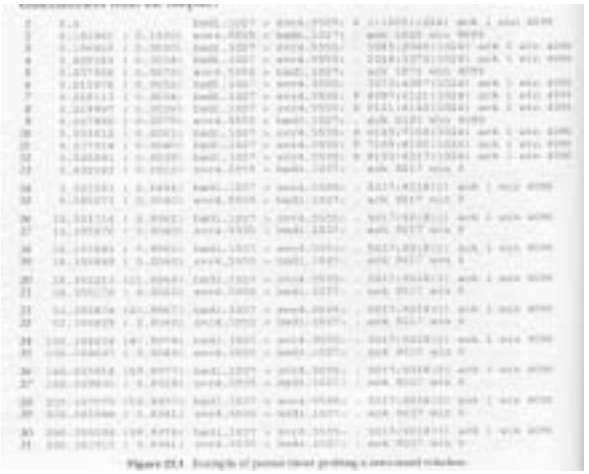
Chapter 22:
TCP Persist Timer

Introduction

- **Why we use persist timer:**
  - If an acknowledgment is lost, we could end up with both sides waiting for the other
    - To prevent deadlock from occurring the sender uses a persist timer
    - The sender queries the receiver periodically to find out if the window has been increased. These segments from the sender are called *window probes*.
- **An Example:**
  - We'll invoke the server as
    - svr4 % sock -i -s -P100000 5555
  - The server sleep for 100,000 seconds (27.8 hours) before reading from the network
  - The client on host bsdi and performs 1024-byte writes to port 5555 on the server

An Example



Silly Window Syndrome

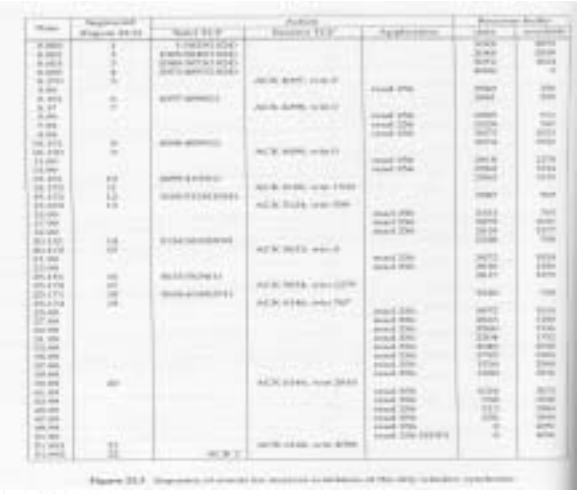
- **About persist timer:**
  - It always bounded between 5 and 60 seconds
  - The persist state which is different from the retransmission timeout is that TCP never gives up sending window probes.
- **When it occurs small amounts of data are exchanged across the connection, instead of full-sized seqments.**
- **What is Silly Window Syndrome**
  - It can be caused by either end:
    - The receiver can advertise small windows
    - The sender can transmit small amounts of data.

Silly Window Syndrome (Cont.)

- **Correct avoidance is performed on both ends:**
  - The receiver must not advertise small windows
  - The sender is not transmitting unless one of conditions is true:
    - A full-sized segment can be sent
    - We can send at least one-half of the maximum sized window ever advertised
    - We can send everything we have if:
      - ✓ *We are not expecting an ACK (no outstanding data), or*
      - ✓ *The Nagle algorithm is disabled (W ≠ 1)*

Silly Window Syndrome

Silly Window Syndrome



Summary

- TCP's persist timer is set by one end of a connection when it has data to send, but has been stopped because the other end has advertised a zero-sized window.
- TCP's avoidance of the silly window syndrome is to prevent TCP from advertising small windows or from sending small segments