CSE 222
Graduate Networking

Winter 2001

Lecture 6: Routers and Congestion

Geoffrey M. Voelker

Roadmap: The Class So Far

- Internet design and architecture: The network we're studying
  - → Minimal assumptions of network requirements, functionality
- End-to-end: Principles guiding design of Internet arch/protocols
  - → Functionality in end hosts for correctness, network for performance
- Reliable delivery w/ flow control: Essential transport layer
  - → Reliability, flow control done at hosts (TCP, sliding window)
- End-to-end congestion control: In hosts according to E2E
  - → Need congestion control to prevent collapse, also done at hosts
- End-to-end congestion control: Now needed in routers, too
  - → RED: Congestion support in routers improves performance
  - → Router incentives: Congestion support in routers to prevent collapse, i.e., now needed for correctness

Leftover: Short Connections

- How do short connections and Slow-Start interact?
  - What happens when there is a drop in Slow-Start?
  - What happens when the SYN is dropped?
- Bottom line: Not all drops are equal
  - SYN: 6 seconds
  - Slow-Start: No longer probe multiplicatively
  - Congestion avoidance: RTT
- Do you think most flows are short or long?
  - What's the current most popular application?
  - What were the most popular applications when Slow-Start was developed?

RED Congestion Avoidance

- Problem: Want to keep router queue sizes low
  - TCP reacts to drops, needs to fill queues to induce drops
  - Filled queues => large delay in network (bad RTT)
  - With high-speed networks, queues could be huge (bw*delay)
- Ideal
  - Large queues to handle bursts, but average size remains low
- Approach
  - Have routers help prevent end hosts from filling up queues
  - To work with TCP congestion control, need to help TCP anticipate congestion

Router Queueing Disciplines

- Assortment of router queueing disciplines (Ch 6.2)
  - FIFO, FCFS drop tail
    - » When buffer fills up, drop all incoming (easy and simple)
    - » Murder on TCP – drops in bursts, bad news for Slow-Start, Reno
    - » Induces synchronization – packets from all flows get dropped, all flows react to congestion, router/link becomes underutilized, flows overcommit again, repeat wasteful cycle
  - Fair Queueing
    - » Queue for each flow, round robin across queues
    - » => A flow can only fill up its own queue
    - » Spreads out bursts from a single flow (not necessarily good)
    - » Significantly more complex than FIFO drop tail
- Which do you think is most commonly used?

Active Queue Management

- Idea: Drop policy be more congestion-control friendly
  - IP Source Quench
    - » Router sends ICMP packet to host, “hey, partner, slow down”
  - Early Random Drop
    - » When buffer beyond drop level, drop incoming packets according to a drop probability
    - » Does not control misbehaving users
    - » Biases bursty traffic
  - DECbit
    - » Set congestion-indication bit in packets when average queue length is greater than a threshold
    - » Source reduces window when it sees half of packets with bit set

Random Early Detection

- RED
  - Provision routers with large queues to handle short bursts
    - » How large should queue be (general question)?
  - Anticipate congestion before encountering it
    - » Proactive vs. reactive
  - When average queue size is large, randomly mark packets
- Other goals
  - Do not bias against bursts
  - Fairness: packets across flows get marked in proportion to the size of the flows

RED Algorithm

- Router reacts according to average queue size
  - size < min_thresh → do nothing
  - size > max_thresh → mark indiscriminantly
  - min_thresh < size < max_thread → mark with prob p
- What does router need to know?
  - Define min_thresh, max_thresh, w (weighted average)
  - Calculate avg queue size, probability p
  - Want p to vary linearly from min to max
- Collectively known as RED parameters
  - To compare RED behavior, need to indicate RED params

RED Marking

- With RED, you can either mark or drop packets
  - When would it be better to mark instead of drop? Vice versa?
  - Think about assumptions you make about the hosts
- Marking represents another congestion signal to TCP
  - Bit in header: Explicit Congestion Notification (ECN)
    - » Proposed, RFCed, not deployed
  - Forced a drop before queue fills up (c.f., FIFO)
- Randomization: Powerful mechanism
  - Prevents biases against bursts, provides fairness
  - Prevents malicious sender from fooling router
  - Problem: Need good, fast random # generator in router (hard)

RED and the Web

- There was a paper in SIGCOMM that simulated the effect of RED on Web traffic (HTTP 1.0 req/resp)
  [Christiansen et al. 00, UNC]
- The conclusion was that RED had negligible benefit over FIFO queueing
  - Also, at heavy loads, RED parameters that gave the best link utilization induced the poor response times
- What do you think Sally Floyd's response to the paper might have been?
  - Think about the kind of flows used in HTTP 1.0

RED Discussion

- Questions about the paper, RED?
- RED has been implemented, deployed sparingly
  - Why do you think it hasn't taken off?
- What is the most challenging aspect of using RED?

Promoting E2E Congest Ctrl

- Problem: Hark! The sky is falling!
  - Non-congestion-controlled best-effort traffic is going to cause another congestion collapse in the Internet
  - Streaming UDP, TCP "accelerators", etc.
- Congestion collapse
  - 80s: Classical – unnecessary retransmission of packets
  - 00s: Undelivered packets – packets dropped in network before reaching receiver (hence shouldn't have been sent)
  - Other kinds, but not as imminent or potentially widespread

Network Helps Itself

- Implication: The network must now participate in controlling its utilization
  - Does this change the underlying assumptions of the Internet?
  - Does this violate the E2E argument?
- But still need E2E congestion control
  - Network mechanisms assist E2E congestion control
  - Fundamental: Packet switched network => E2E CC

Incentivize Good Behavior

- The approach advocated in this paper is to have the network provide hosts with the incentive to behave
  - If you're a good citizen, no one bothers you
  - If you misbehave, you get your hand slapped
  - => Remove incentive to misbehave
- Mechanisms
  - Per-flow scheduling mechanisms to regulate traffic
    - » Problem: Can still lead to congestion collapse
  - Pricing mechanisms to control sharing (get what you pay for)
    - » Problem: Big ?, also clear you need a lot of state at routers
  - Reinforce end-to-end congestion control at routers

Regulating Flows

- Three approaches for regulating flows at routers
  - Non TCP-Friendly flows
  - Unresponsive flows
  - Flows consuming disproportionate bandwidth
- Only do this for high bandwidth flows
  - Too much processing to do for all flows
  - High bandwidth flows are the ones causing the problem
- Only do this when router senses imminent congestion
  - Still have to do more work at the worst time, though

Aside: TCP Throughput & RTT

- The approaches in this paper examine flow throughput
  - Use flow throughput to decide whether a flow is not TCP-friendly, unresponsive, consumes too much b/w
- Problem: TCP throughput is a function of RTT
  - TCP is clocked by ACKs, e.g., Slow-Start
  - TCP needs to wait RTTs to open congestion window
  - Larger RTT => lower TCP throughput
- Implication: Two TCP flows can have different throughputs even though both are behaving correctly
  - Need to make assumptions about when behavior is not RTT dependent, but actually bad behavior

TCP-Friendly Flows

- Approach: Constrain non TCP-Friendly flows
  - Defn: A TCP-Friendly flow has an arrival rate limited by VJ congestion avoidance (mult. decrease, add. increase)
  - Implies an upper bound on b/w given MTU, RTT, drop rate
- Action: Drop flow's packets to throttle to expected b/w
- Limitations
  - Need to know MTU, RTT, drop rate
  - B/w depends upon RTT, need to use low RTT estimate
  - What if TCP changes?
- Is Vegas TCP-Friendly by this definition?
  - What would happen to Vegas flows?

Unresponsive Flows

- Approach: Constrain flows unresponsive to drops
  - Router drops when there is congestion
  - Cooperative flows will decrease throughput in response
    - » Drop rate grows * x, throughput should decrease $x^{1/2}$
  - If a flow's throughput does not drop, it is unresponsive
- Action: Drop packets for unresponsive flows
  - Throttle flow back to $x^{1/2}$ level (can be rough approx)
- Limitations
  - Flows with variable demands (e.g., short flows)
- Is Vegas unresponsive?
  - What does Vegas do in response to drops?

Bandwidth Hogs

- Approach: Constrain flows using disproportionate amount of bandwidth
  - Disproportionate share: Intuitively, a flow experiencing drops should get no more than 1/n of the bandwidth
  - Relative to level of congest: Drops due to congestion should limit bandwidth of a cooperating flow
- Action: Drop until flow is only using its portion
- Limitations
  - Gauging level of unsatisfied demand
- Is Vegas a bandwidth hog?

Discussion

- Which approach do you think has the best tradeoffs?
- Do you agree with the high-level argument?
  - Routers should provide incentive to do congestion control
- Do you think the solutions are too TCP-centric?
  - Based upon assumptions of TCP congestion control
- Tired of congestion control yet?