Congestion Notification

Bill Cheng

http://merlot.usc.edu/cs551-f12

Congestion Avoidance

- TCP's approach is reactive:
  - Detect congestion after it happens
  - Increase load trying to maximize utilization until loss occurs
  - TCP has a *congestion avoidance phase*, but that's different from what we're talking about here
- Alternatively, we can be proactive:
  - We can try to predict congestion and reduce rate before loss occurs
  - This is called *congestion avoidance*

Router Congestion Notification

- Routers well-positioned to detect congestion
  - Router has unified view of queueing behavior
  - Routers can distinguish between propagation and persistent queueing delays
  - Routers can decide on transient congestion, based on workload
- Hosts themselves are limited in their ability to infer these from perceived behavior

Router Mechanisms

- Congestion notification
  - The DEC-bit scheme
    - explicit congestion feedback to the source
  - Random Early Detection (RED)
    - implicit congestion feedback to the source
    - well suited for TCP

Congestion Avoidance (DEC-bit)

[Ramakrishnan90a]

Bill Cheng

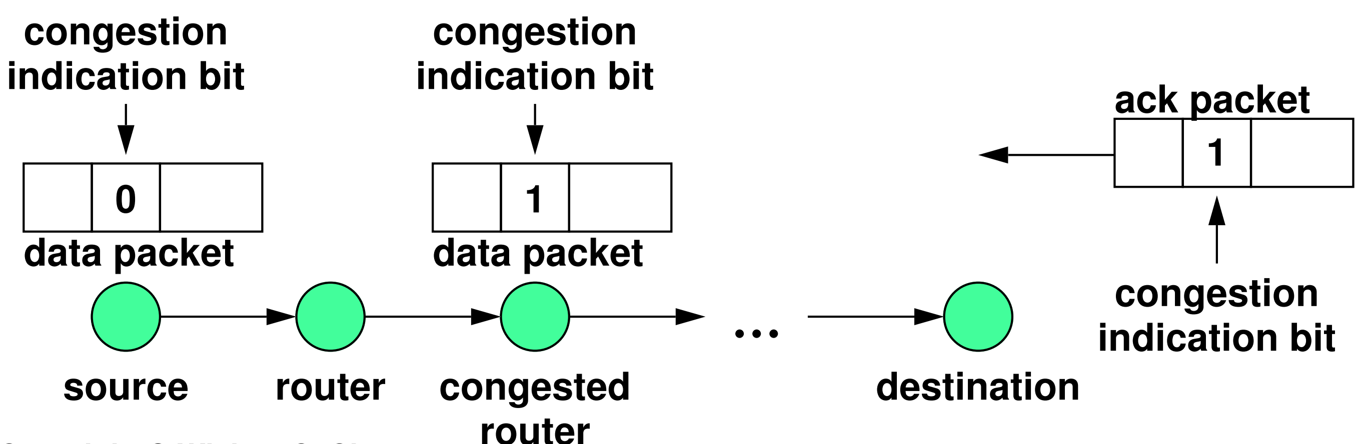
http://merlot.usc.edu/cs551-f12

Key Ideas

- Approach to do congestion avoidance
  - alternative to TCP
- First use of explicit congestion notification (for window-based protocols)
  - uses information from routers, not just end-to-end
- Defines several terms
  - power, efficiency, fairness

The DEC-bit Scheme

- **Basic ideas:**
  - On congestion, router sets a bit (CI) bit on packet
  - Receiver relays bit to sender in acknowledgements
  - Sender uses feedback to adjust sending rate
- **Key design questions:**
  - *Router:* feedback policy (how and when does a router generate feedback)
  - *Source:* signal filtering (how does the sender respond?)



Design Choices for Feedback

- What kind of feedback
  - Separate packets (source quench)
  - Mark packets, receiver propagates marks in ACKs
- When to generate feedback
  - Based on router utilization
    - you can be near 100% utilization without seeing a throughput degradation
  - Queue lengths
    - but what queue lengths (instantaneous, average)?

Options

- Congestion avoidance vs. congestion control
  - which is TCP?
  - which is DEC-bit?
- Feedback mechanisms:
  - packet loss
  - source quench packet
  - CI-bit (or DEC-bit): congestion indication

Components of a Congestion Avoidance System

- Router
  - detection mechanism
  - feedback sending mechanism
- User
  - feedback receiving mechanism
  - decision policy
    - decision frequency?
    - filtering?
  - response

Components of a Congestion Avoidance System *(for DEC-bit)*

- Router
  - detection mechanism *(average queue length)*
  - feedback sending mechanism *(DEC-bit)*
- User
  - feedback receiving mechanism *(DEC-bit in ACK)*
  - decision policy
    - decision frequency? *(2 RTT)*
    - filtering? *(> 50% with DEC-bit set)*
  - response *(AIMD: cwnd=0.875×cwnd)*

Why Queue Lengths?

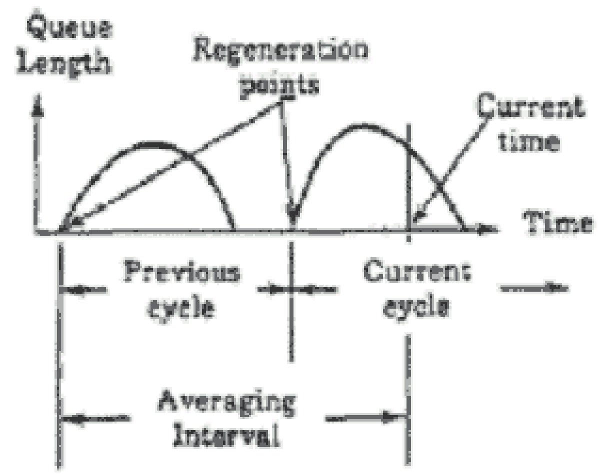
- It is desirable to implement FIFO
  - Fast implementations possible
  - Shares delay among connections
  - Gives low delay during bursts
- FIFO queue length is then a natural choice for detecting the onset of congestion

Measuring Queue Size

- Measuring queue size
  - need to consider average, not instantaneous
    - want to give smoother feedback to the user (want to identify longer term congestion, not just transient bursts)
  - option: exponential weighted moving average (EWMA) of queue size
    - $Q_{avg} = \alpha \, Q_{avg} + (1 - \alpha) \, Q_{inst}$
  - choice: average over *regeneration cycles*
    - average queue length is the area under the curve divided by the total time for the regeneration cycle
    - why not use fixed averaging interval? (perhaps self-tuning)

Computing Average Queue Lengths

- Possibilities:
  - Instantaneous
    - premature, unfair
  - Averaged over a fixed time window, or exponential average
    - can be unfair if time window different from round-trip time
- Solution
  - Adaptive queue length estimation: busy/idle cycles
  - But need to account for long current busy periods



Sender Behavior

- How often should the source change window?
- In response to what received information should it change its window?
- By how much should the source change its window?
  - We already know the answer to this: AIMD
    - DEC-bit scheme uses a multiplicative factor of 0.875

How Often to Change Window?

- Not on every ACK received
  - Window size would oscillate dramatically because it takes time for a window change's effects to be felt
- Correct policy: wait for (W+W') ACKs
  - Where W is window size before update and W' is size after update

Using Received Information

- Use the CI bits from W' acks in order to decide whether congestion still persists
- Clearly, if some fraction of bits are set, then congestion exists
- What fraction?
  - Depends on the policy to set the threshold
  - When queue size threshold is 1, cutoff fraction should be 0.5
  - This has the nice property that the resulting power is relatively insensitive to this choice

Changing the Sender's Window

- Sender policy
  - Monitor packets within a window
  - Make change if more than 50% of packets had CI set:
    - if < 50% had CI set, then increase window by 1
    - else new window = window * 0.875
  - Additive increase, multiplicative decrease for stability

Metrics

- Fairness
- Power
- Efficiency

Fairness

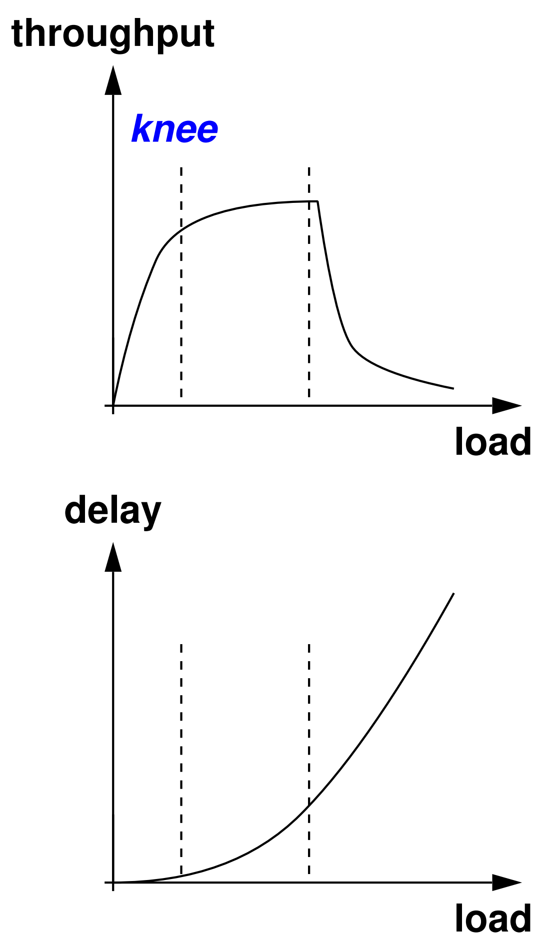
- Tries to evenly split bandwidth between all flows
  - $\left(\sum x_i\right)^2 / n \left(\sum x_i^2\right)$ where $x_i = A_i / D$
    - A=allocation, D=demand (identical for all i)

- Is this good or bad? Why is this hard?
  - not everyone needs the same bandwidth
  - fairness needs measured over long periods of time
  - evaluating this definition requires per-flow state
  - other granularities of fairness: per-user (don't let multiple flows get more bandwidth), per-host
  - TCP does not always split bandwidth equally between flows (packets get dropped at routers)
    - TCP throughput depends on RTT
    - Old flow gets buffer, new flow cannot get in

Power and Efficiency

- Power = throughput$^{\alpha}$ / response time
  - why not consider throughput and response time separately?
    - want to evaluate both, and they trade-off against each other
- Shows trade-off between throughput and response time
- Efficiency?
  - efficiency = power / (power at knee)

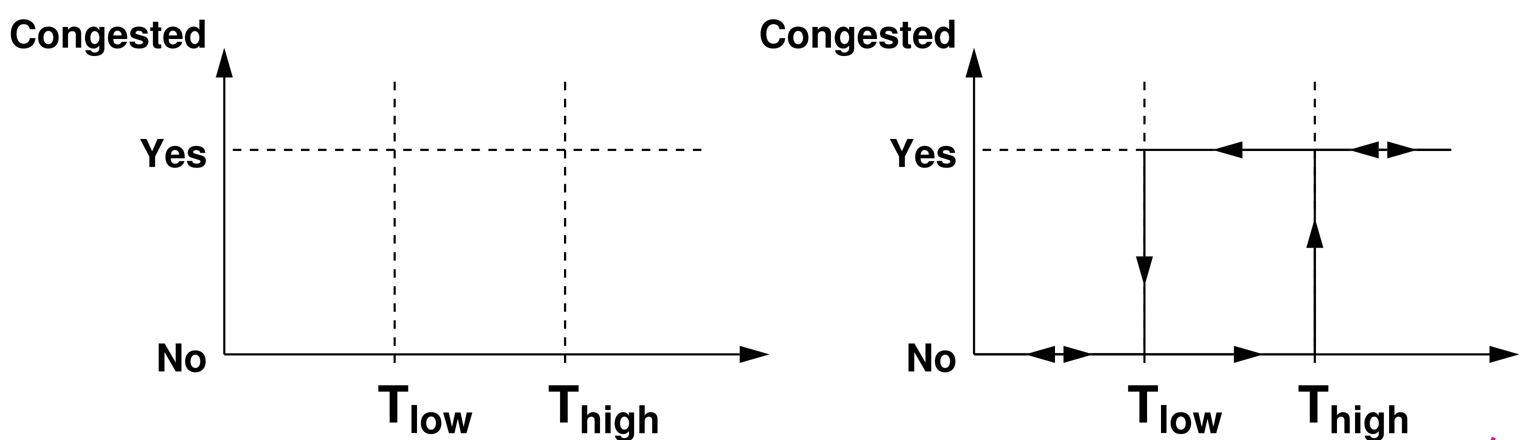
Power and Load



- Throughput and delay change due to load
  - want to optimize power

Other Issues: Measuring Congestion

- Measuring congestion
  - either utilization > T_{util} or queue length > T_{ql}
  - should use T_{low} and T_{high} (high and low watermarks) to provide hysteresis
  - why? use hysteresis to reduce the rate of change in congestion feedback



The Use of Hysteresis

- If we use queue lengths, at what queue lengths should we generate feedback?
  - Threshold or hysteresis?
  - Conventional wisdom says hysteresis

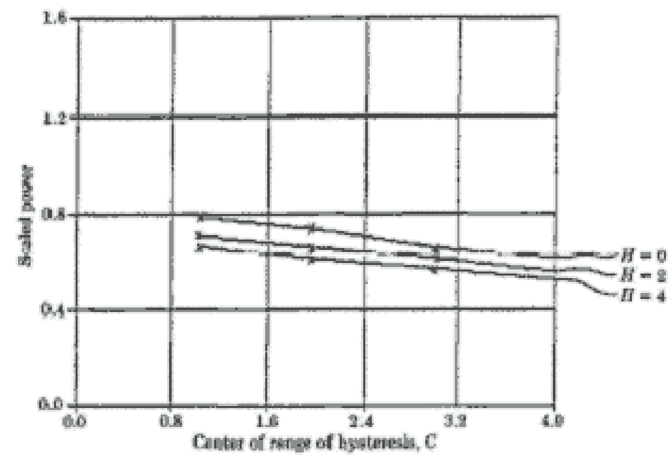


Fig. 2. Behavior of power with hysteresis.

- Surprisingly, simulations showed that if you want to increase power
  - Use *no* hysteresis
  - Use average queue length threshold of 1
  - Maximizes power function
    Power = throughput/delay

Policies Summary

- Decision frequency
  - adjust once per window (wait one RTT after adjustment for next adjustment)
- Use of information
  - keep history or not? (no)
- "signal filtering"
  - how many congestion bits $\Rightarrow$ congestion? (50%)
- Increase/decrease algorithms
  - AIMD

DEC-bit Evaluation

- Relatively easy to implement
- No per-connection state
- Stable
- Assumes cooperative sources
- Conservative window increase policy
- Some analytical intuition to guide design
  - Most design parameters determined by extensive simulation