Computer Communication Networks

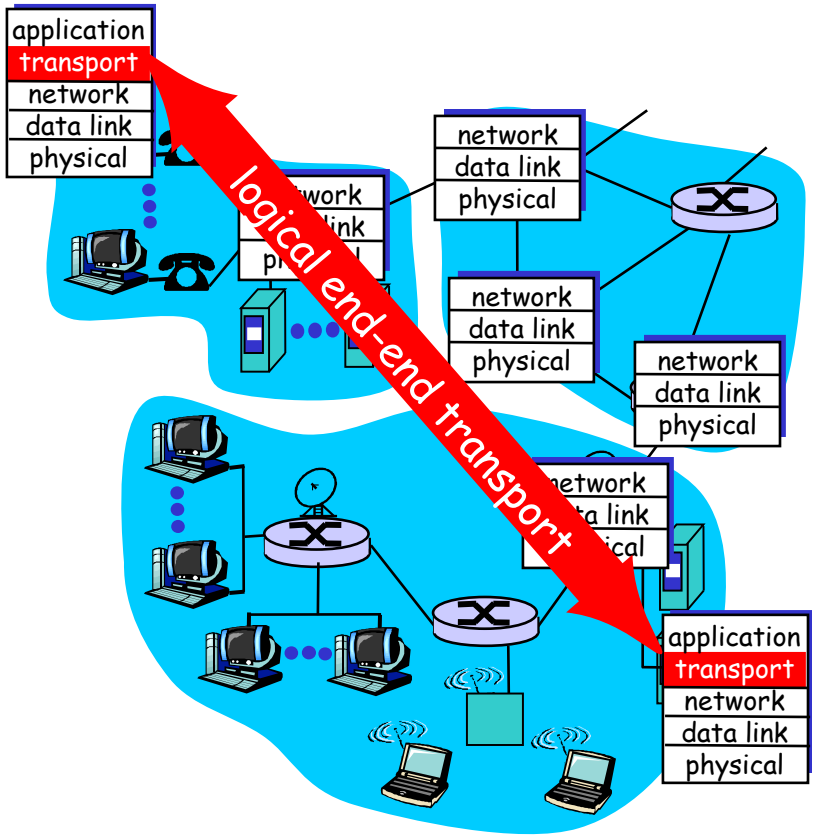
Transmission Control Protocol

Transport layer services

- Services offered by transport layer
  - endpoint-to-endpoint communication
    - *endpoint*: an application process in end-hosts
  - connection vs connectionless
  - reliable vs unreliable

Transport layer services and protocols

- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into segments, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP



Multiplexing/demultiplexing

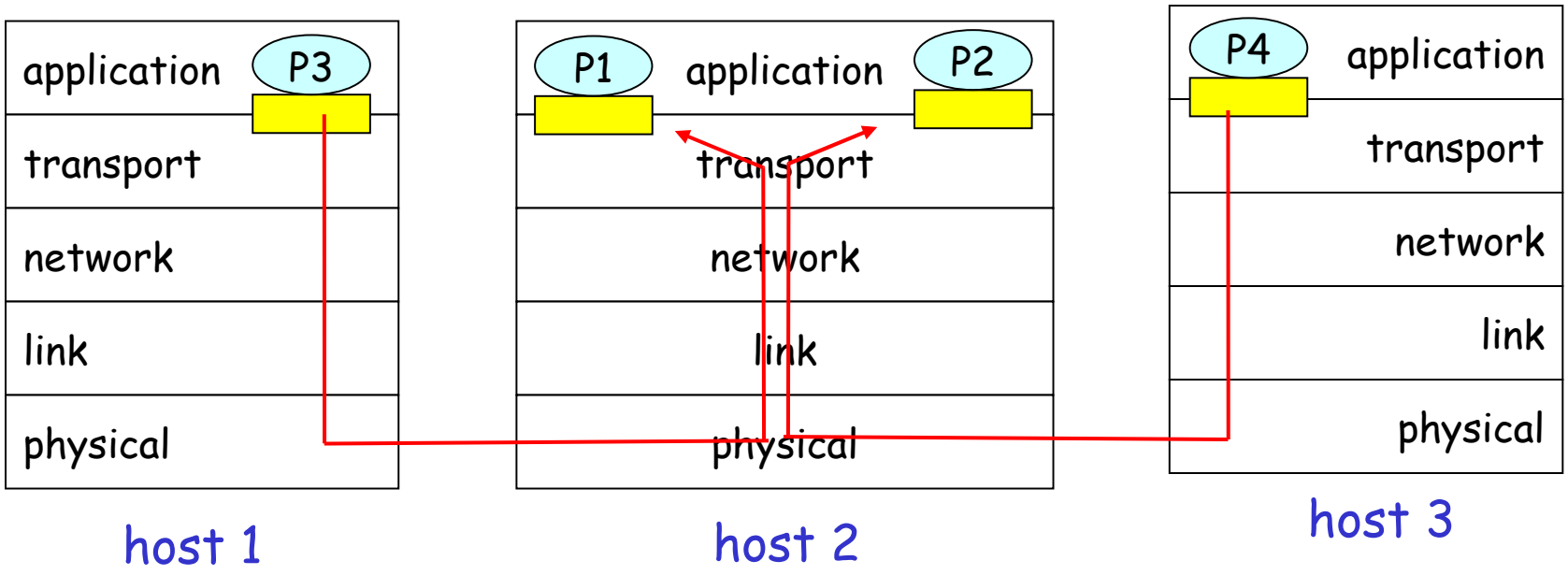
Demultiplexing at rcv host:

delivering received segments to correct socket

Multiplexing at send host:

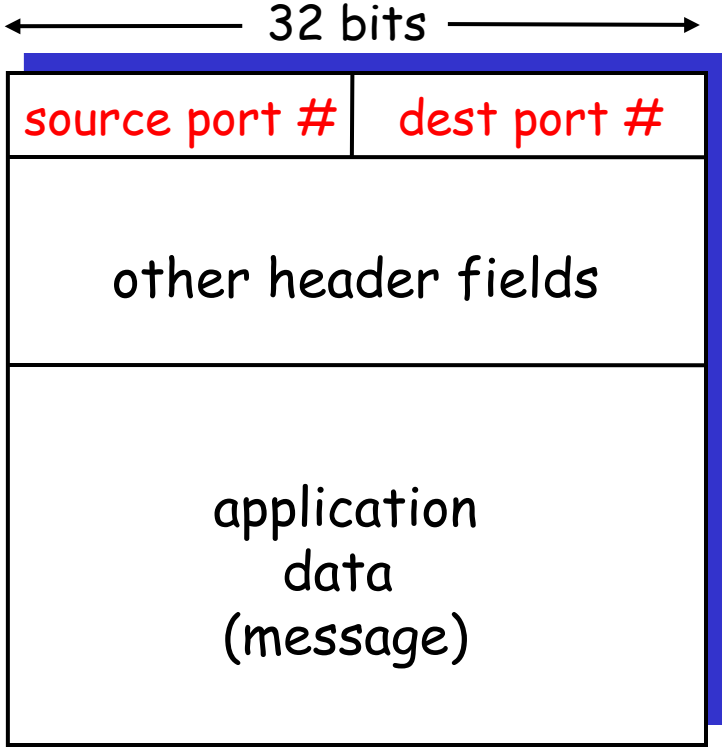
gathering data from multiple sockets, enveloping data with header (later used for demultiplexing)

= socket = process



How demultiplexing works

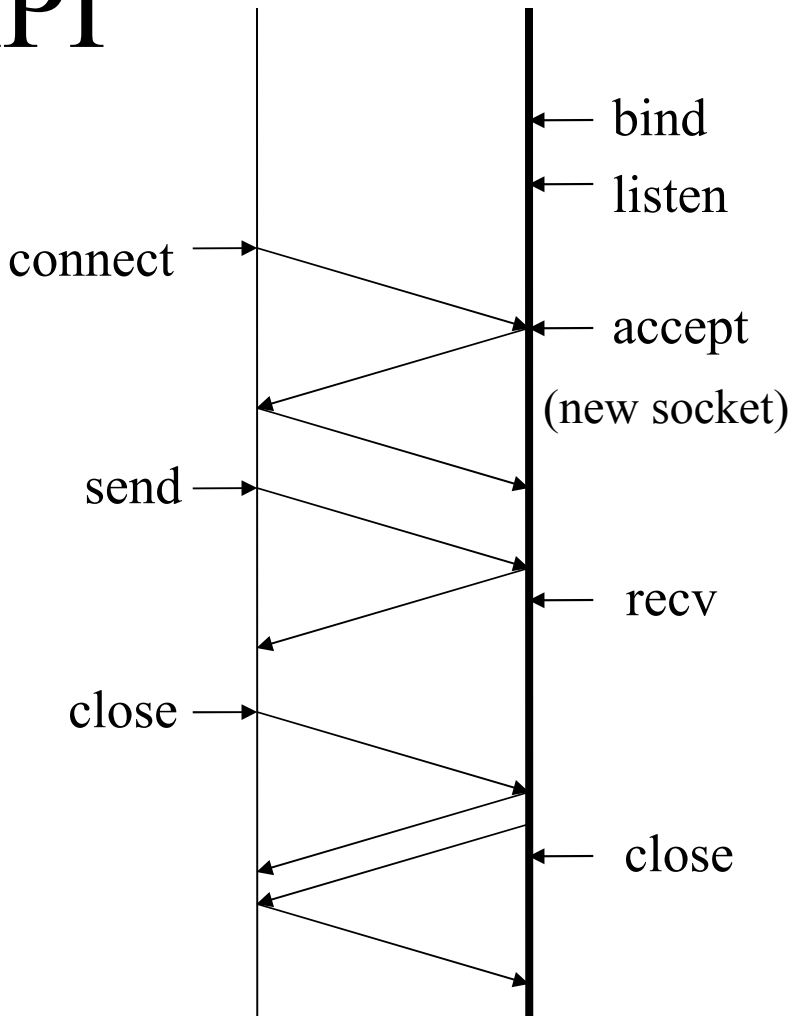
- host receives IP datagrams
  - each datagram has source IP address, destination IP address
  - each datagram carries 1 transport-layer segment
  - each segment has source, destination port number
- host uses IP addresses & port numbers to direct segment to appropriate socket



TCP/UDP segment format

Socket API

- Server
  - bind, listen
- Client
  - connect
- Server
  - accept
- Client-server
  - send, recv, close



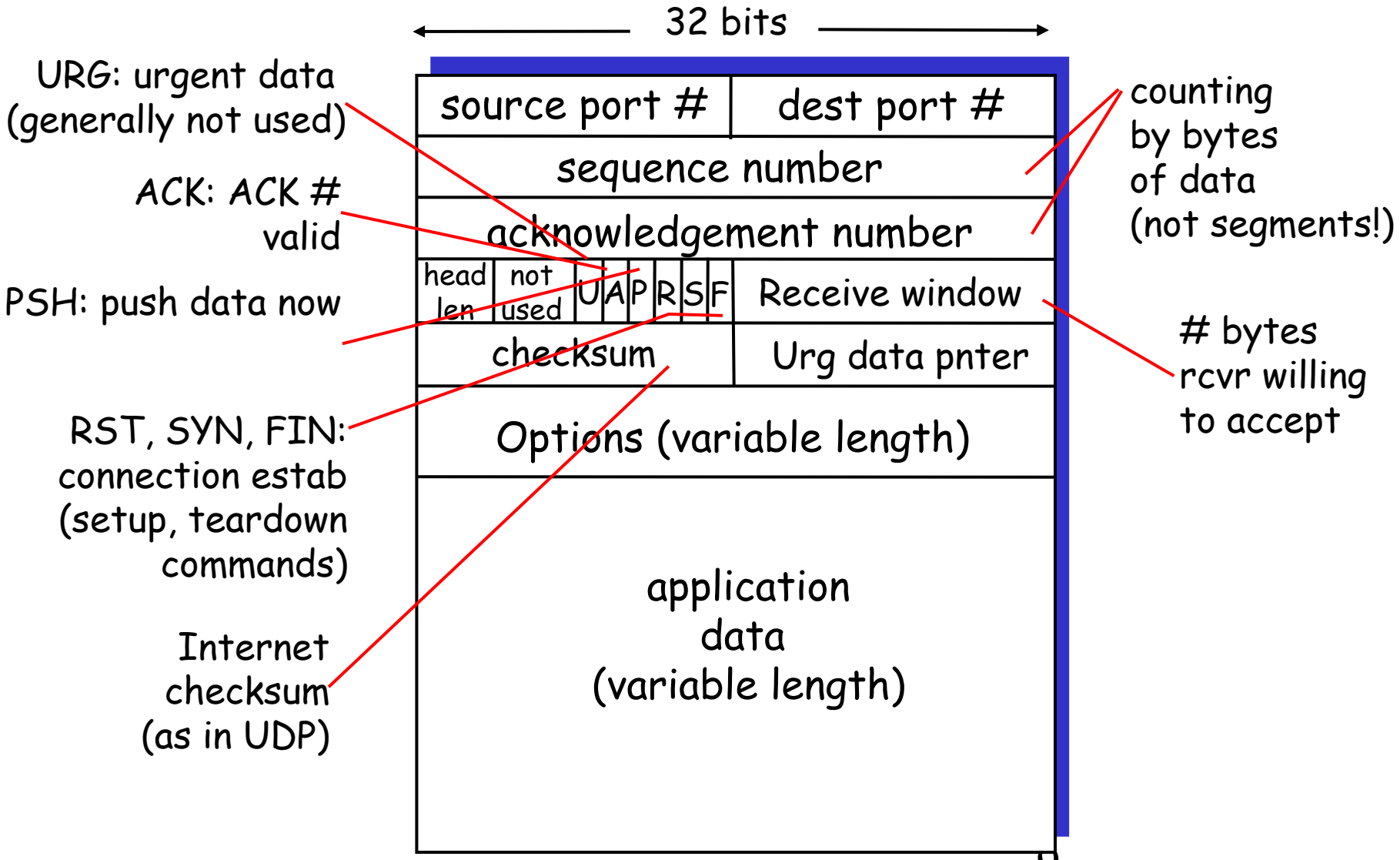
Socket, TCP, IP

- Socket
  - send (socketid, pointer_to_buffer, length, flags);
- TCP
  - TCP segmentation: TCP segments
  - TCP *packet* header: TCP control information
- IP
  - IP packetization: IP packets
  - IP packet header: IP control information

TCP

- Transmission control protocol [RFC793]
- Services offered by TCP
  - connection-oriented, bi-directional
  - reliable, in-sequence, stream-like
- Packets delivered by IP
  - maybe duplicated, lost, reordered, corrupted
- TCP protocol mechanisms
  - connection management
  - flow, error and congestion control

TCP packet header



TCP port number

- Port number (16-bit)
  - source, destination port numbers
    - multiplexing and de-multiplexing
- Port number allocation (ref: iana.org)
  - well-known port numbers (0~1023, privileged)
    - 80: http; 443: https
  - registered port numbers (1024~49151)
    - 8080: http-alt
  - dynamically allocated port numbers (49152~65535)

TCP connection ID

- TCP connections
  - connection: initiator, responder
    - (initiator IP, initiator port, responder IP, responder port)
- One connection: one flow in each direction
  - for each flow: source, destination
    - (source IP, source port, destination IP, destination port)
  - 5-tuple (or 4-tuple when protocol is implied)
    - (src IP, src port, protocol ID, dst IP, dst port)
- Socket, connection, flow

TCP sequence number

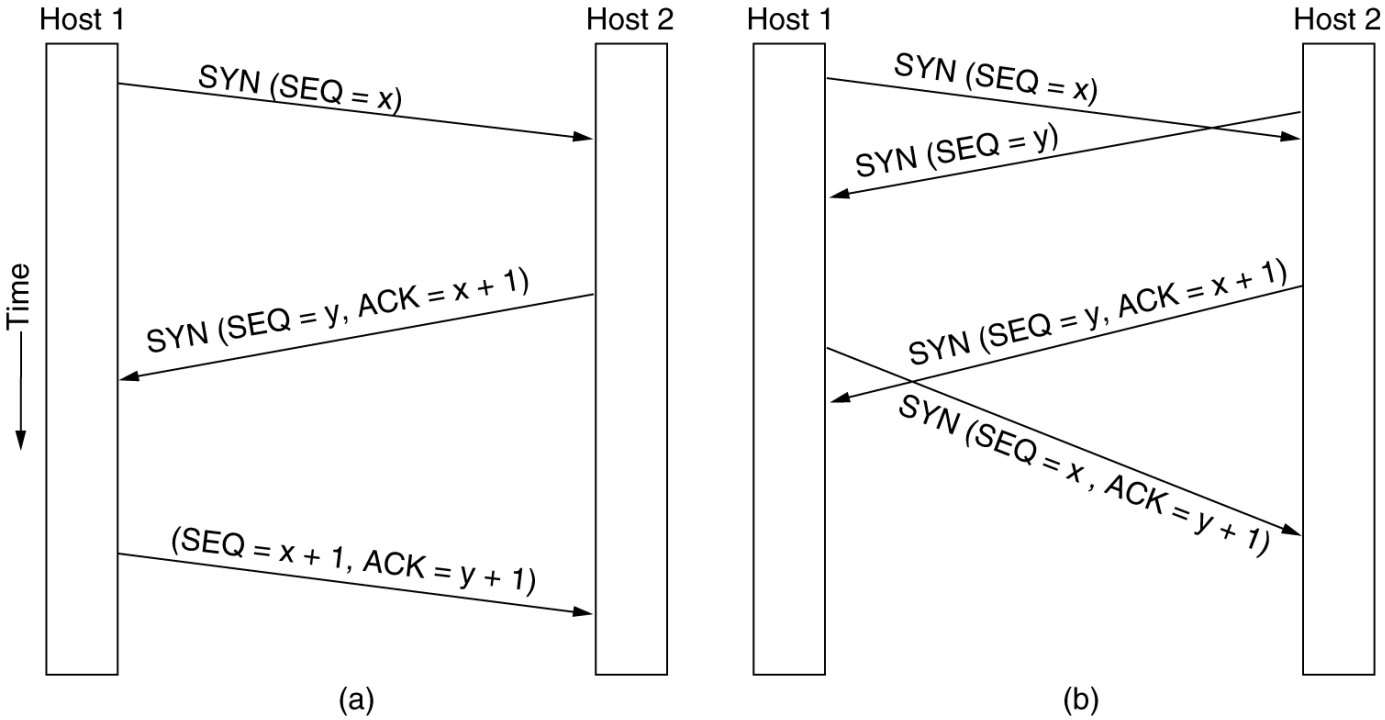
- Sequence number (32-bit)
  - byte sequence for the ***first*** byte in payload
    - exception: SYN/FIN sequence number
  - random initial sequence number
    - exchanged during 3-way handshake
  - sequence number rollover
- Acknowledgment number (32-bit)
  - byte sequence for the ***next*** byte to expect

TCP control flags

- URG: urgent pointer meaningful
- ACK
  - acknowledgment number meaningful
- PSH: logic message boundary
- RST: connection rest
- SYN
  - synchronization (connection establishment)
- FIN
  - finish (graceful connection release)

TCP connection establishment

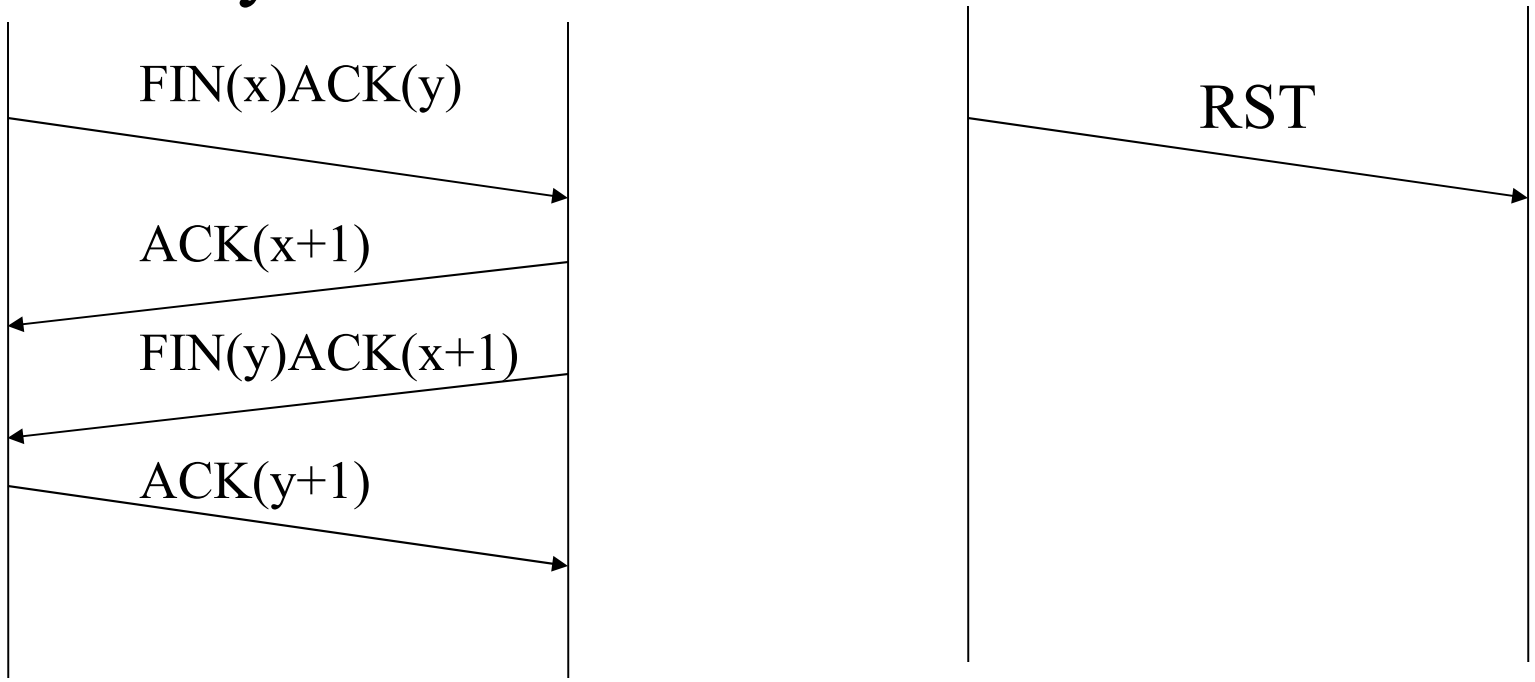
- 3-way packet handshake



- Simultaneous establishment attempts

TCP connection release

- 2-way handshake in each direction



- Connection reset

Summary

- TCP
  - services offered by TCP
  - TCP connection management
    - connection establishment
    - connection release
- Explore further
  - tcpdump (or Ethereal, Wireshark)

Next

- TCP flow and error control