Retransmissions

• Focus
  – How do we decide when to retransmit?

• Topics
  – RTT estimation
  – Karn/Partridge algorithm
  – Jacobson/Karels algorithm
But first, a brief Internet history...

- **1969**: ARPANET created
- **1972**: TELNET created, RFC 318
- **1973**: FTP created, RFC 454
- **1977**: MAIL created, RFC 733
- **1980**: TCP & IP created, RFC 793 & 791
- **1982**: DNS created, RFC 883
- **1984**: NNTP created, RFC 977
- **1986**: ARPANET dissolved
- **1990**: ARPANET dissolved
- **1991**: WWW/HTTP
- **1992**: MBONE
- **1995**: Multi-backbone Internet
TCP: This is your life...

1974
TCP described by Vint Cerf and Bob Kahn
In IEEE Trans. Comm.

1975
Three-way handshake
Raymond Tomlinson
In SIGCOMM 75

1982
TCP & IP
RFC 793 & 791

1983
Nagel’s algorithm to reduce overhead of small packets; predicts congestion collapse

1985
BSD Unix 4.2 supports TCP/IP

1986
Congestion collapse observed

1987
Karn’s algorithm to better estimate round-trip time

1988
Van Jacobson’s algorithms for congestion avoidance and congestion control
(most implemented in 4.3BSD Tahoe)

1990
4.3BSD Reno
fast retransmit delayed ACK’s

1976
Congestion collapse observed

1984
Van Jacobson’s algorithms implemented

1985
4.3BSD Tahoe

1988
TCP & IP

1989
TCP & IP

1990
4.3BSD Reno
fast retransmit delayed ACK’s
TCP: After 1990

1993
TCP Vegas
(Brakmo et al)
real congestion avoidance

1994
T/TCP
(Braden)
Transaction TCP
ECN
 Explicit Congestion Notification

1996
SACK TCP
(Floyd et al)
Selective Acknowledgement
Hoe
Improving TCP startup
FACK TCP
(Mathis et al)
extension to SACK

2006
PCP

djw // CSE 461, Fall 2009
Deciding When to Retransmit

• How do you know when a packet has been lost?
  – Ultimately sender uses timers to decide when to retransmit

• But how long should the timer be?
  – Too long: inefficient (large delays, poor use of bandwidth)
  – Too short: may retransmit unnecessarily (causing extra traffic)
  – A good retransmission timer is important for good performance

• Right timer is based on the round trip time (RTT)
  – Which varies greatly in the wide area (path length and queuing)
Buffers at routers used to absorb bursts when input rate > output
Loss (drops) occur when sending rate is persistently > drain rate
Effects of Early Retransmissions

![Graph showing packet sequence number vs. send time]
Congestion Collapse

• In the limit, early retransmissions lead to congestion collapse
  – Sending more packets into the network when it is overloaded exacerbates the problem of congestion
  – Network stays busy but very little useful work is being done

• This happened in real life ~1987
  – Led to Van Jacobson’s TCP algorithms, which form the basis of congestion control in the Internet today
    [See “Congestion Avoidance and Control”, SIGCOMM’88]
Estimating RTTs

- Idea: Adapt based on recent past measurements

- Simple algorithm:
  - For each packet, note time sent and time ack received
  - Compute RTT samples and average recent samples for timeout
    - $\text{EstimatedRTT} = \alpha \times \text{EstimatedRTT} + (1 - \alpha) \times \text{SampleRTT}$
    - This is an exponentially-weighted moving average (low pass filter) that smoothes the samples. Typically, $\alpha = 0.8$ to $0.9$.
    - Set timeout to small multiple (2) of the estimate
Estimated Retransmit Timer
Karn/Partridge Algorithm

- Problem: RTT for retransmitted packets ambiguous

  ![Diagram](image)

- Solution: Don’t measure RTT for retransmitted packets and do not relax backed of timeout until valid RTT measurements
Jacobson/Karels Algorithm

- **Problem:**
  - Variance in RTTs gets large as network gets loaded
  - So an average RTT isn’t a good predictor when we need it most

- **Solution:** Track variance too.

  - Difference = SampleRTT – EstimatedRTT
  - EstimatedRTT = EstimatedRTT + (δ x Difference)
  - Deviation = Deviation + δ(|Difference| - Deviation)
  - Timeout = µ x EstimatedRTT + φ x Deviation
  - In practice, δ = 1/8, µ = 1 and φ = 4
Estimate with Mean + Variance