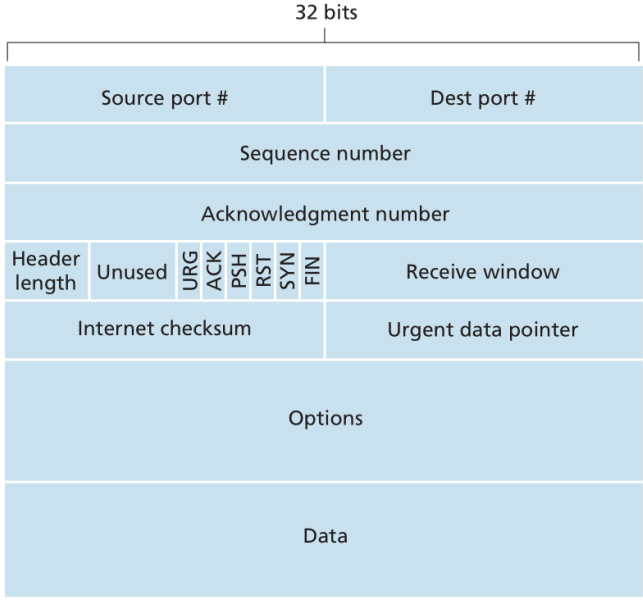
TCP packet structure

How is the TCP packet different from a UDP packet?

- UDP
  - Source port, destination port, length, checksum, data
- TCP
  - Sequence numbers
  - Flags
  - Receive window
  - Urgent data



Mountains & Minds

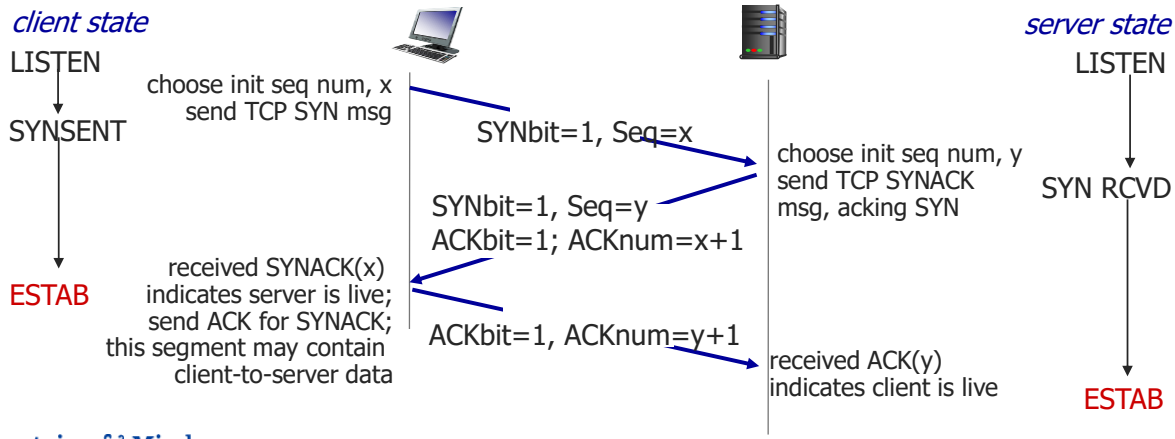
139

139

TCP 3-way handshake

Before exchanging data, sender/receiver "handshake":

- Agree to establish connection (each knowing the other willing to establish connection)
- Agree on connection parameters
- Allocate resources

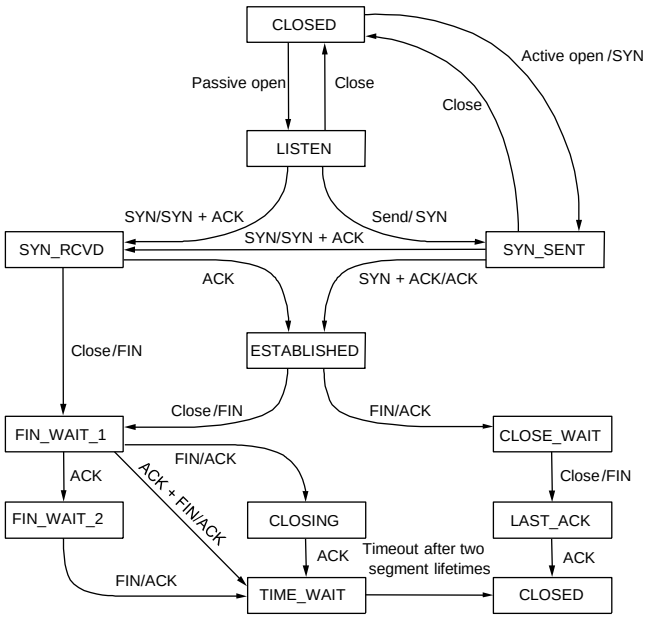


Mountains & Minds

140

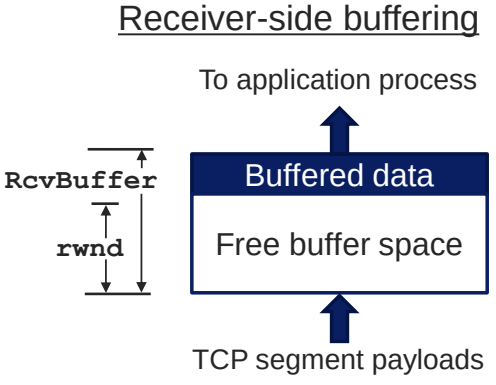
140

State Transition Diagram



Flow Control

- Receiver throttles sender to avoid receiver buffer overflow



- Receiver "advertises" free buffer space by including **`rwnd`** value in TCP header of receiver-to-sender segments
  - **`RcvBuffer`** size set via socket options (typical default is 4096 bytes)
  - Many operating systems autoadjust **`RcvBuffer`**
- Sender limits amount of unacked ("in-flight") data to receiver's **`rwnd`** value
- Guarantees receive buffer will not overflow

What happens after **`rwnd`** set to 0?

  - Sender stops sending and is not notified when buffer empties!
  - Sender will continue to send 1B packets to get **`rwnd`** updates

Nagle's Algorithm

- How long does sender delay sending data?
  - too long: hurts interactive applications
  - too short: poor network utilization – a packet may be mostly empty
  - strategies: timer-based vs self-clocking
- When application generates additional data
  - if fills a max segment (and window open): send it
  - else
    - if there is unack'ed data in transit: buffer it until ACK arrives
    - else: send it

- Nagle's algorithm can interact poorly with TCP's 'delayed acknowledgement' algorithm, so a way to disable it is quite commonly provided

TCP: Push vs Urgent

- PSH
  - TCP sender doesn't wait for more data (Nagle's algo)
  - Receiver forwards buffer content to application immediately
- URG
  - Receiver gets offset of urgent bytes (could be the whole packet)
  - Read by application first (potentially out of order)