Transport Layer

- Provides reliable, efficient, data server to application layer
  - End point of the efforts of the other layers
  - Applications can use standard set of interfaces
  - Underlying layers can be unreliable.
    - Some duplication of error handling of network layer
    - Better efficiency if network layer handles problems
      - But better to duplicate in Transport layer than having every application implement reliability features
Transport Layer Primitives

- Listen - wait for connection
- Connect -- establish connection
- Send -- send data
- Receive -- receive data
- Disconnect -- end connection
Berkeley Sockets

- Socket--create socket
- Bind--attach local address to socket
- Listen--accept connections, Q size
- Accept--block until data received
- Connect--establish connection
- Send--send data
- Receive--receive data
- Close
Establishing connections

- Sounds simple
  - Request, ack, ack
- Problems
  - Lost packets are not bad
  - Delayed duplicate packets are bad
    - Restrict subnet to time out packets
    - Set a low hop count limit
    - Timestamp each packet
Tomlinson packet lifetime bounding

- Consider sequence numbers which track clock
  - Can’t send packets faster than clock
    - Otherwise restart may result in duplicates
  - Sending too slow also causes problems.
    - Resynchronize sequence numbers
Three way handshake

- Different sequence numbers on each side
  - Duplicates are handled
Connection release

- Symmetric and asymmetric release
- Two Army problem
  - Easy proof no solution
- TCP double timer disconnect
TCP timers

- packet timeout varies
  - dynamic modification
  - \( T = a \times RTT + (1-a)M \)
  - Calculate deviation
    - \( d = aD + (1-a)|RTT-M| \)
- Karn’s algorithm
- persistence timer
- keepalive timer
`TCP Performance guidelines

• CPU speed more important than netspeed
  • 96 microS. Ether slot time vs 10 mS disk seek time
• Reduce packet count
  • Software overhead on packets counts
    • Nagel
• Minimize context switches
  • Expensive for current generation OS
  • Avoid additional copying
TCP Performance (Cont.)

- Easy to buy bandwidth
  - But not lower delay
- Avoid congestion
  - Rather than recover
- Avoid timeouts
  - Early timeout is expensive
Fast Packet Processing

- Per packet overhead
  - Prototype header
    - Fast copy
      - TCP: seq, checksum change
      - IP: header checksum, id change
  - Fast receive
    - Last packet pointer (port#)
    - Header prediction
    - Trailers
      - VM and checksums
  - Per byte overhead
Network speeds

- 1970s 1 Mip computer
  - Arpanet 56k connection
    - 18000 instructions per packet
- 2000 100 MIP computer
  - 30k packets/sec on 1 Gigabit
    - 1500 instructions/packet
  - Delay dominates over bandwidth