - Multicast Tomography
  - Points of Congestion

3

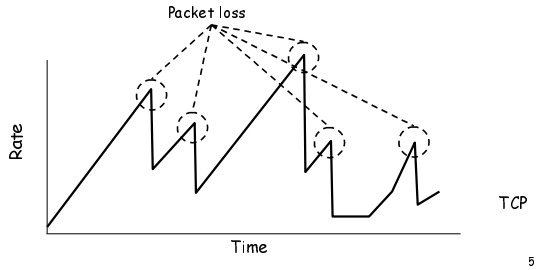
Problems with current routing for TCP

- Current IP routing is
  - non-priority
  - drop-tail
- Benefit of current IP routing infrastructure is its simplicity
- Problems
  - Cannot guarantee delay bounds
  - Cannot guarantee loss rates
  - Cannot guarantee fair allocations
  - Losses occur in bursts (due to drop-tail queues)
- Why is bursty loss a problem for TCP?

4

TCP Synchronization

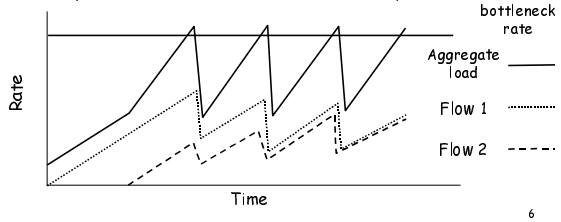
- Like many congestion control protocols, TCP uses packet loss as an indication of congestion



5

TCP Synchronization (cont'd)

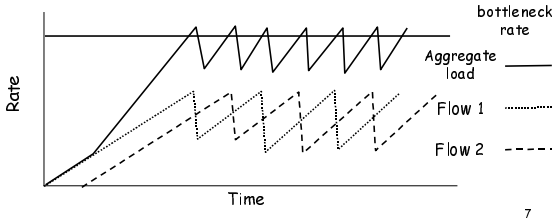
- If losses are synchronized
  - TCP flows sharing bottleneck receive loss indications at around the same time
  - decrease rates at around the same time
  - periods where link bandwidth significantlyunderutilized



6

Stopping Synchronization

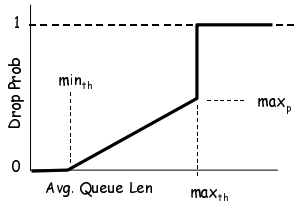
- Observation: if rate synchronization can be prevented, then bandwidth will be used more efficiently
- Q: how can the network prevent rate synchronization?



7

One Solution: RED

- Random Early Detection
  - track length of queue
  - when queue starts to fill up, begin dropping packets randomly
- Randomness breaks the rate synchronization



- min_{th}: lower bound on avg queue length to drop pkts
- max_{th}: upper bound on avg queue length to not drop every pkt
- max_p: the drop probability as avg queue len approaches max_{th}

8

RED: Average Queue Length

- RED uses an average queue length instead of the instantaneous queue length
  - loss rate more stable with time
  - short bursts of traffic (that fill queue for short time) do not affect RED dropping rate
- $avg(t_{i+1}) = (1-w_q)\ avg(t_i) + w_q\ q(t_{i+1})$
  - t_i = time of arrival of ith packet
  - avg(x) = avg queue size at time x
  - q(x) = actual queue size at time x
  - w_q = exponential average weight, $0 < w_q < 1$
- Note: Recent work has demonstrated that the queue size is more stable if the actual queue size is used instead of the average queue size!

9

Marking

- Originally, RED was discussed in the context of dropping packets
  - i.e., when packet is probabilistically selected, it is dropped
  - non-conforming flows have packets dropped as well
- More recently, marking has been considered
  - packets have a special Early Congestion Notification (ECN) bit
  - the ECN bit is initially set to 0 by the sender
  - a "congested" router sets the bit to 1
  - receivers forward ECN bit state back to sender in acknowledgments
  - sender can adjust rate accordingly
  - senders that do not react appropriately to marked packets are called misbehaving

10

Marking v. Dropping

- Idea of marking was around since '88 when Jacobson implemented loss-based congestion control into TCP (see Jain/Ramakrishnan paper)
- Dropping vs. Marking
  - Marking does not penalize misbehaving flows at all (some packets will be dropped in misbehaving flows if dropping is used)
  - With Marking, flows can find steady state fair rate without packet loss (assumes most flows behave)
- Status of Marking:
  - TCP will have an ECN option that enables it to react to marking
  - TCPs that do not implement the option should have their packets dropped rather than marked

11

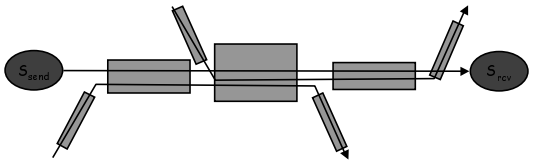
Network Fairness

- Assumption: bandwidth in the network is limited
- Q: What is / are fair ways for sessions to share network bandwidth?
  - TCP fairness: send at the average rate that a TCP flow would send at along same path
  - TCP friendliness: send at an average rate less than what a TCP flow would send at along same path
  - TCP fairness is not really well-defined
    - What timescale is being used?
    - What about for multicast? Which path should be used?
    - Which version of TCP?
  - Other more formal fairness definitions?

12

Max-Min Fairness

- Fluid model of network (links have fixed capacities)
- Idea: every session has equal "right" to bandwidth on any given link
- What does this mean for any session, S?



S can take use as much bandwidth on links as possible
but must leave the same amount for other sessions using the links
unless those other sessions' rates are constrained on other links

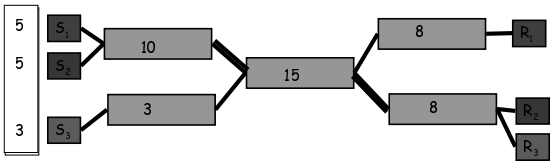
Max-Min Fairness formal def

- Let C_L be the capacity of link L
- Let s(L) be the set of sessions that traverse link L
- Let A be an allocation of rates to sessions
  - Let A(S) be the rate assigned to session S under allocation A
  - A is feasible iff for all L, $\sum_{S \in s(L)} A(S) \leq C_L$
- An allocation, A, is max-min fair if it is feasible and for any other allocation B, for every session S
  - either S is the only session that traverses some link and it uses the link to capacity or
  - if $B(S) > A(S)$, then there is some other session S' where $B(S') < A(S') \leq A(S)$

Max-min fair identification example

- Q: Is a given allocation, A, max-min fair?
- Write the allocation as a vector of session rates, e.g., A = <10,9,4,2,4>
  - session 1 is given a rate of 10 under A
  - session 2 is given a rate of 9 under A
  - there are 5 sessions in the network
- Let B = <10,7,5,3,6> be another feasible allocation
- Then A is not max-min fair
  - $B(S_3) = 5 > 4 = A(S_3)$
  - There is no other session S_i where $B(S_i) < A(S_i) \leq A(S_3)$
    - The only session where $B(S_i) < A(S_i)$ is S_2
    - but $A(S_2) = 9 > A(S_3)$

Max-min fair example



- Intuitive understanding: if A is the max-min fair allocation, then by increasing A(S) by any ε forces some A(S') to decrease where $A(S') \leq A(S)$ to begin with…

Max-Min Fair algorithm

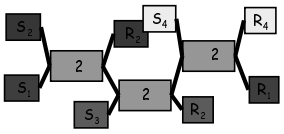
FACT: There is a unique max-min fair allocation!

- Set A(S) = 0 for all S
- Let T = {S: $\sum_{S' \in s(L)} A(S') \le C_L$ for all L where S ∈ s(L)}

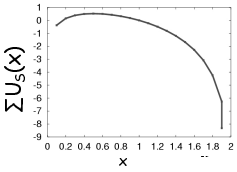
3. If T = {} then end
4. Find the largest δ where for all L, $\sum_{S' \in s(L)} A(S') + \delta I_{S' \in T} \le C_L$
5. For all S ∈ T, A(S) += δ
6. Go to step 2

Proportional Fairness

- Each session S has a utility function, $U_S()$, that is increasing, concave, and continuous
  - e.g., $U_S(x) = \log x$, $U_S(x) = 1 - 1/x$
- The proportional fair allocation is the set of rates that maximizes $\sum U_S(x)$ without links used beyond capacity

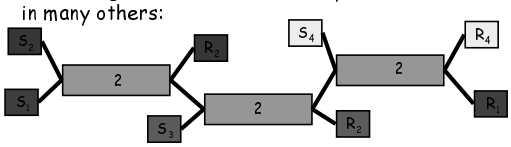


$U_S(x) = \log x$ for all sessions:



Problems with max-min fairness

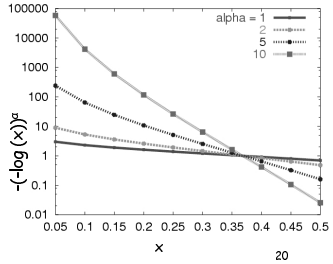
- Does not account for session utilities
  - one session might need each unit of bandwidth more than the other (e.g., a video session vs. file transfer)
  - easily remedied using utility functions
- Increasing one session's share may force decrease in many others:



- Max-Min fair allocation: all sessions get 1
- By decreasing S_1's share by ε, can increase all other flows' shares by ε

Proportional to Max-Min Fairness

- Proportional Fairness can come close to emulating max-min fairness:
  - Let $U_S(x) = -(-\log(x))^\alpha$
  - As $\alpha \to \infty$, allocation becomes max-min fair
  - utility curve "flattens" faster: benefit of increasing one low bandwidth flow a little bit has more impact on aggregate utility than increasing many high bandwidth flows



Fairness Summary

- TCP fairness
  - formal definition somewhat unclear
  - popular due to the prevlance of TCP within the network
- Max-min fairness
  - gives each session equal access to each link's bandwidth
  - difficult to implement using end-to-end means
  - e.g., requires fair queuing
- Proportional fairness
  - maximize aggregate session utility
  - ongoing work to explore how to implement via end-to-end means with simple marking strategies

Network Inference

- Idea: application performance could be improved given knowledge of internal network characteristics
  - loss rates
  - end-to-end round trip delays
  - bottleneck bandwidths
  - route tomography
  - locations of network congestion
- Problem: the Internet does not provide this information to end-systems explicitly
- Solution: desired characteristics need to be inferred

Some Simple Inferences

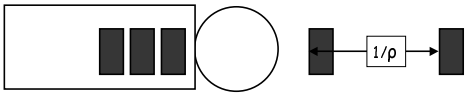
- Some inferences are easy to make
  - loss rate: send N packets, n get lost, loss rate is n/N
  - round trip delay:
    - record packet departure time, T_D
    - have receiving host ACK immediately
    - record packet arrival time, T_A
    - RTT = $T_A - T_D$
- Others need more advanced techniques…

Bottleneck Bandwidth

- A session's bottleneck bandwidth is the minimum rate at which a its packets can be forwarded through the network
- Q: How can we identify bottleneck bandwidth?
  - Idea 1: send packets through at rate, r, and keep increasing r until packets get dropped
  - Problem: other flows may exist in network, congestion may cause packet drops

Probing for bottleneck bandwidth

- Consider time between departures of a non-empty G/D/1/K queue with service rate ρ:

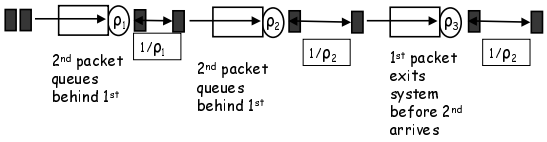


- Observation 1: packet's departure times are spaced by 1/ρ

25

Multi-queue example

- Slower queues will "spread" packets apart
- Subsequent faster queues will not fill up and hence will not affect packet spacing
  - e.g., $\rho_1 > \rho_2$, $\rho_3 > \rho_2$



- NOTE: requires queues downstream of bottleneck to be empty when 1st packet arrives!!!

26

Bprobe: identifying bottleneck bandwidth

- Bprobe is a tool that identifies the bottleneck bandwidth:
- sends ICMP packet pairs
  - packets have same packet size, M
  - depart sender with (almost) 0 time spaced between them
  - arrive back at sender with time T between them
  - Recall T = 1/ρ, where ρ is bottleneck rate
  - Assumes ρ is a linear function of packet size,
    - For a packet of size M, ρ = M · r
    - r = bit-rate bottleneck bandwidth
- Bottleneck bandwidth = r = M / T

27

BProbe Limitations

- BProbe must filter out invalid probes
  - another flow's packet gets between the packet pair
  - a probe packet is lost
  - downstream (higher bandwidth) queues are non-empty when first packet in pair arrives at queue
- Solution:
  - Take many sample packet pairs
  - use different packet sizes
    - No packet in the middle: estimates come out same with different packet sizes
    - Packet in the middle: estimates come out different

28

Different Packet Sizes

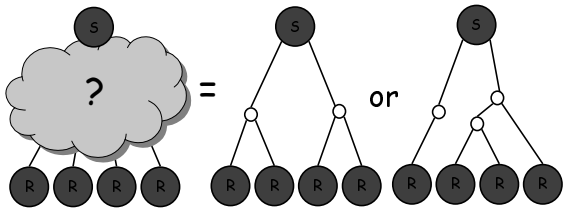
- To identify samples where "background" packet squeezed between the probes
- Let x be the size of the background packet
- Let r be the actual available bandwidth
- Let r_{est} be the estimated available bandwidth
- When background packet gets between probes:
  - $r_{est} = M / (x / r + M / r) = M r / (x + M)$
  - Let r = 5, x = 10
    - M = 5, r_{est} = 5/3
    - M = 10, r_{est} = 5/2

  } different packet sizes yield different estimates!
- Otherwise, $r_{est} = r$: different packet sizes yield same estimate

29

Multicast Tomography

- Given: sender, set of receivers
- Goal: identify multicast tree topology (which routers are used to connect the sender to receivers)



or some other configuration?

30

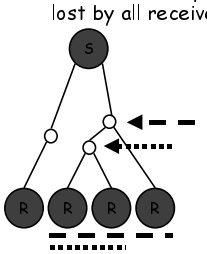
mtraceroute

- One possibility: mtraceroute
  - sends packets with various TTLs
  - routers that find expired TTL send ICMP message indicating transmission failure
  - used to identify routers along path
- Problem with mtraceroute
  - requires assistance of routers in network
  - not all routers necessarily respond

31

Inference on packet loss

- Observation: a packet lost by a shared router is lost by all receivers downstream

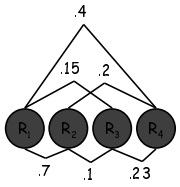


- Idea: receivers that lose same packet likely to have a router in common
- Q: why does losing the same packet not guarantee having router in common?

32

Mcast Tomography Steps

- 4 step process
  - Step 1: multicast packets and record which receivers lose each packet
  - Step 2: Form groups where each group initially contains one receiver
  - Step 3: Pick the 2 groups that have the highest correlation in loss and merge them together into a single group
  - Step 4: If more than one group remains, go to Step 3

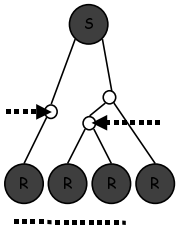


loss correlation graph

33

Ruling out coincident losses

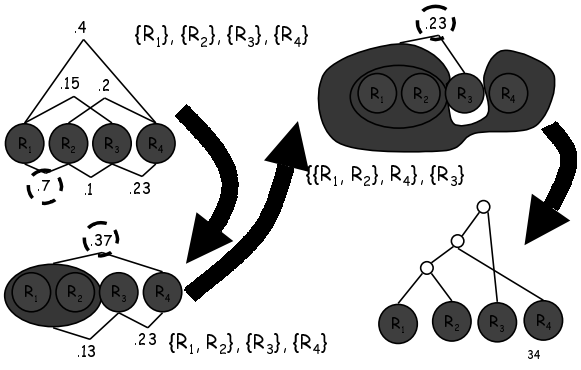
- Losses in 2 places at once may make it look like receivers lost packet under same router



- Q: can end-systems distinguish between these occurrences?
- Assumption: losses at different routers are independent

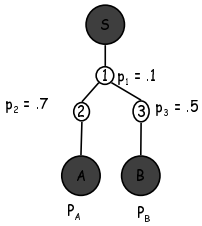
35

Tomography Grouping Example



34

Example

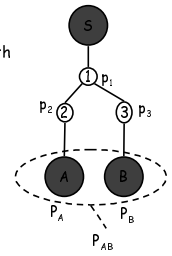


- Actual shared loss rate is .1, but the likelihood that both packets are lost is $p_1 + (1-p_1)\, p_2\, p_3 = .415$

36

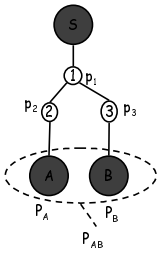
A simple multicast topology model

- A sender and 2 receivers, A & B
  - packets lost at router 1 are lost by both receivers
  - packets lost at router 2 are lost by A
  - packets lost at router 3 are lost by B
- Packets dropped at router i with probability p_i
- Receivers compute
  - P_{AB}: P(both receivers lose the packet)
  - P_A: P(just rcvr A loses the packet)
  - P_B: P(just rcvr B loses the packet)
- To solve: Given topology, P_{AB}, P_A, P_B, compute p_1, p_2, p_3



Solving for p_1, p_2, p_3

- $P_{AB} = p_1 + (1-p_1)\, p_2\, p_3$
- $P_A = (1-p_1)\, p_2\, (1-p_3)$
- $P_B = (1-p_1)(1-p_2)\, p_3$
- Let $X_A = 1 - P_{AB} - P_A = (1-p_1)(1-p_2)$
- Let $X_B = 1 - P_{AB} - P_A = (1-p_1)(1-p_3)$
- X_i = P(packet reaches i)
- $p_2 = P_B / X_A$
- $p_3 = P_A / X_B$
- $p_1 = 1 - P_A / (p_2\,(1-p_3))$

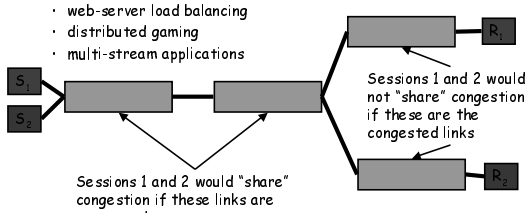


Multicast Tomography: wrapup

- Approach shown here builds binary trees (router has at most 2 children)
  - In practice, router may have more than 2 children
  - Research has looked at when to merge new group into previous parent router vs. creating a new parent
- Comments on resulting tree
  - represents virtual routing topology
  - only routers with significant loss rates are identified
  - routers that have one outgoing interface will not be identifed
  - routers themselves not identified

Shared Points of Congestion (SPOCs)

- When sessions share a point of congestion (POC)
  - can design congestion control protocols that operate on the aggregate flow
  - the newly proposed congestion manager takes this approach
  - Other apps:
    - web-server load balancing
    - distributed gaming
    - multi-stream applications



Detecting Shared POCs

Q: Can we identify whether two flows share the same Point of Congestion (POC)?

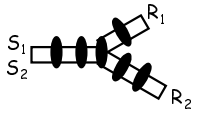
Network Assumptions:

- routers use FIFO forwarding
- The two flows' POCs are either all shared or all separate

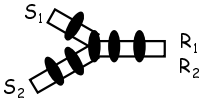
Techniques for detecting shared POCs

- Requirement: flows' senders or receivers are co-located

co-located senders



co-located receivers

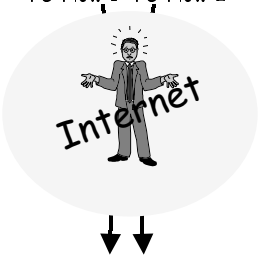


- Packet ordering through a potential SPOC same as that at the co-located end-system
- Good SPOC candidates

Simple Queueing Models of POCs for two flows

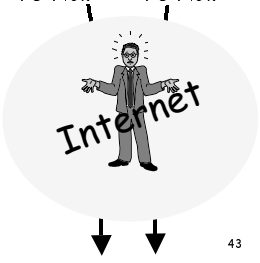
A Shared POC

FG Flow 1 FG Flow 2



Separate POCs

FG Flow 1 FG Flow 2



Approach (High level)

- Idea: Packets passing through same POC close in time experience loss and delay correlations
- Using either loss or delay statistics, compute two measures of correlation:
  - M_c: cross-measure (correlation between flows)
  - M_a: auto-measure (correlation within a flow)
- such that
  - if $M_c < M_a$ then infer POCs are separate
  - else $M_c > M_a$ and infer POCs are shared

The Correlation Statistics...

Loss-Corr for co-located senders:

M_c = Pr(Lost(**i**) | Lost(**i-1**))

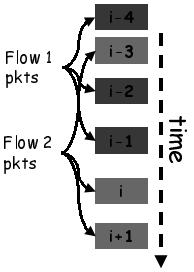
M_a = Pr(Lost(**i**) | Lost(prev(**i**)))

Loss-Corr for co-located receivers:
in paper (complicated)

Delay: Either co-located topology:

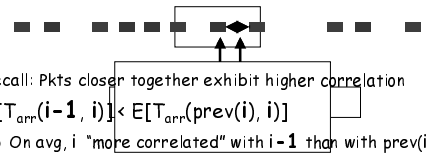
M_c = C(Delay(i), Delay(**i-1**))

M_a = C(Delay(i), Delay(prev(i))



$$C(X,Y) = \frac{E[XY] - E[X]E[Y]}{\sqrt{(E[X^2] - E^2[X])(E[Y^2] - E^2[Y])}}$$

Intuition: Why the comparison works



- Recall: Pkts closer together exhibit higher correlation
- $E[T_{arr}(\mathbf{i-1}, \mathbf{i})] < E[T_{arr}(prev(\mathbf{i}), \mathbf{i})]$
  - On avg, i "more correlated" with **i-1** than with prev(i)
  - True for many distributions, e.g.,
    - deterministic, any
    - poisson, poisson

Summary

- Covered today:
  - Active Queue Management
  - Fairness
  - Network Inference
- Next time:
  - network security