

Congestion Notification

Bill Cheng

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



Copyright © William C. Cheng

Computer Communications - CS551

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*



Copyright © William C. Cheng

Computer Communications - CS551

Router Congestion Notification

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



Copyright © William C. Cheng

Computer Communications - CS551

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



Copyright © William C. Cheng

Computer Communications - CS551

Congestion Avoidance

(DEC-bit)

[Ramakrishnan90a]

Bill Cheng

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



Copyright © William C. Cheng

Computer Communications - CS551

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



Copyright © William C. Cheng

Computer Communications - CS551

Copyright © William C. Cheng

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

Computer Communications - CSC1 504

Copyright © William C. Cheng

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: $cwnd = 0.875 \times cwnd$)

Computer Communications - CSC1 504

Copyright © William C. Cheng

Components of a Congestion Avoidance System

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

Computer Communications - CSC1 504

Copyright © William C. Cheng

Options

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

Computer Communications - CSC1 504

Copyright © William C. Cheng

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)?

Computer Communications - CSC1 504

Copyright © William C. Cheng

The DEC-bit Scheme

- Basic ideas:
 - On congestion, router sets a bit (C) 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?)

Computer Communications - CSC1 504

Copyright © William C. Cheng

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)

Copyright © William C. Cheng

Copyright © William C. Cheng

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

Copyright © William C. Cheng

Copyright © William C. Cheng

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

Copyright © William C. Cheng

Copyright © William C. Cheng

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

Copyright © William C. Cheng

Copyright © William C. Cheng

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

Copyright © William C. Cheng

Copyright © William C. Cheng

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

Copyright © William C. Cheng