Congestion control

How does congestion arise?

1 M segs/sec
1 M segs/sec
1 M segs/sec (max capacity)

- Router receives data faster than it can send it
  - Downstream link or router congested or failed
  - Extreme traffic on one output link

- Usually a transient condition, but crisis can occur if not relieved quickly
  - Congestion cascades back into upstream routers
  - Whole sections of the network can be blocked quickly
Congestion can lead to catastrophe

Max network capacity
Ideal performance
Desired performance
Effects of large-scale congestion

Congestion indicators

• At routers:
  – Average queue lengths
  – Number of dropped queue items
  – Ping delay to neighbors

• At senders, receivers:
  – Number of timeouts
  – RTT (measured from data send to ACK receive)
  – Ping delay
  – Receive buffer fullness
Some things to keep in mind

- Generating additional traffic to report congestion makes the situation worse – piggybacking on existing traffic is better.
- Short timeouts lead to needless re-transmits, aggravating congestion.
- Some parts of the network can be fine at the same time that other parts are crashing.

Dealing with congestion
Some Methods for controlling congestion

- Add resources
- Prevent it (Open-loop methods)
- Detect and Relieve it (Closed-loop methods)

Adding resources

- Spread load over multiple routes
- Activate backup routers, links
- Increase power (satcom)

- Adding resources won’t be feasible in many situations – we need to be able to deal with congestion using the resources we have
Open-loop methods

The idea

• Prevent congestion from happening
  – Usually, no feedback from the network

• Some approaches:
  – Adjust retransmission policy
  – Adjust ACK policy
  – Spread traffic
  – Adjust TTL usage
  – Improve queuing methods
Retransmission policy

- The more retransmissions, the more likely there will be congestion
  - Increasing timeouts tends to reduce congestion
  - Minimizing amount of data retransmitted reduces congestion
    - Selective repeat instead of go-back-n
    - Receiver able to order out-of-order data instead of asking for repeat

Acknowledgement Policy

- ACK each packet immediately → extra ACK traffic

- Save ACKs to piggyback on reverse data → more timeouts, so more retransmission traffic
Routing approaches

- Spreading traffic over multiple paths tends to prevent congestion

TTL

- Long TTL $\rightarrow$ Lost packets clog up the network
- Short TTL $\rightarrow$ Less direct clogging, but more timeouts $\rightarrow$ more retransmissions
Improved queuing

Standard FIFO

Priority FIFO

“Fair” queue

• One user penalized for another’s traffic
• No way to recognize hi-priority traffic

• Need some way to mark segment or channel priority

• No way to bias toward a hi-pri channel

Weighted fair queuing

Combines features of priority and fair queuing

Each queue is assigned a weight that determines the number of queued items that will be taken each output cycle
Closed-Loop Approaches

The idea

• Monitor the network to detect increase in congestion, then act to reduce it

• Some approaches:
  – Allow routers to report congestion to senders
  – Allow routers to drop load
  – Adjust the TCP send window size dynamically
“DEC bit”

- When a router is nearing overload, it sets a special "congestion bit" in the packets/segments it processes.
- The congestion bit is returned to senders as part of ACKs
- When a sender sees congestion bit set, it can reduce send rate to that destination

Choke packets

- When a router is nearing overload, it sends a special "choke packet" to the source(s) sending traffic at the moment. The choke packet identifies the destination from the offending packets.
- A source that receives a choke packet reduces the rate of traffic to the destination by a fixed percentage. This lasts for a fixed amount of time.
- An alternate: Choke packets on a hop-by-hop basis
Random Early Deletion (RED)

- Designed to work with TCP Congestion Window approach
- When router nears overload, it starts randomly deleting “low priority” packets
- This triggers timeouts, which will ultimately cause congestion window size to be reduced.
- How does router determine “low priority”?
  - Randomly (across channels, within a channel)
  - Special flags (“Do Not Delete”…)

TCP Congestion Window

- Original TCP did not include congestion control, ARPAnet nearly collapsed (early ’80s) → TCP patched
- The idea: Make TCP slow down send rate as network congestion increases
- Method:
  - Send Window size set at MAX(advertised_size, CW_size)
  - Sending TCP sets CW_size dynamically based on perceived network congestion (as indicated by rate of timeouts)
    - Each time send is successful, CW_size = CW_size + 1
    - Each timeout, CW_size = CW_size/2
  - The effect: slow acceleration of send rate when the network is healthy, slam on the brakes when network starts slowing down (“additive increase, multiplicative decrease”)
Source-based control (TCP-Vegas)

- Sending TCP determines “no-congestion” RTT (usually determined by ACK for first seg to a destination)
- Dynamic RTT (measured from ACKs) is compared to no-congestion RTT.
- As RTT increases, CW_size is reduced