Congestion and the Role of Routers

Jeff Chase
Duke University

Overview

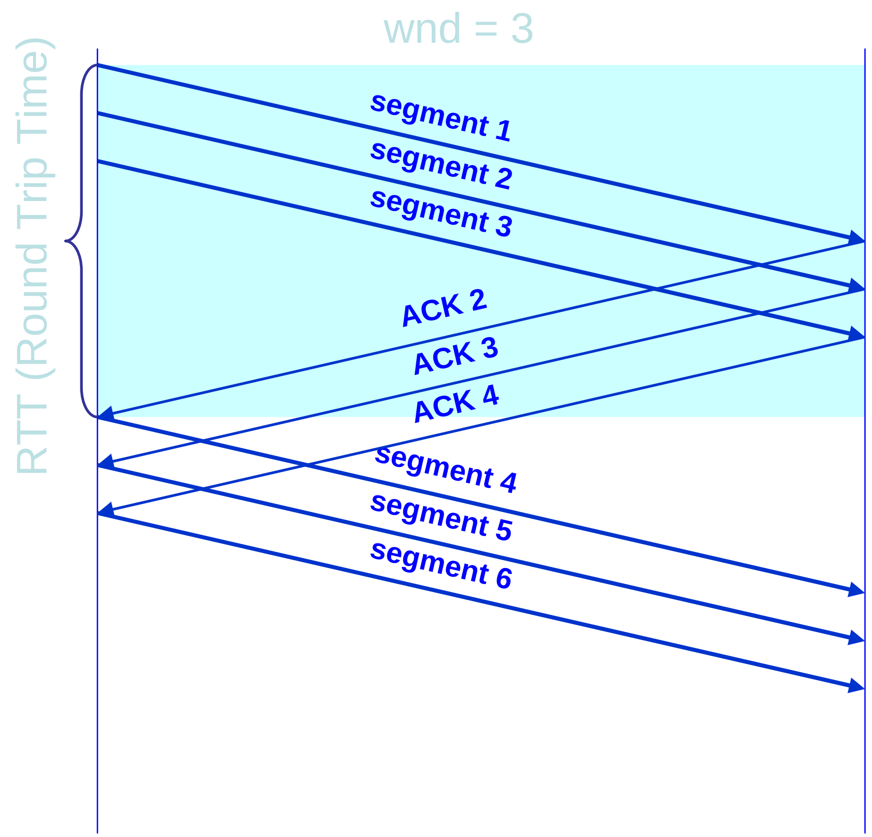
- Problem is "Bullies, Mobs, and Crooks" [Floyd]
- AQM / RED / REM
- ECN
- Robust Congestion Signaling
- XCP
- Pushback

Stoica

- Following slides are from Ion Stoica at Berkeley, with slight mods.

Flow control: Window Size and Throughput

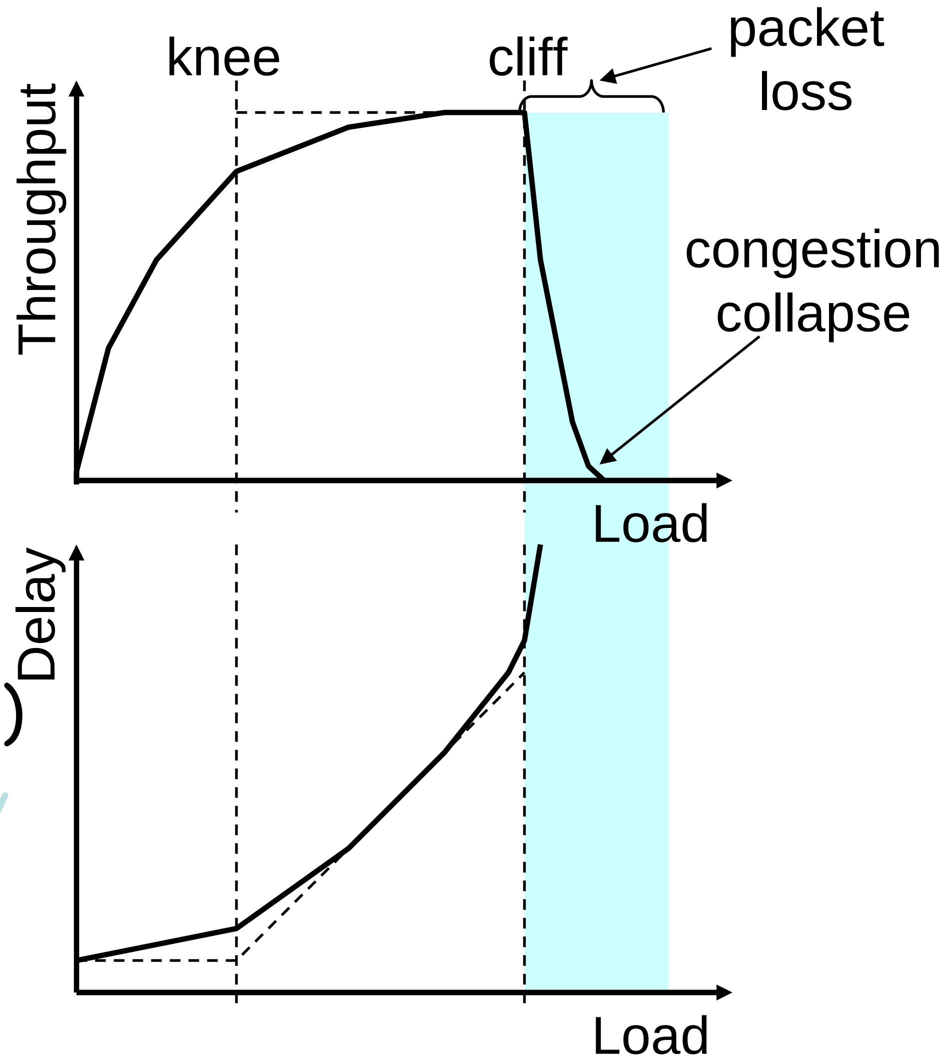
- Sliding-window based flow control:
  - Higher window → higher throughput
    - Throughput = wnd/RTT
  - Need to worry about sequence number wrapping
- Remember: window size control throughput



What's Really Happening?

- Knee – point after which
  - Throughput increases very slow
  - Delay increases fast
- Cliff – point after which
  - Throughput starts to decrease very fast to zero (congestion collapse)
  - Delay approaches infinity

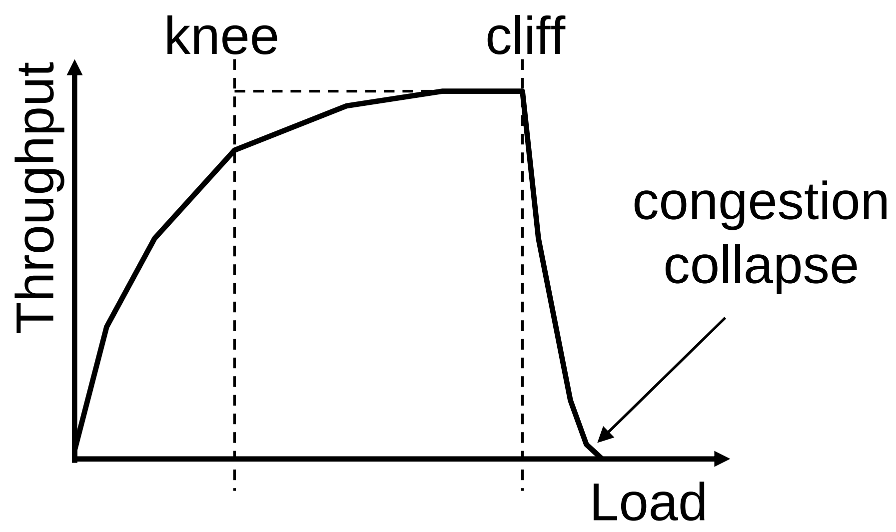
- Note (in an M/M/1 queue)
  - Delay = 1/(1 – utilization)



istoica@cs.berkeley.edu

Congestion Control vs. Congestion Avoidance

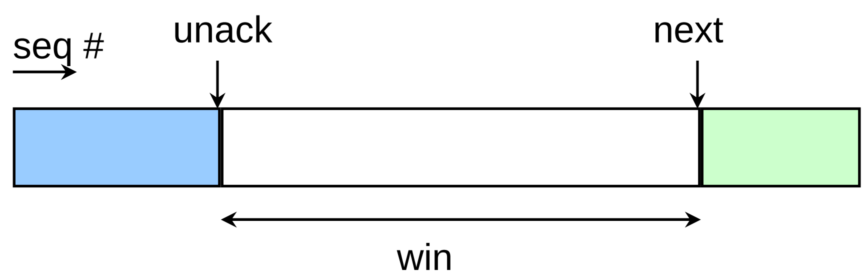
- Congestion control goal
  - Stay left of cliff
- Congestion avoidance goal
  - Stay left of knee



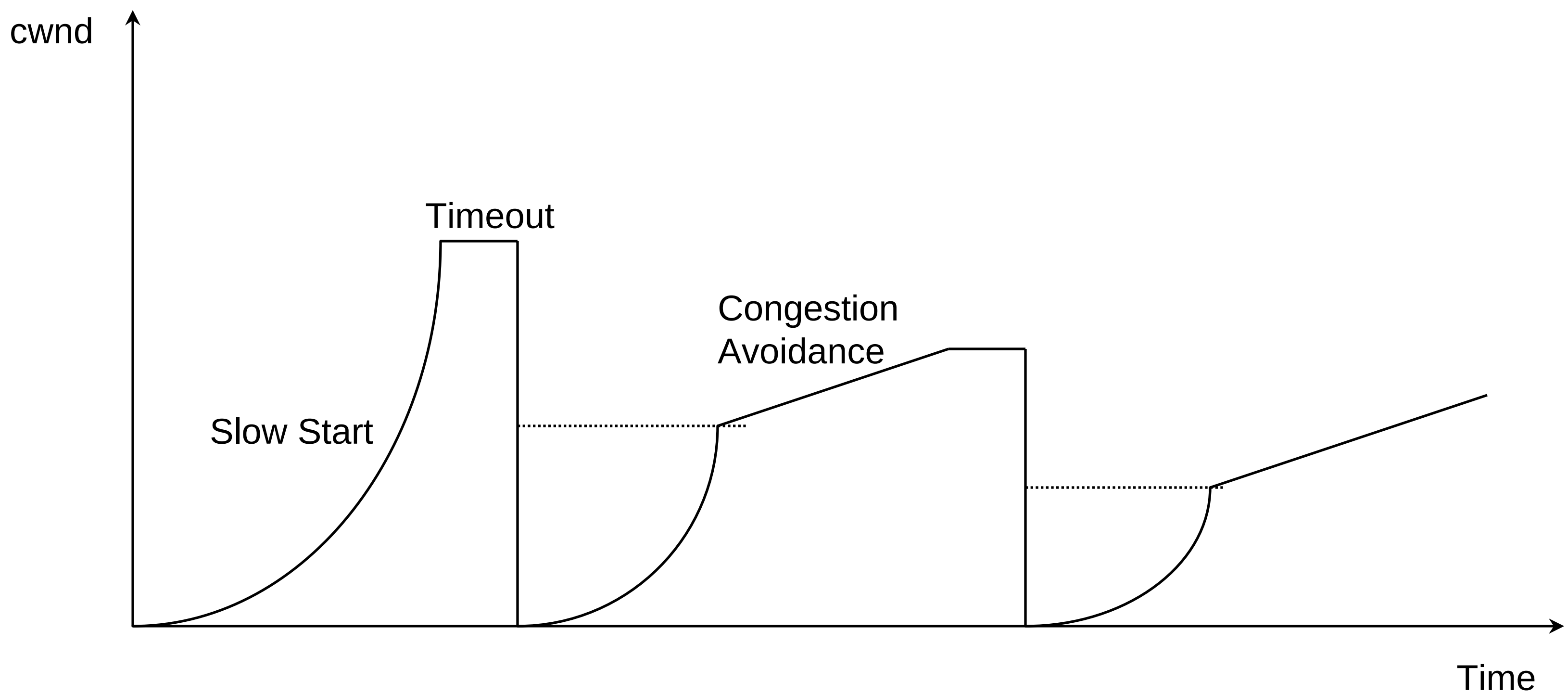
Putting Everything Together: TCP Pseudocode

```
Initially:
    cwnd = 1;
    ssthresh = infinite;
New ack received:
    if (cwnd < ssthresh)
        /* Slow Start*/
        cwnd = cwnd + 1;
    else
        /* Congestion Avoidance
    */
        cwnd = cwnd + 1/cwnd;
Timeout:
    /* Multiplicative decrease */
    ssthresh = cwnd/2;
    cwnd = 1;
```

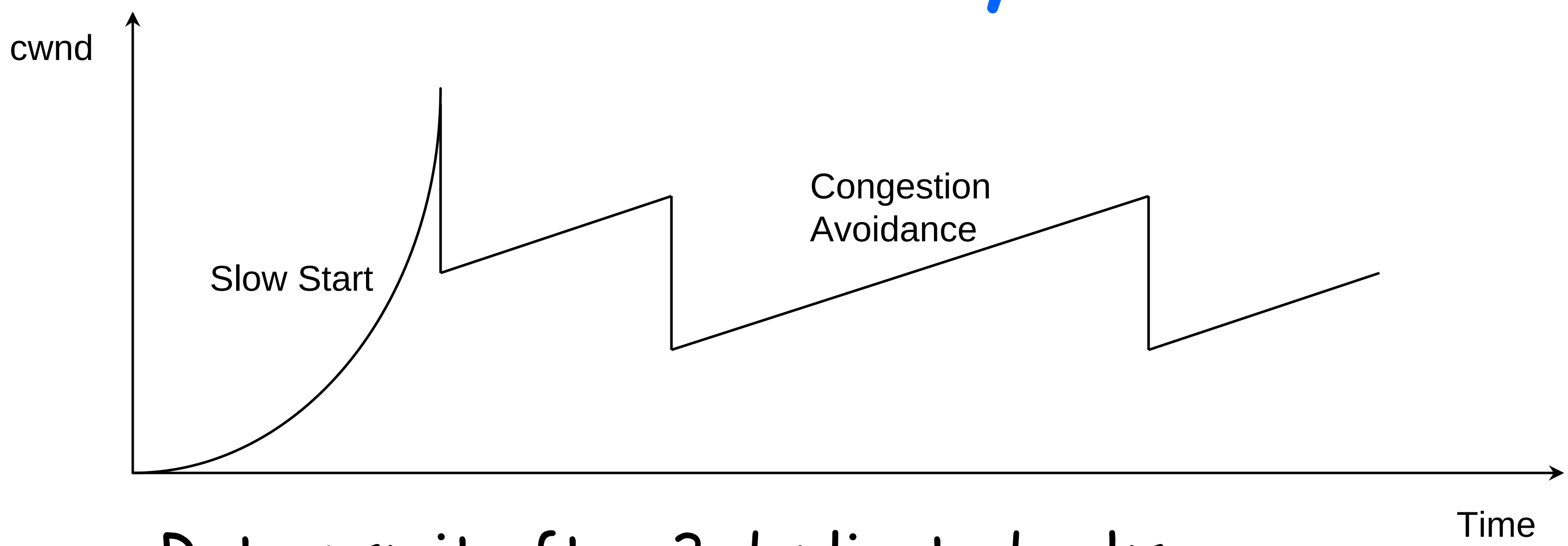
```
while (next < unack + win)
    transmit next packet;

where   win = min(cwnd,
                  flow_win);
```



The big picture

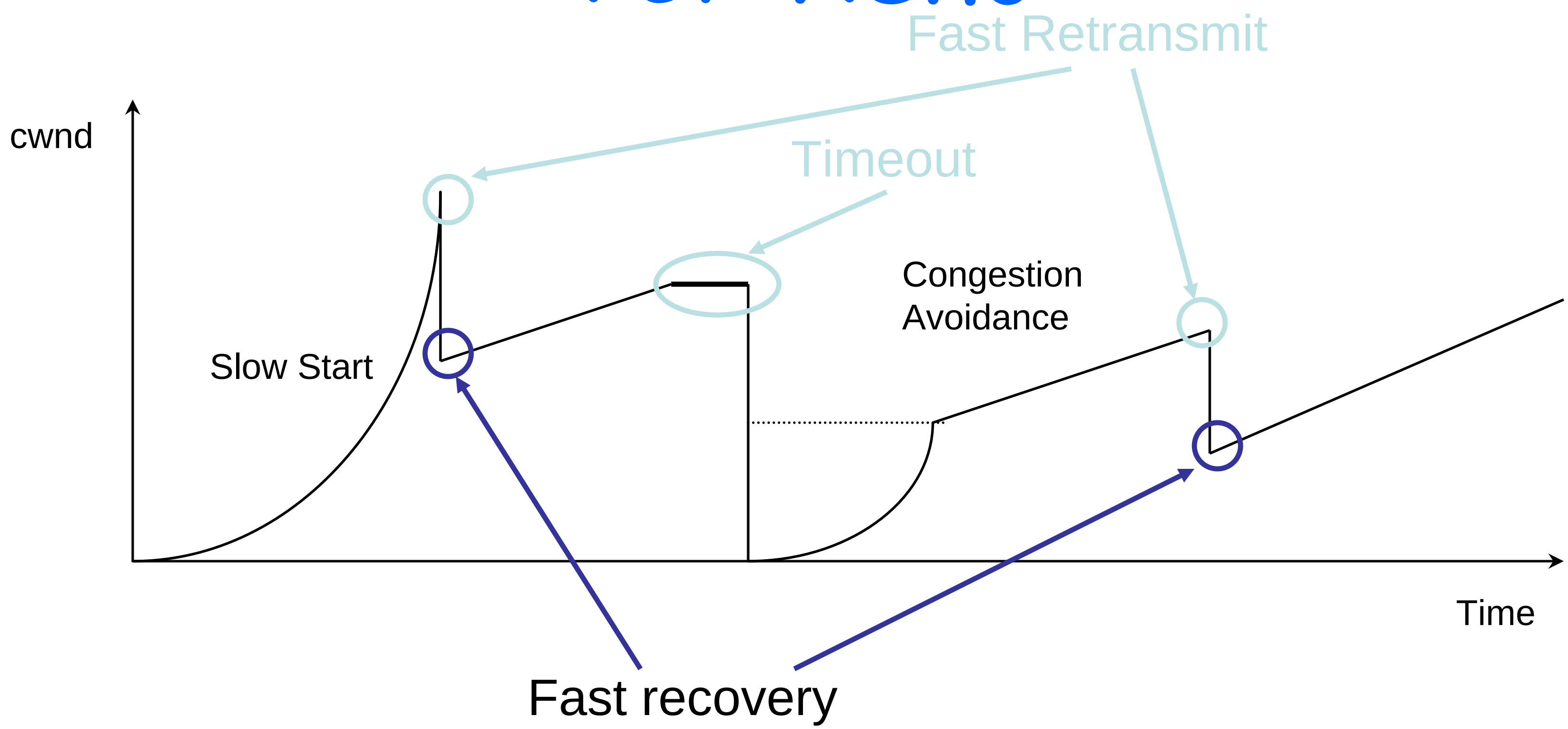


Fast Retransmit and Fast Recovery



- Retransmit after 3 duplicated acks
  - prevent expensive timeouts
- No need to slow start again
- At steady state, *cwnd* oscillates around the optimal window size.

TCP Reno



- Fast retransmit: retransmit a segment after 3 DUP Acks
- Fast recovery: reduce cwnd to half instead of to one

Significance

- Characteristics
  - Converges to efficiency, fairness
  - Easily deployable
  - Fully distributed
  - No need to know full state of system (e.g. number of users, bandwidth of links) (why good?)
- Theory that enabled the Internet to grow beyond 1989
  - Key milestone in Internet development
  - Fully distributed network architecture requires fully distributed congestion control
  - Basis for TCP

TCP Problems

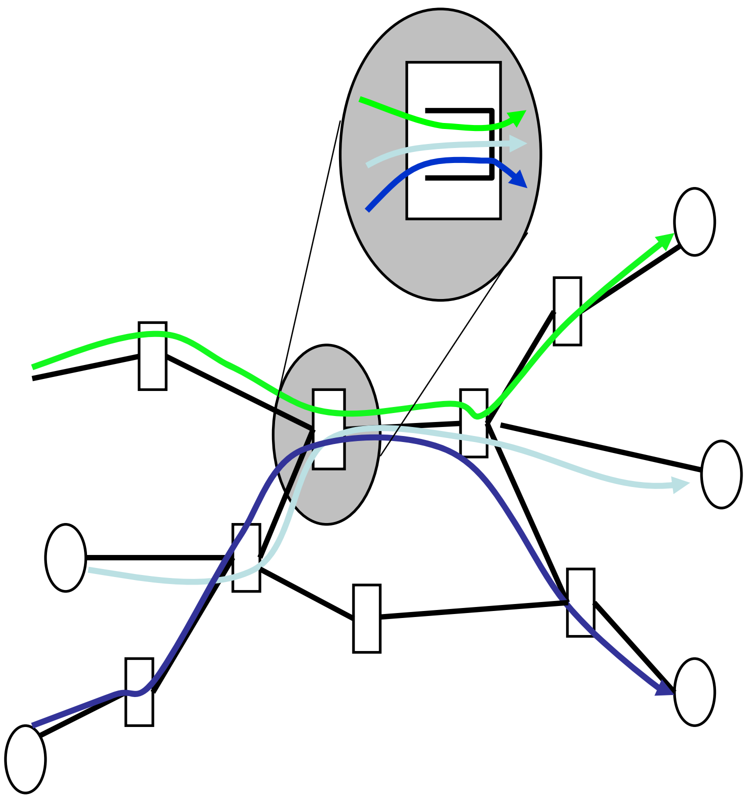
- When TCP congestion control was originally designed in 1988:
  - Key applications: FTP, E-mail
  - Maximum link bandwidth: 10Mb/s
  - Users were mostly from academic and government organizations (i.e., well-behaved)
  - Almost all links were wired (i.e., negligible error rate)
- Thus, current problems with TCP:
  - High bandwidth-delay product paths
  - Selfish users
  - Wireless (or any high error links)

Reflections on TCP

- Assumes that all sources cooperate
- Assumes that congestion occurs on time scales greater than 1 RTT
- Only useful for reliable, in order delivery, non-real time applications
- Vulnerable to non-congestion related loss (e.g. wireless)
- Can be unfair to long RTT flows

Router Support For Congestion Management

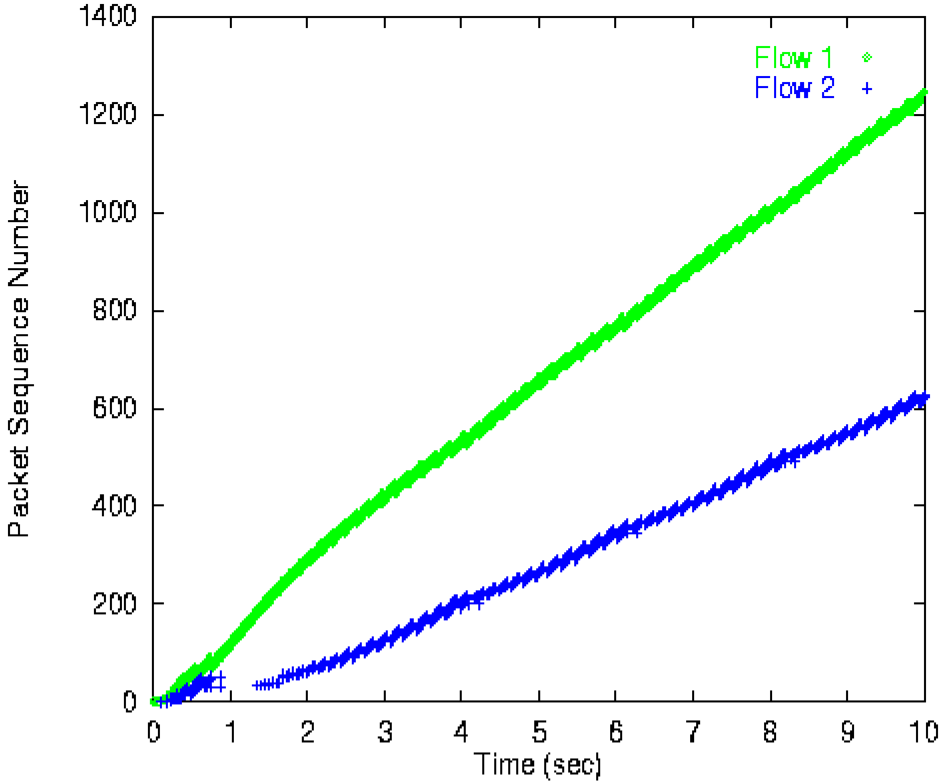
- Traditional Internet
  - Congestion control mechanisms at end-systems, mainly implemented in TCP
  - Routers play little role
- Router mechanisms affecting congestion management
  - Scheduling
  - Buffer management
- Traditional routers
  - FIFO
  - Tail drop

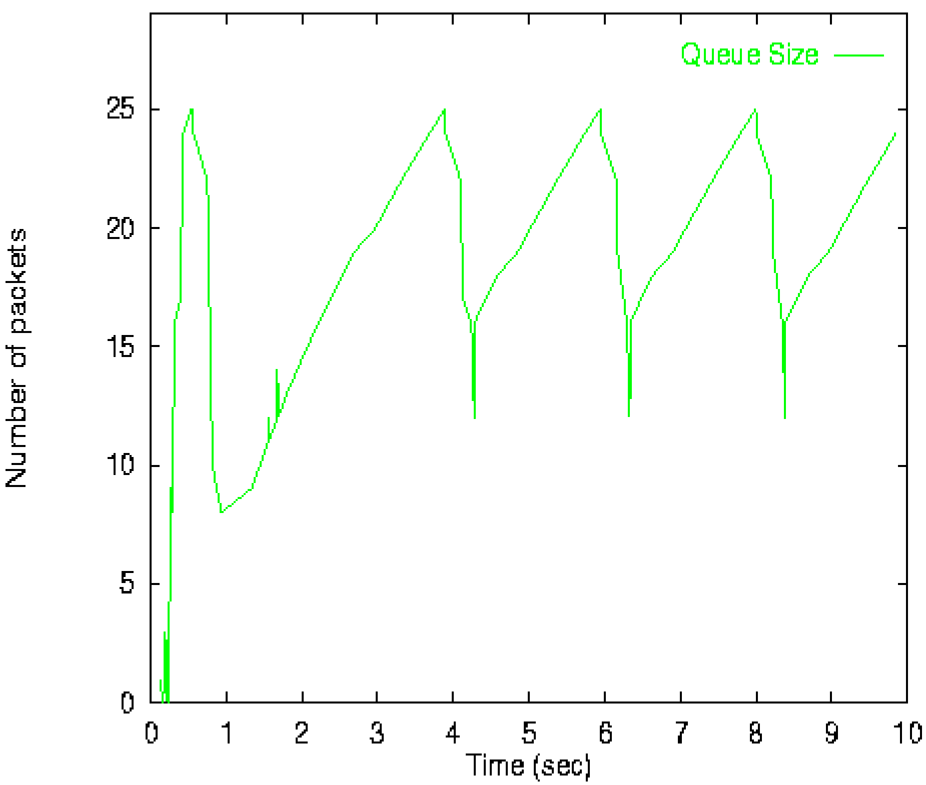


Drawbacks of FIFO with Tail-drop

- Buffer lock out by misbehaving flows
- Synchronizing effect for multiple TCP flows
- Burst or multiple consecutive packet drops
  - Bad for TCP fast recovery
- Low-bandwidth, bursty flows suffer

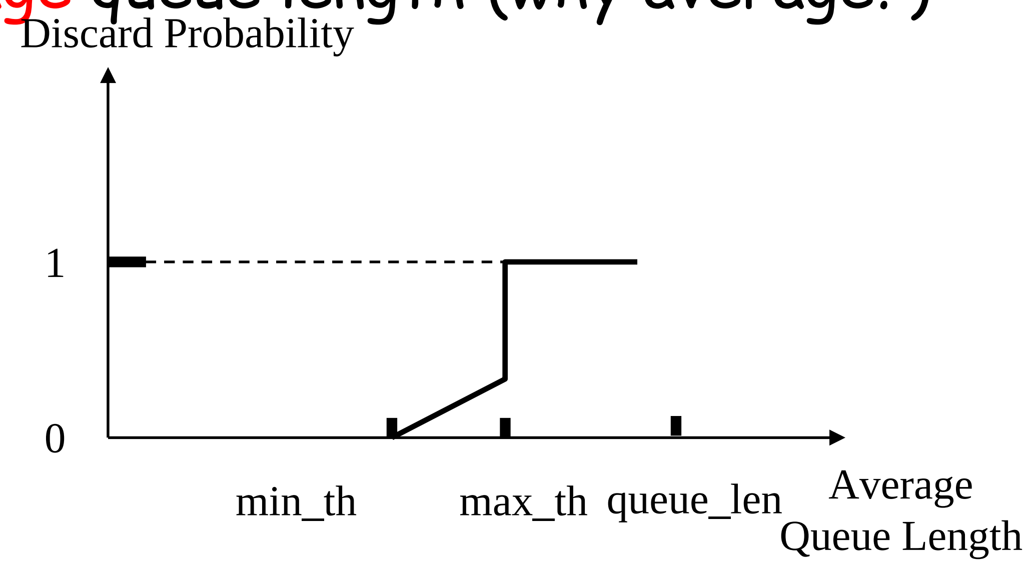
FIFO Router with Two TCP Sessions





RED

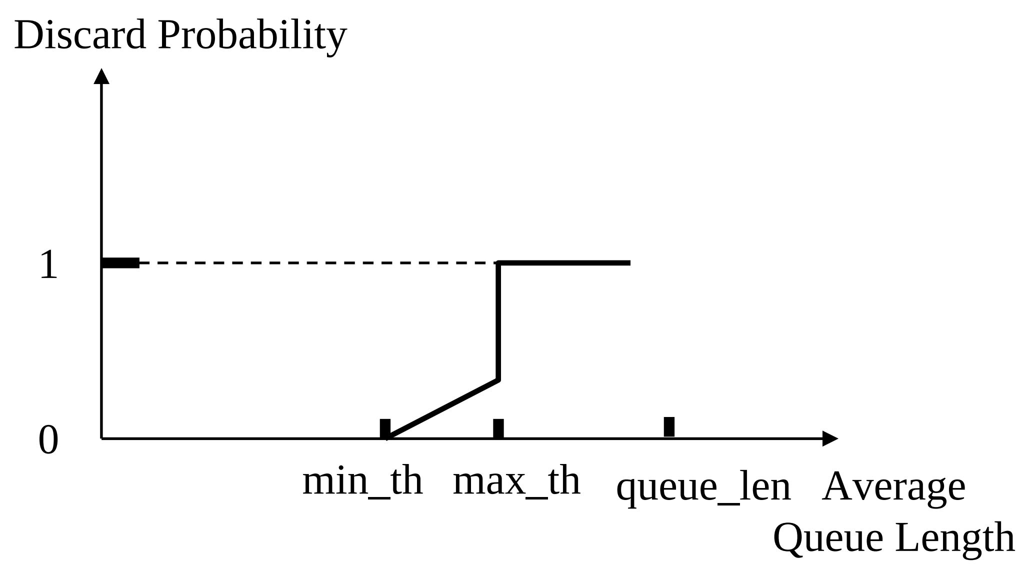
- FIFO scheduling
- Buffer management:
  - Probabilistically discard packets
  - Probability is computed as a function of average queue length (why average?)



istoica@cs.berkeley.edu

RED (cont'd)

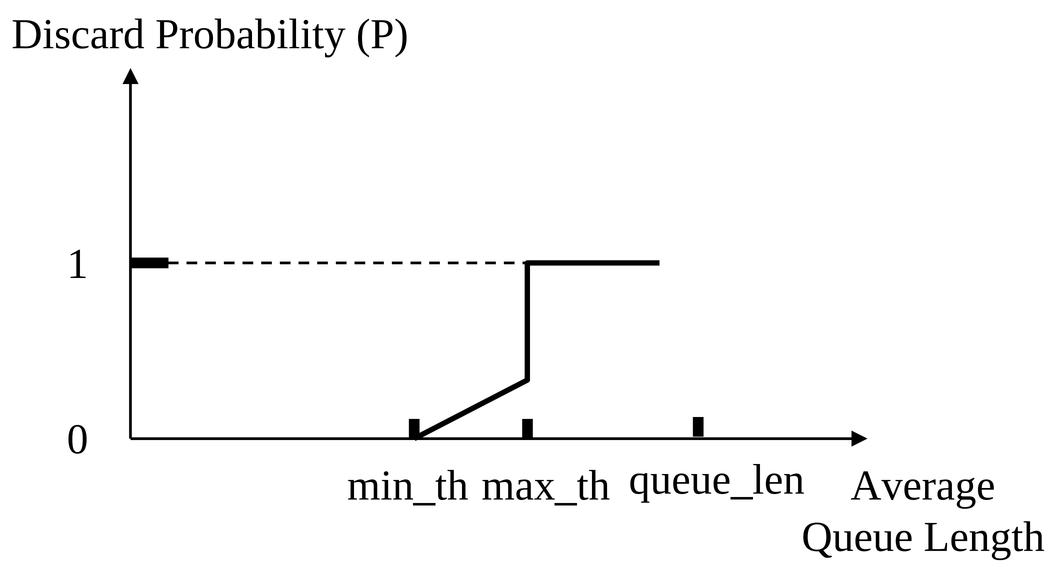
- min_th – minimum threshold
- max_th – maximum threshold
- avg_len – average queue length
  - avg_len = (1-w)*avg_len + w*sample_len



istoica@cs.berkeley.edu

RED (cont'd)

- If (avg_len < min_th) → enqueue packet
- If (avg_len > max_th) → drop packet
- If (avg_len >= min_th and avg_len < max_th) → enqueue packet with probability P



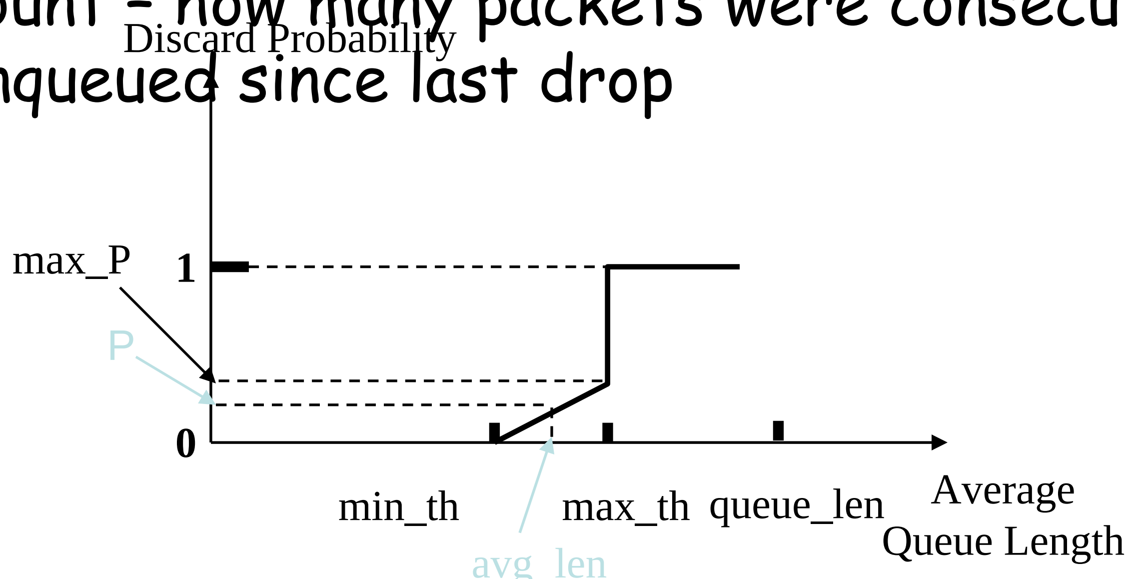
istoica@cs.berkeley.edu

RED (cont'd)

- P = max_P*(avg_len - min_th)/(max_th - min_th)
- Improvements to spread the drops

  P' = P/(1 - count*P), where

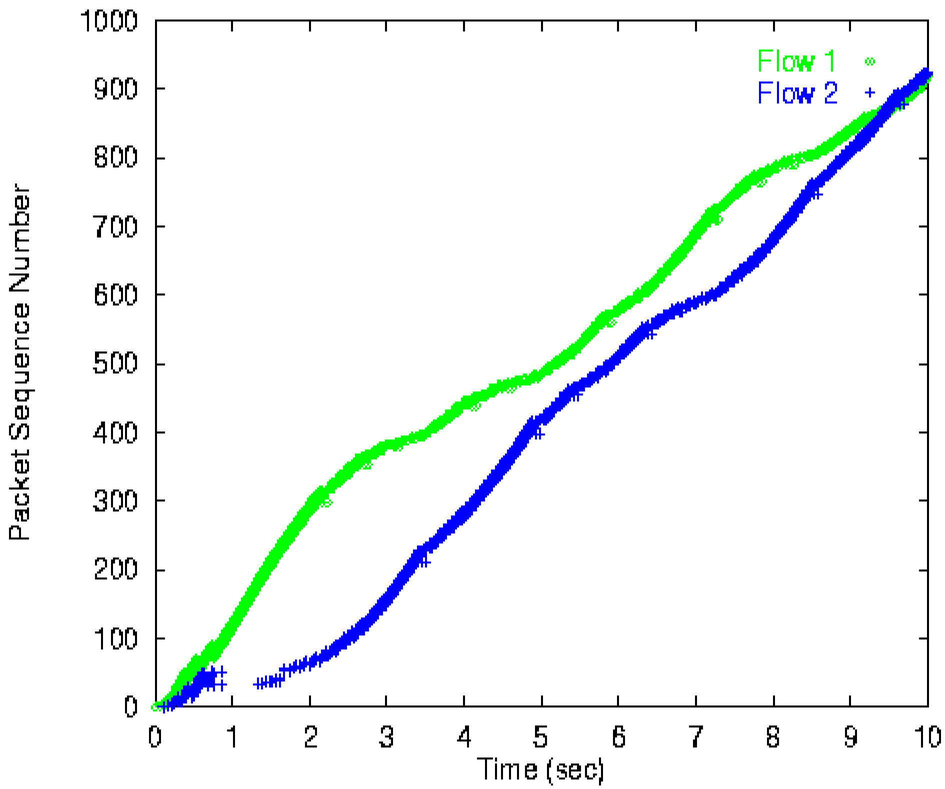
  - count - how many packets were consecutively enqueued since last drop

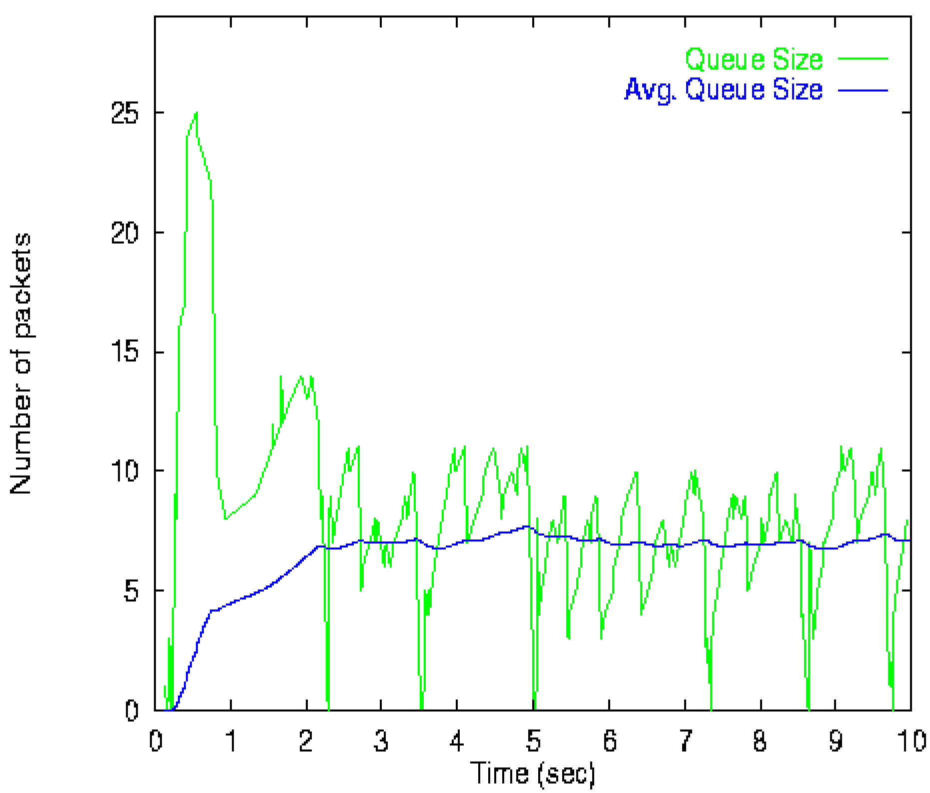


RED Advantages

- Absorb burst better
- Avoids synchronization
- Signal end systems earlier

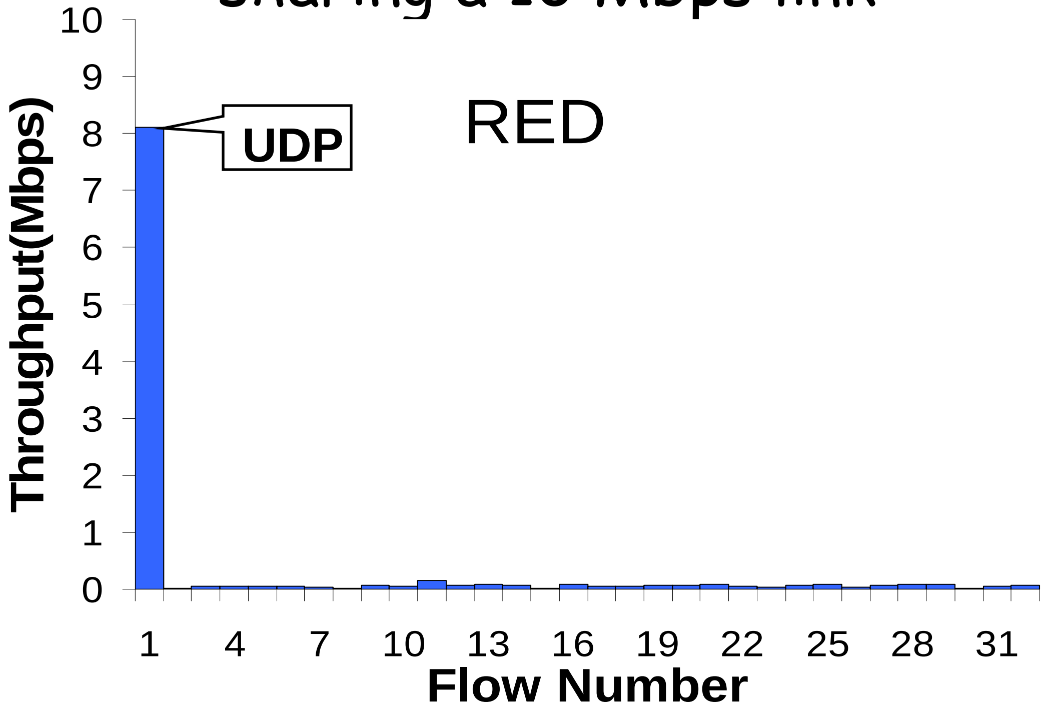
RED Router with Two TCP Sessions





Problems with RED

- No protection: if a flow misbehaves it will hurt the other flows
- Example: 1 UDP (10 Mbps) and 31 TCP's sharing a 10 Mbps link



istoica@cs.berkeley.edu

Promoting…

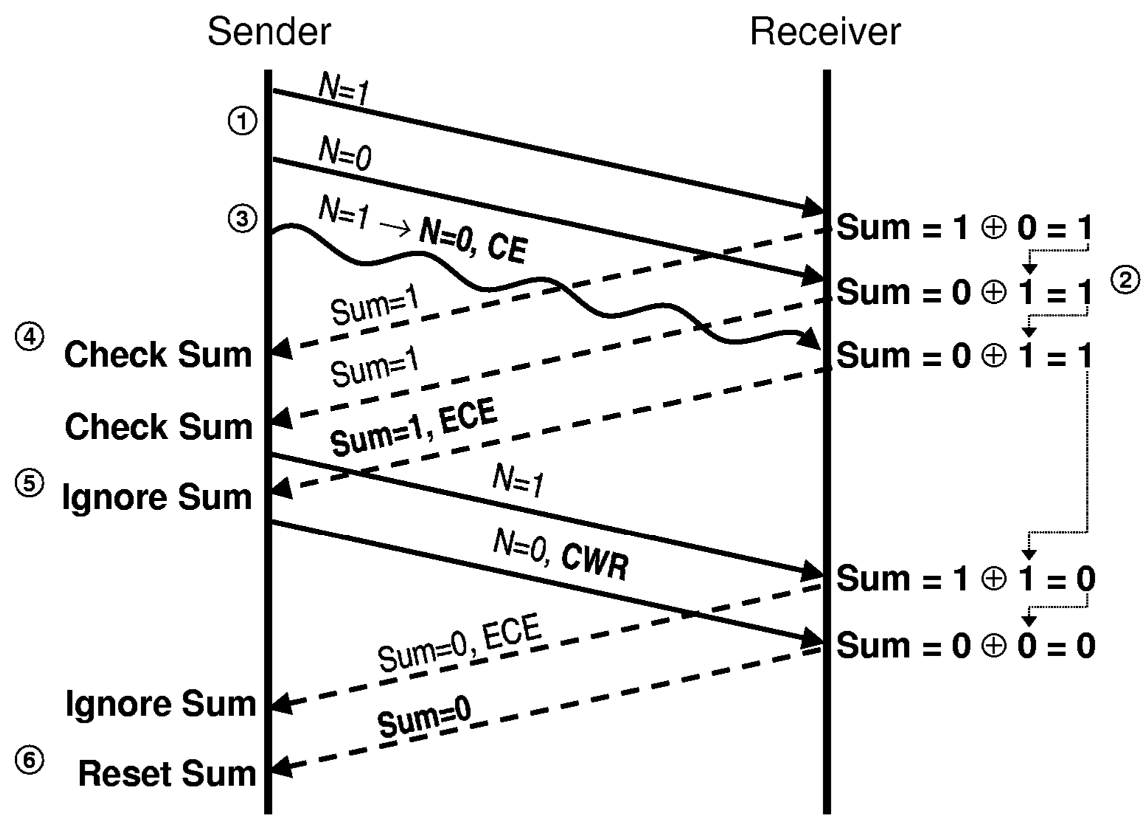
- Floyd and Fall propose that routers preferentially drop packets from unresponsive flows.

ECN

- Explicit Congestion Notification
  - Router sets bit for congestion
  - Receiver should copy bit from packet to ack
  - Sender reduces cwnd when it receives ack
- Problem: Receiver can clear ECN bit
  - Or increase XCP feedback
- Solution: Multiple unmarked packet states
  - Sender uses multiple unmarked packet states
  - Router sets ECN mark, clearing original unmarked state
  - Receiver returns packet state in ack

ECN

- Receiver must either return ECN bit or guess nonce
- More nonce bits → less likelihood of cheating
  - 1 bit is sufficient



istoica@cs.berkeley.edu

Selfish Users Summary

- TCP allows selfish users to subvert congestion control
- Adding a nonce solves problem efficiently
  - must modify sender and receiver
- Many other protocols not designed with selfish users in mind, allow selfish users to lower overall system efficiency and/or fairness
  - e.g., BGP

Slides from
srini@cs.cmu.edu

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!

TCP Performance

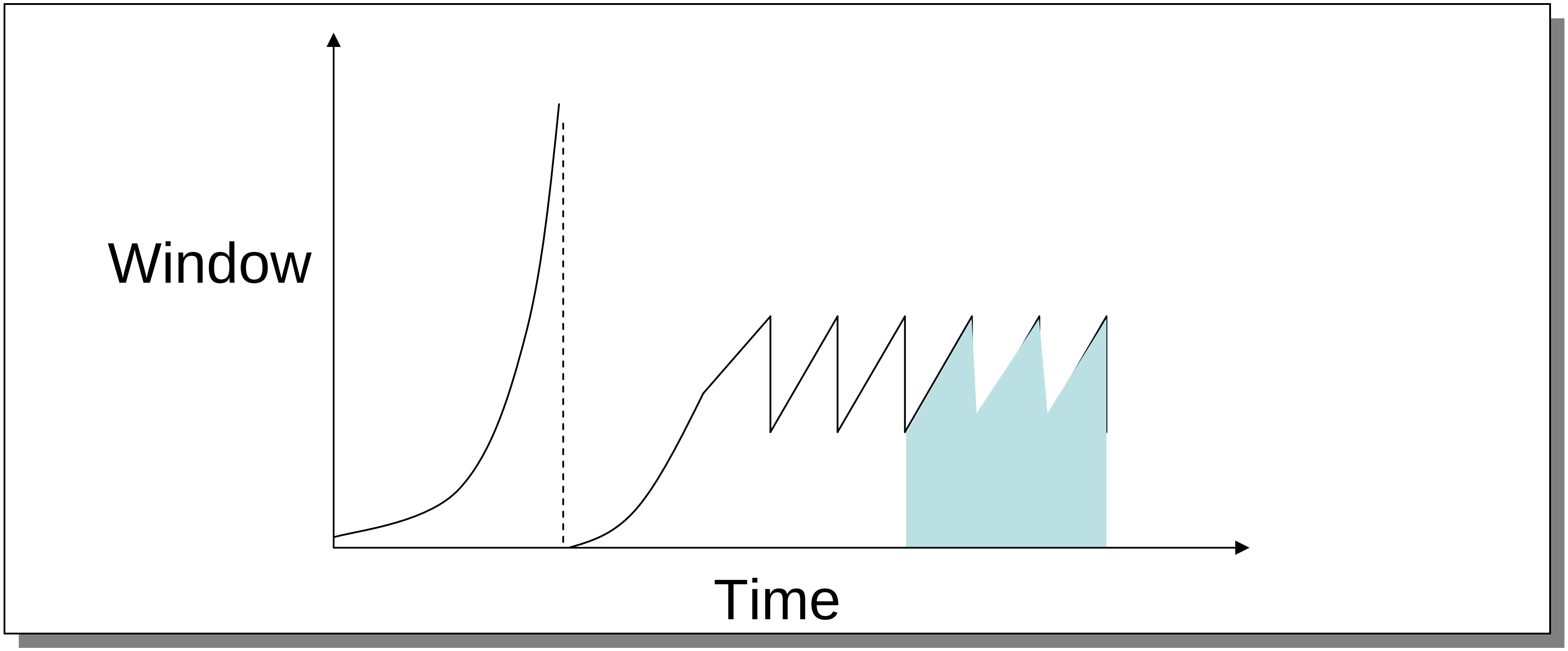
- If we have a large router queue → can get 100% utilization
  - <u>But, router queues can cause large delays</u>
- How big does the queue need to be?
  - Windows vary from W → W/2
    - Must make sure that link is always full
    - W/2 > RTT * BW
    - W = RTT * BW + Qsize
    - Therefore, Qsize ≈ RTT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

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

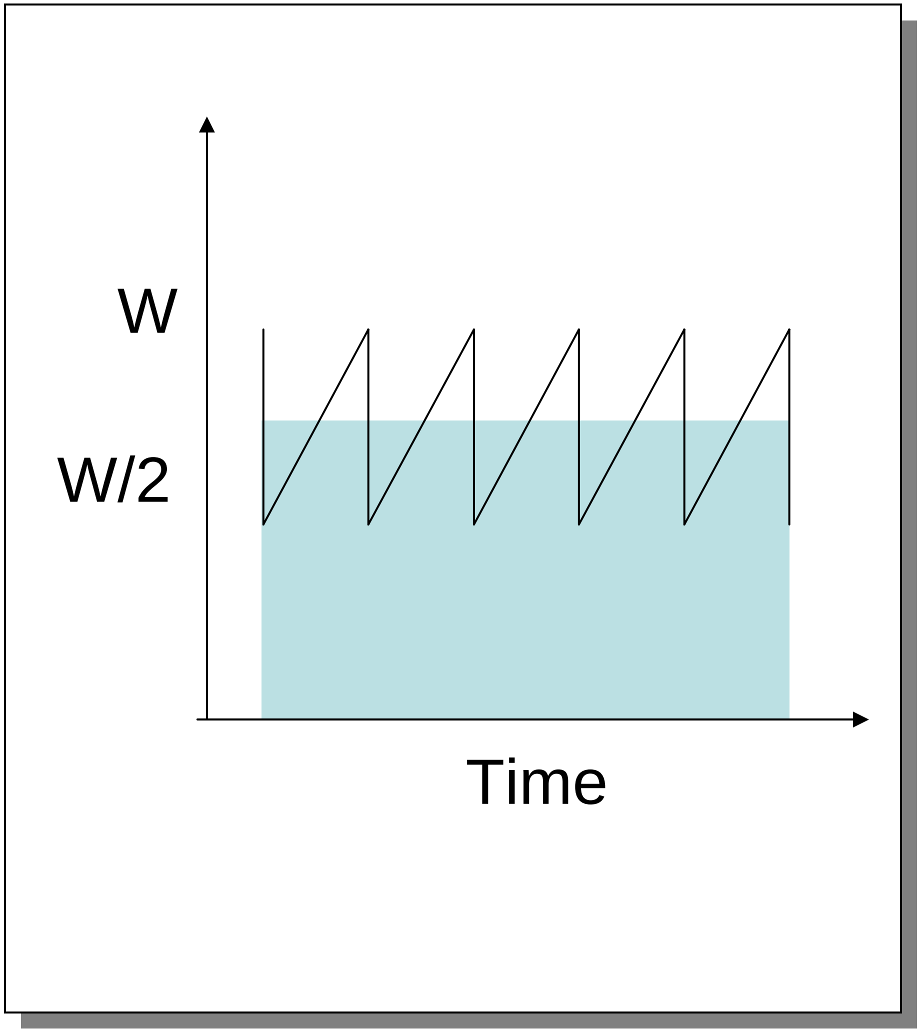
Overall TCP Behavior

- Let's concentrate on steady state behavior with no timeouts and perfect loss recovery
- Packets transferred = area under curve



Transmission Rate

- What is area under curve?
  - W = pkts/RTT, T = RTTs
  - A = avg window * time = ¾ W * T
- What was bandwidth?
  - BW = A / T = ¾ W
    - In packets per RTT
  - Need to convert to bytes per second
  - BW = ¾ W * MSS / RTT
- What is W?
  - Depends on loss rate



Simple TCP Model

Some additional assumptions

- Fixed RTT
- No delayed ACKs
- In steady state, TCP loses 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

Simple Loss Model

- What was the loss rate?
  - Packets per loss $(\frac{3}{4} W/RTT) * (W/2 * RTT) = 3W^2/8$
  - 1 packet lost → loss rate = p = $8/3W^2$
  - $W = \sqrt{\frac{8}{3p}}$
- BW = $\frac{3}{4}$ * W * MSS / RTT
  - $W = \sqrt{\frac{8}{3p}} = \frac{4}{3} \times \sqrt{\frac{3}{2p}}$
  - $BW = \frac{MSS}{RTT \times \sqrt{2p/3}}$

srini@cs.cmu.edu

Fairness

- BW proportional to 1/RTT?
- Do flows sharing a bottleneck get the same bandwidth?
  - NO!
- TCP is RTT fair
  - If flows share a bottleneck and have the same RTTs then they get same bandwidth
  - Otherwise, in inverse proportion to the RTT

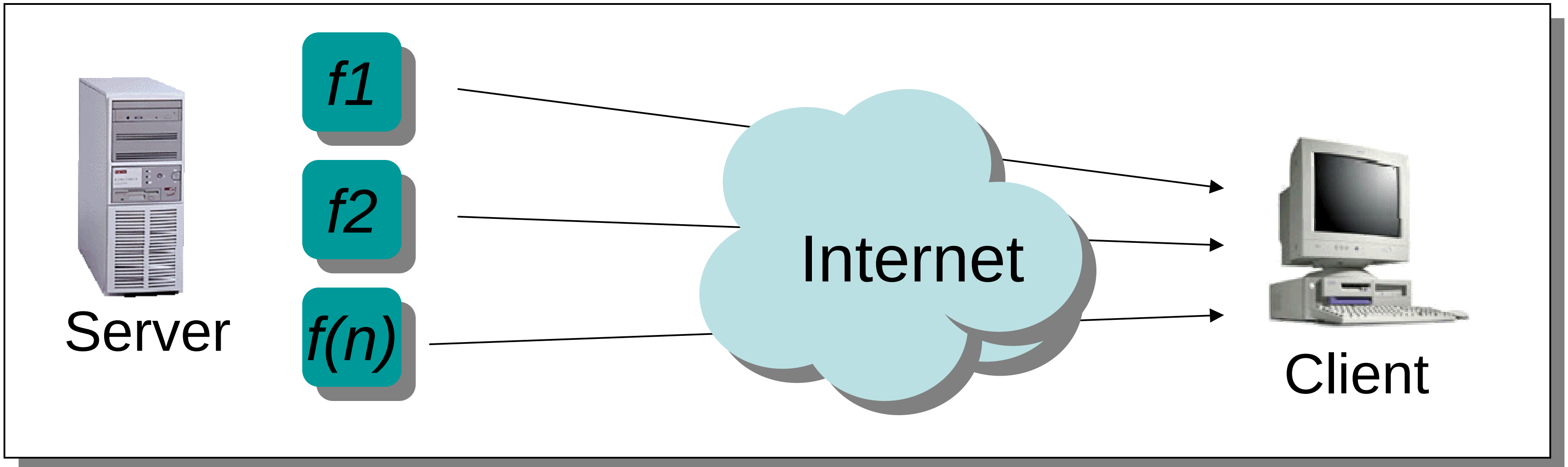
TCP Friendliness

- 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?

Changing Workloads

- New applications are changing the way TCP is used
- 1980's Internet
  - Telnet & FTP → long lived flows
  - Well behaved end hosts
  - Homogenous end host capabilities
  - Simple symmetric routing
- 2000's Internet
  - Web & more Web → large number of short xfers
  - Wild west – everyone is playing games to get bandwidth
  - Cell phones and toasters on the Internet
  - Policy routing

Problems with Short Concurrent Flows



- Compete for resources
  - N "slow starts" = aggressive
  - No shared learning = inefficient
- Entire life is in slow start
- Fast retransmission is rare

TCP Fairness Issues

- Multiple TCP flows sharing the same bottleneck link do not necessarily get the same bandwidth.
  - Factors such as roundtrip time, small differences in timeouts, and start time, … affect how bandwidth is shared
  - The bandwidth ratio typically does stabilize
- Modifying the congestion control implementation changes the aggressiveness of TCP and will change how much bandwidth a source gets.
  - Affects "fairness" relative to other flows
  - Changing timeouts, dropping or adding features, ..
- Users can grab more bandwidth by using parallel flows.
  - Each flow gets a share of the bandwidth to the user gets more bandwidth than users who use only a single flow

(End of borrowed slides.)

XCP

- TCP is unfair (bandwidth proportional to 1/RTT).
- TCP is unstable (depends on # of flows and RTT).
- TCP is inefficient (takes too long to grab the window)
- All exacerbated by "long" and/or "fat" networks.
- Solution:
  - Change all the routers.
  - Generalize ECN.
  - Separate efficiency (MIMD) and fairness (AIMD) controllers.
- Slides by Dina Katabi, SIGCOMM 2002.

ACC and Pushback: Background

- Router can use inverse square-root law to identify nonresponsive flows, or other means to identify high-bandwidth flows (bullies).
- Drop preferentially at congested router.
  - Floyd and Fall, Promoting…
  - Mahajan and Floyd, RED-PD.
- What about aggregate flows from many sources?
  - Mobs: flash crowds
  - Crooks or vandals/terrorists (DDOS)
- "Bullies, Mobs, and Crooks" talk by Sally Floyd
  - (on pushback web page)
- Controlling High-Bandwidth Aggregates in the Network

ACC and Pushback: Issues

- Am I in trouble?
- Whose fault is it?
- Should I punish (throttle) them?
- If so, how much?
- Should I ask somebody else to throttle them for me?
- When should I stop?

ACC and Pushback: Trigger

- Am I in trouble? Monitor packet drops.
- Whose fault is it?
  - Examine packets dropped by AQM/RED.
  - Identify congestion signature: dest prefix.
  - Fair?
  - Per-flow state?

ACC and Pushback: Action

- Should I punish (throttle) the aggregate?
  - Yes.
- If so, how much?
  - Just enough to ensure reasonable service for others. Nothing “Draconian”.
- Should I ask somebody else to throttle them for me?
  - If you can identify substantially contributing upstream routers, ask them for help.
- When should I stop?
  - May need feedback from upstream routers.

When and Who?

- ACC Agent in router maintains rolling drop history.
- Drop above threshold for last K seconds?
- Identify aggregates.
  - Group rates by 24-bit destination prefixes.
  - Merge adjacent prefixes.
  - Narrow to longest common prefix.
- Don't penalize more than some max configured number of aggregates.
- Keep ACC rare.

How and How Much?

- Preferentially drop from aggregates to bring ambient drop rate down to configured threshold.
- Don't drive aggregates below their competitors.
- Identify uniform rate limit L sufficient to distribute all the excess drops among the i aggregates.
  - Fair distribution of pain?
- Apply leaky bucket for aggregates to rate limit L.

Pushback

- If aggregates don't respond (drop rate is high), then ask for help from upstream routers with pushback.
- Identify contributing upstream routers.
- Assess their flow rates.
- Distribute restriction across them in proportion to their flow rates.
- The restriction is a lease (requires maintenance).
- Upstream routers apply restriction only to the traffic that will traverse the congested router.

Discussion

- How does pushback reduce collateral damage?
- Is it enough?
- Could pushback itself be an attack vector?
- What about XCP?
- How could an attacker defeat ACC?
- Trigger time, release time
- Validation methodology: enough?
- Will this stuff ever get deployed? If not, what good is doing the research?