Chapter 21: TCP Timeout and Retransmission

Introduction

- **Two examples of timeout and retransmission had already seen:**
  - 1. In the ICMP port unreachable (TFTP, Section 6.5)
  - 2. In the ARP example to a nonexistent host (Section 4.5)
- **TCP manage four different timers for each connection:**
  - 1. A retransmission timer (this chapter)
  - 2. A persist timer (Chapter 22)
  - 3. A keepalive timer (Chapter 23)
  - 4. A 2MSL timer (Section 18.6)
- **Simple Timeout and Retransmission Example:**
  - Line 4 is the transmission of “hello,world” and line 5 is its ACK.
  - Line 6 shows “and hi”. Line 7-18 are 12 retransmissions
  - Line 19 is the TCP finally gives up and sends a reset

Simple Timeout and Retransmission Example

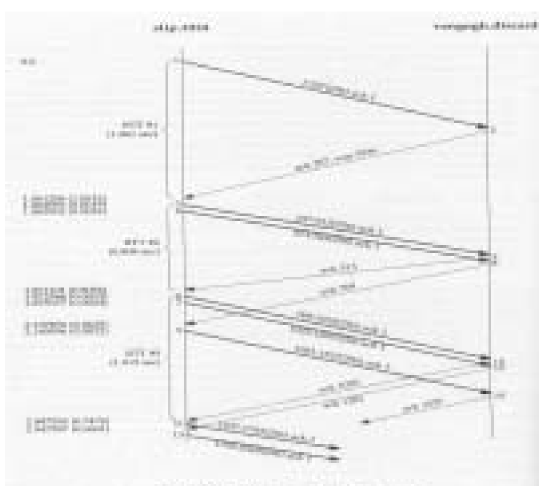
Round-Trip Time (RTO) Measurement

- **Two methods of RTO calculate:**
  - The original TCP specification method
    - $R \leftarrow \alpha R + (1 - \alpha) M$
    - $RTO = R\beta$
      - ✓ *R=>smoothed RTT estimator, α is smoothing factor = 0.9*
      - ✓ *β is delay variance factor = 2*
  - The Jacobson method
    - $Err = M - A$
    - $A \leftarrow A + g\,Err$
    - $D \leftarrow D + h(|Err| - D)$
    - $RTO = A + 4D$
      - ✓ *A is the smoothed RTT average,D is mean deviation*
      - ✓ *The gain g is 1/8(0.125),h is 0.25*

Round-Trip Time Measurement (Cont.)

- **Karn's Algorithm:**
  - A packet is transmitted,a timeout occurs,the packet is retransmitted with the longer RTO, and an acknowledgment is received
  - Is the ACK for the first transmission or the second?
  - This is called the *retransmission ambiguity problem*
  - Karn and Partridge: don't count retransmitted packets into the RTT estimator.
- **An RTT Example:**
  - sent 32768 bytes of data from *slip* to *vangogh.cs.berkeley.edu*
    - slip% sock -D -i -n32 vangogh.cs.berkeley.edu. discard
  - slip is connected to the 140.252.1 Ethernet by two SLIP links
  - MTU between slip and bsdi is 296
  - 32 1024-byte => 128 segment with 256 bytes of user data

An RTT Example

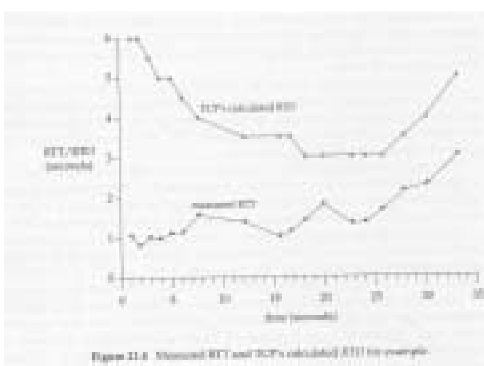


An RTT Example

- **Round-Trip Time Measurements:**
  - One connection, one timer
  - RTT#1 is 1.061 seconds => 3 clock ticks
  - RTT#2 is 0.808 seconds => 1 clock tick
  - RTT#3 is 1.015 seconds => 2 clock ticks
  - Segment 4,7,9 cannot be timed,since the timer is already being used by segment 3 and 6

An RTT Example

- In this complete example 18 RTT samples were collected



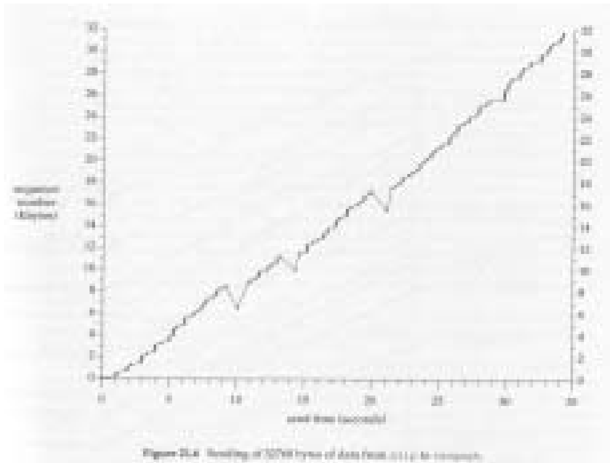
An RTT Example

- **RTT Estimator Calculations:**
  - The initial RTO=A+2D => 0+2×3 = 6 seconds
  - After 5.802 seconds RTO=A+4D => 0+4×3 = 12 seconds
  - The ACK arrives 467 ms after the retransmission.The A and D are not updated because of retransmission ambiguity
  - The ACK on line 4 is not timed since it is only an ACK

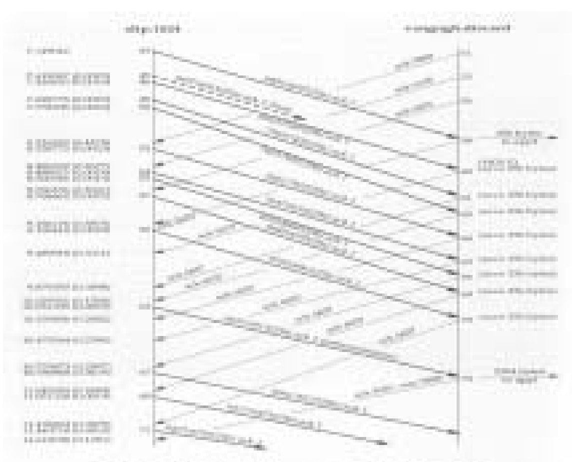
An RTT Example

- **RTO caculations**
  - first segment arrives => RTO=6 seconds
  - second segment arrives => RTO=6.3125 seconds
  - Fixed-point calculations that are actually used =>RTO is 6 seconds (not 6.3125)
- **Slow Start**
  - See the slow start algorithm in Section 20.6
- **Congetion Example:**
  - Normally the data points should move up and to the right, with the slope of the points being the transfer rate.
  - Retransmissions will appear as motion down and to the right.

Congetion Example



Congetion Example



Congestion Example

- **The Jacobson's fast retransmit algorithm:**
  - It is followed by his fast recovery algorithm.The third of the duplicate ACKs was received that forces to retransmit
  - Berkeley-derived implementation when the third one is received, assume that a segment has been lost and retransmit only one segment
- **When the missing data arrives (segment 63):**
  - The receiving TCP now has data bytes 6657-8960 in its buffer, and passes these 2304 bytes to the user process.
  - All 2304 bytes are acknowledged in segment 72

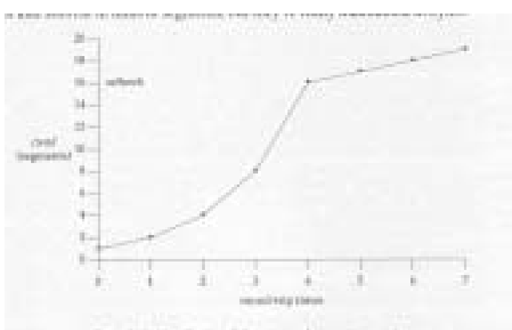
Congestion Avoidance Algorithm

- **What's congestion avoidance?**
  - It is a way to deal with lost packets
  - Assumption: packet loss caused by damage is very small (< 1%)
  - The loss of a packet signals congestion somewhere in the path between the source and the destination.
- **Two indications of packet loss:**
  - A timeout occurring
  - The receipt of duplicate ACKs
- **Congestion avoidance and slow start are independent algorithms with different objectives.**
- **Two variables be maintained for each connection: *cwnd* and *ssthresh***

Congestion Avoidance Algorithm (Cont.)

- **Congestion avoidance algorithm operates:**
  - 1.Initialization=>*cwnd* is one segment, *ssthresh* is 65535 bytes
  - 2.TCP output never sends more than the minimum of cwnd and the receiver's advertised window
  - 3.When congestion occurs, one-half of the current window size is saved in *ssthresh*. If timeout, *cwnd* is set to one segment
  - 4.When new data is acknowledged by the other end, increase *cwnd*
- **If *cwnd* is less than or equal to *ssthresh*, doing slow start. Otherwise, we're doing congestion avoidance.**
- **Congestion avoidance dictates that *cwnd* be increased by *1/cwnd*, plus a small fraction of segment size, each time an ACK is received: *an additive increase*.**
- **Increase *cwnd* by at most one segment each round-trip time (regardless how many ACKs are received in that RTT)**

Congestion Avoidance Algorithm



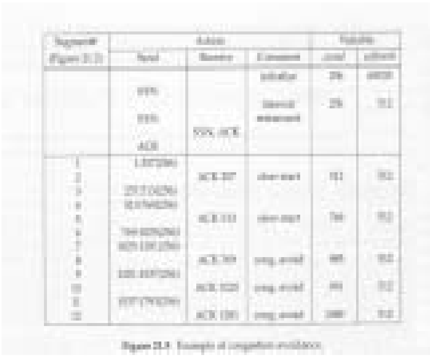
Fast Retransmit and Fast Recovery Algorithms

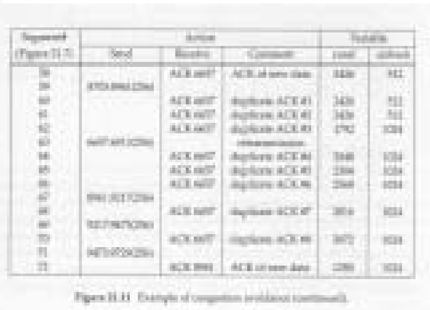
- **TCP is required to generate an immediate ACK when an out-of-order segment is received.**
- **Duplicate ACK: a segment is received out of order.**
  - Two possible situations:
    - Packet loss – if two or more duplicate ACKs are received – a strong indication of segment loss.
    - First out last in
- **The receipt of the duplicate ACKs tells us more than just a packet has been lost:**
  - A segment leaves the network into receiver's buffer.
  - There still is data flowing between the two ends – DON'T reduce the flow abruptly by going into slow start.

Fast Retransmit and Fast Recovery Algorithms (Cont.)

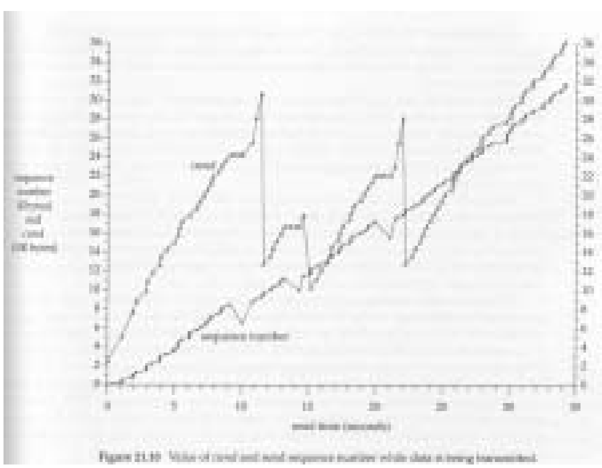
- **Fast retransmit algorithm:**
  - If three or more duplicate ACKs are received in a row, indicate a segment has been lost, then retransmission the missing segment
- **Fast recovery algorithm:**
  - Next, congestion avoidance, but not slow start is performed
- **Congestion Example**
  - In congestion avoidance:
    - cwnd←cwnd＋(segsize ｘ segsize)／cwmd＋ segsize ／ 8
  - By fast retransmit and fast recovery, we can send a new data segment when cwnd > unacknowledged bytes
  - *cwnd* is allowed to keep increasing while the duplicate ACKs are received, since each duplicate ACK means the a segment has left the network.

Congestion Example (Continued)





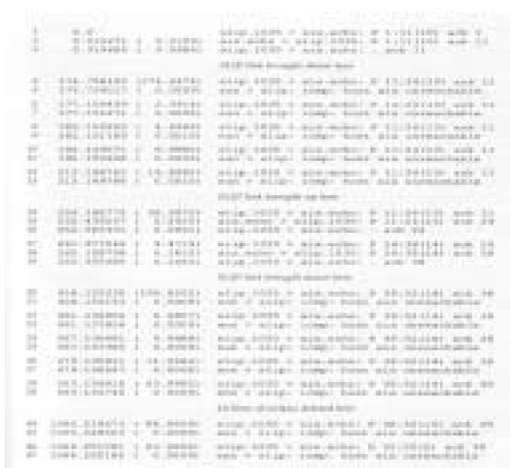
Congestion Example (Continued)



ICMP Errors

- **Berkeley-based implementations handle ICMP errors as follows:**
  - A received source quench causes the *cwnd* set to one segment to initiate slow start, but the *ssthresh* is not changed
  - A received host unreachable or network unreachable is effectively ignored, since these two errors are considered transient.
- **An Example**
  - slip% sock aix echo
  - test line
  - test line
  - SLIP link is brought down here
  - another line — type this line and retransmissions
  - SLIP link is reestablished here
  - …….. — after the last line,SLIP link is brought down
  - read error: No route to host — TCP finally gives up

ICMP Errors



Repacketization

- When TCP times out and retransmits, it does not have to retransmit the identical segment again. Instead, TCP is allowed to perform *repacketization*, sending a bigger segment, which can increase performance.
- notice on bytes of line 3 and 6 in following illustration

Summary

- **TCP calculates a smoothed RTT estimator and a smoothed mean deviation estimator. Then use these two estimators to calculate the next retransmission timeout value.**
- **We see many of TCP's algorithm in action:**
  - slow start
  - congestion avoidance
  - fast retransmit
  - fast recovery
- **Effect which various ICMP errors have on a TCP connection**
- **How TCP is allowed to repacketize its data**