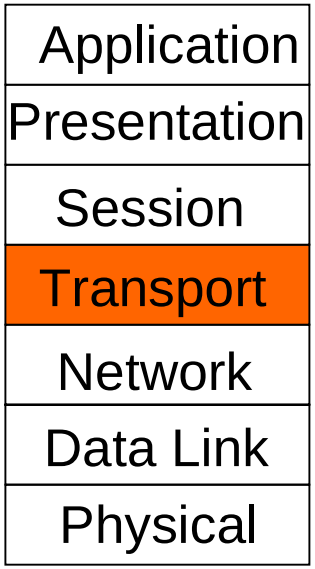
CSE/EE 461 – Lecture 11

David Wetherall

djw@cs.washington.edu

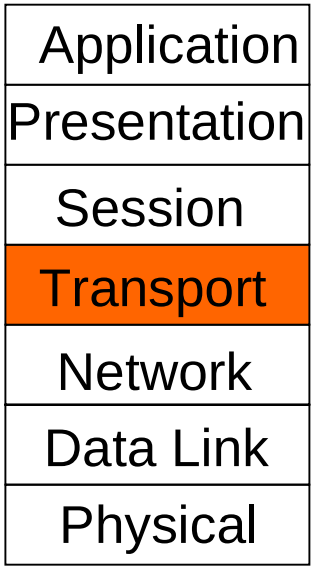
Last Time

- We began on the Transport layer
- Focus
  - How do we send information reliably?
- Topics
  - ARQ and sliding windows



This Time

- More on the Transport Layer
- Focus
  - How do we connect processes?
- Topics
  - Naming processes
  - Connection setup / teardown
  - Flow control



Naming Processes/Services

- Process here is an abstract term for your Web browser (HTTP), Email servers (SMTP), hostname translation (DNS), RealAudio player (RTSP), etc.

- How do we identify for remote communication?
  - Process id or memory address are OS-specific and transient ...

- So TCP and UDP use Ports
  - 16-bit integers representing mailboxes that processes “rent”
  - Identify process uniquely as (IP address, protocol, port)

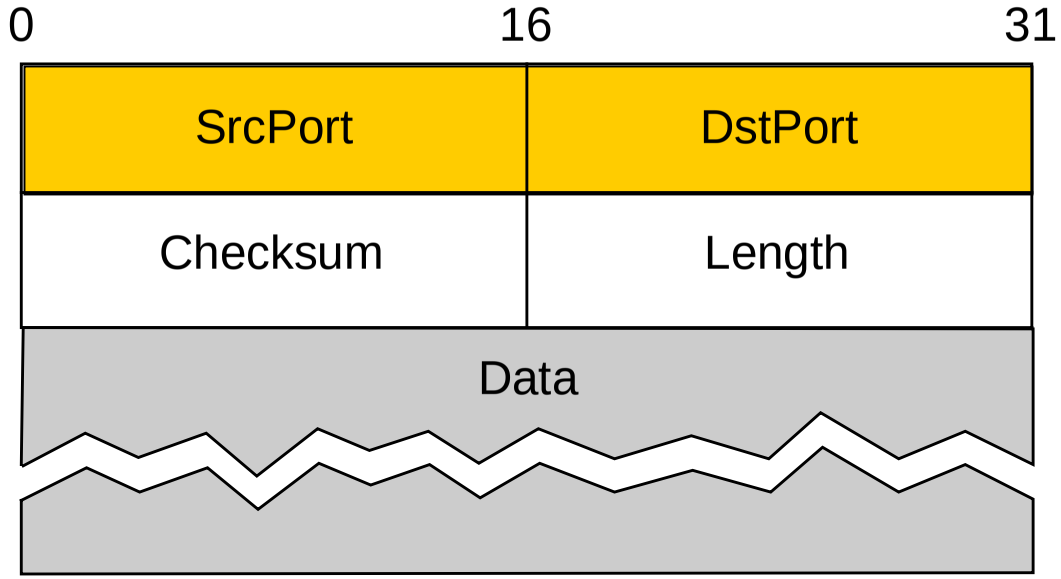
Picking Port Numbers

- We still have the problem of allocating port numbers
  - What port should a Web server use on host X?
  - To what port should you send to contact that Web server?

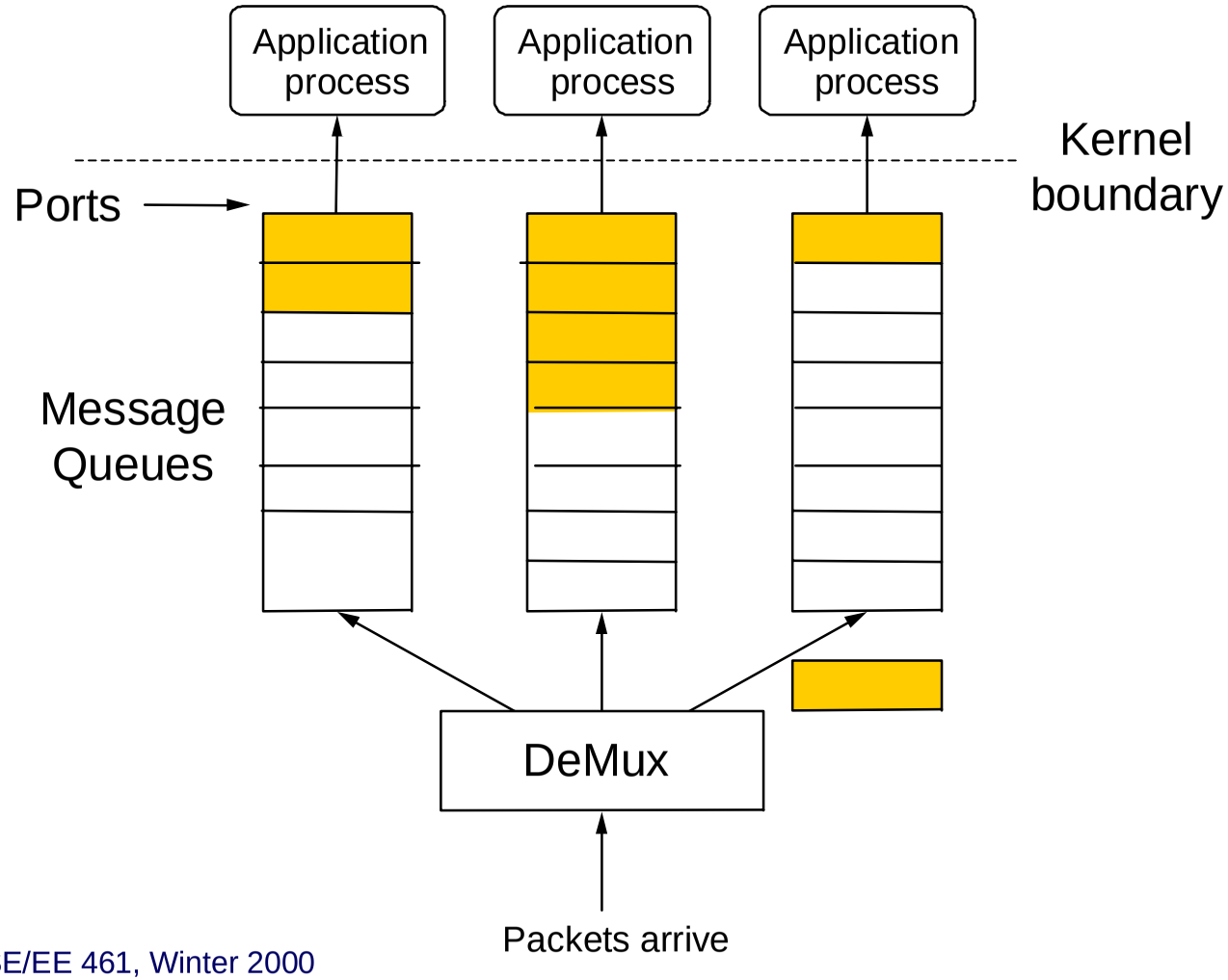
- Servers typically bind to “well-known” port numbers
  - e.g., HTTP 80, SMTP 25, DNS 53, ... look in /etc/services
  - Ports below 1024 reserved for “well-known” services
- Clients use OS-assigned temporary (ephemeral) ports
  - Above 1024, recycled by OS when client finished

User Datagram Protocol (UDP)

- Provides message delivery between processes
  - Source port filled in by OS as message is sent
  - Destination port identifies UDP delivery queue at endpoint

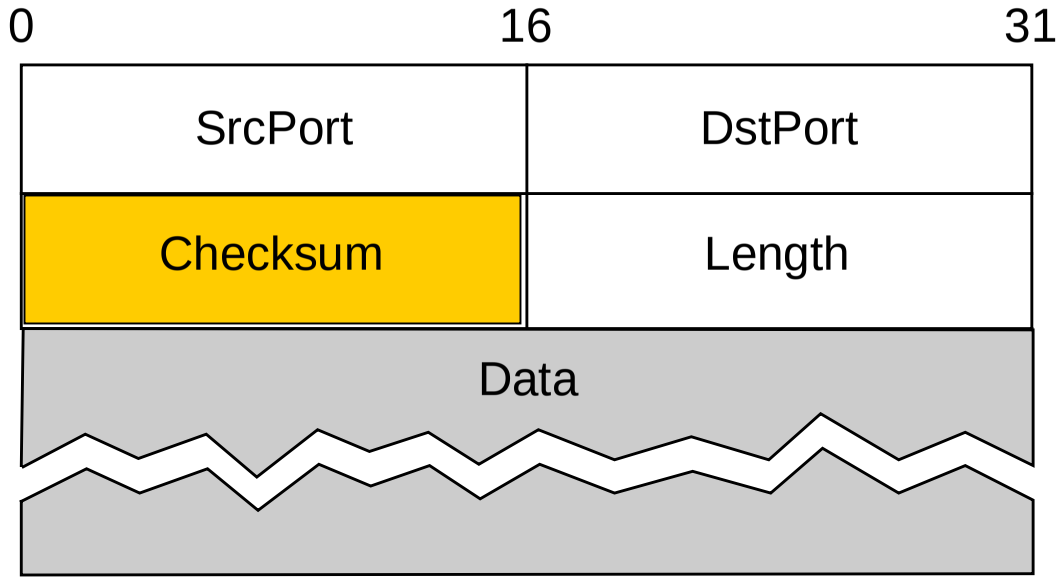


UDP Delivery



UDP Checksum

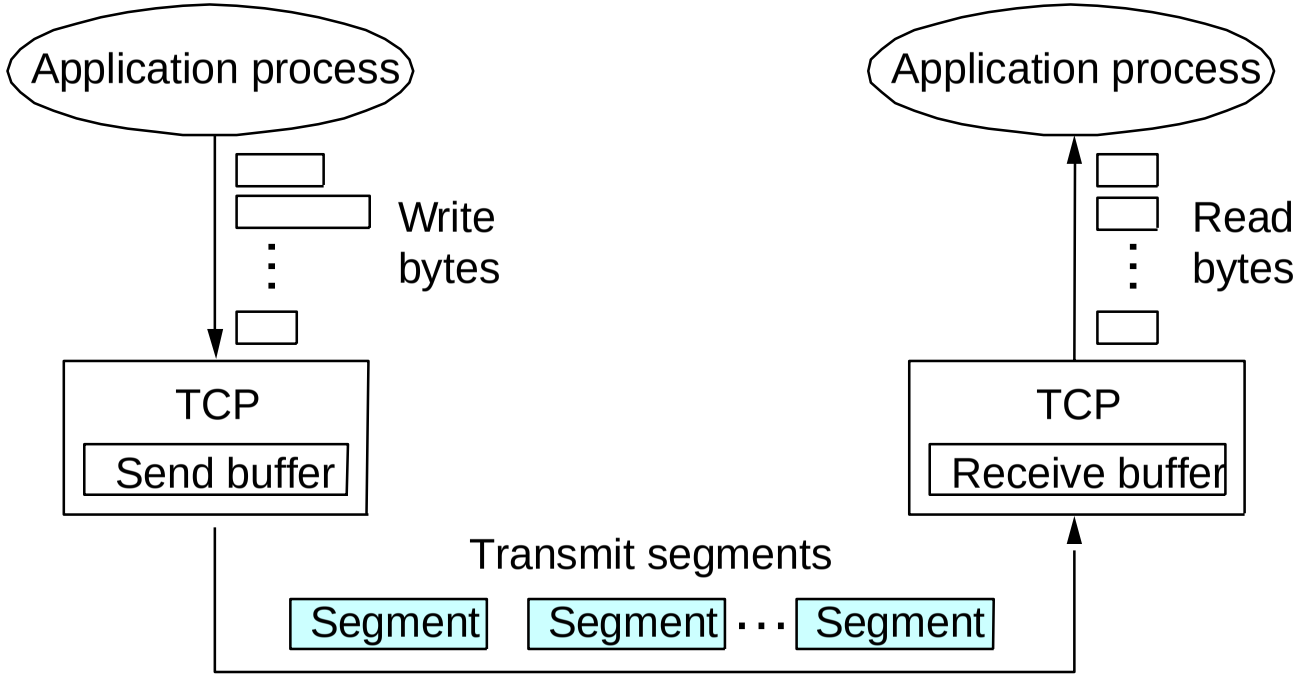
- UDP includes optional protection against errors
  - Checksum intended as an end-to-end check on delivery
  - So it covers data, UDP header, and IP pseudoheader



Transmission Control Protocol (TCP)

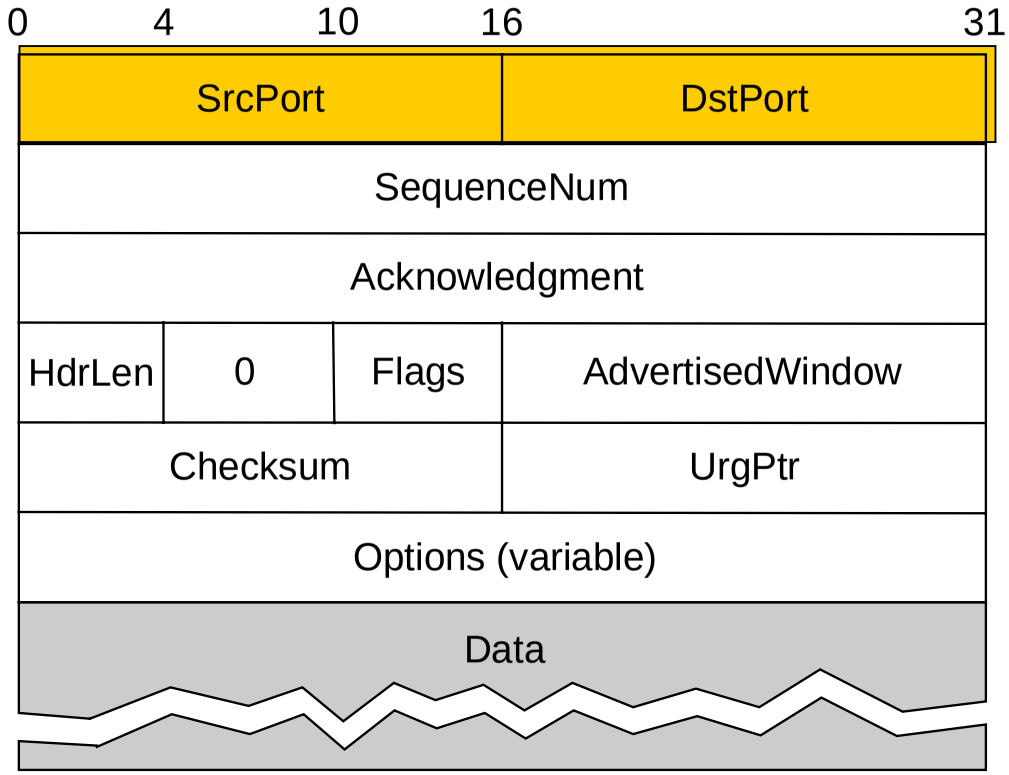
- Reliable bi-directional bytestream between processes
  - Message boundaries are not preserved
- Connections
  - Conversation between two endpoints with beginning and end
- Flow control
  - Prevents sender from over-running receiver buffers
- Congestion control
  - Prevents sender from over-running network buffers

TCP Delivery



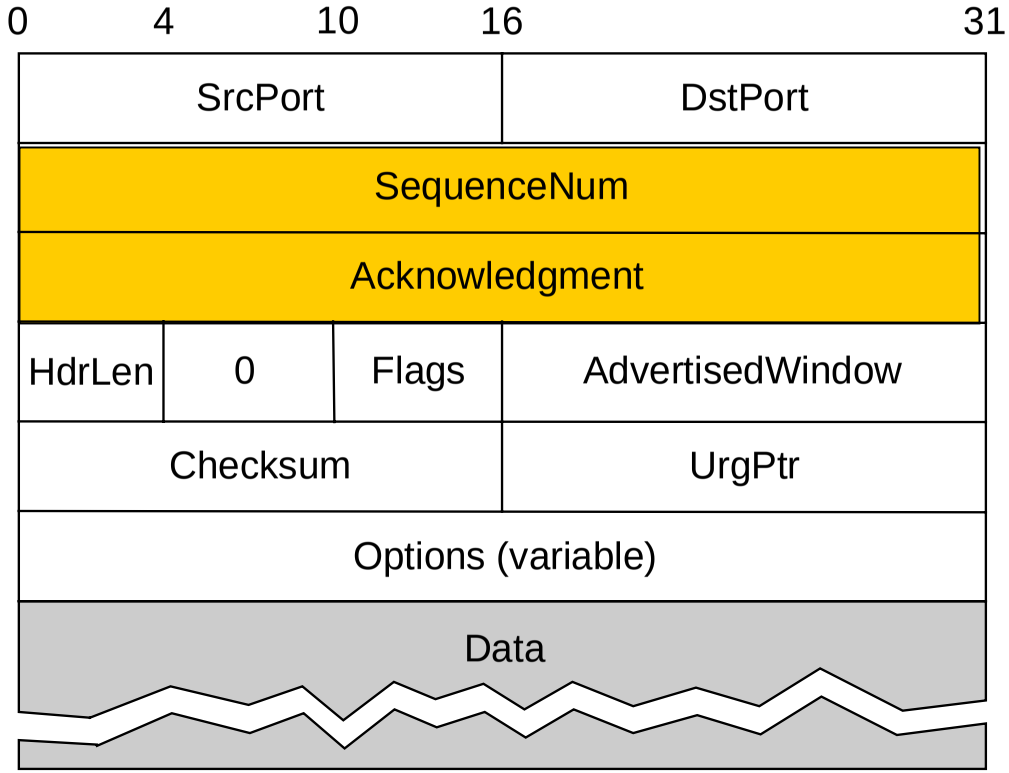
TCP Header Format

- Ports plus IP addresses identify a connection



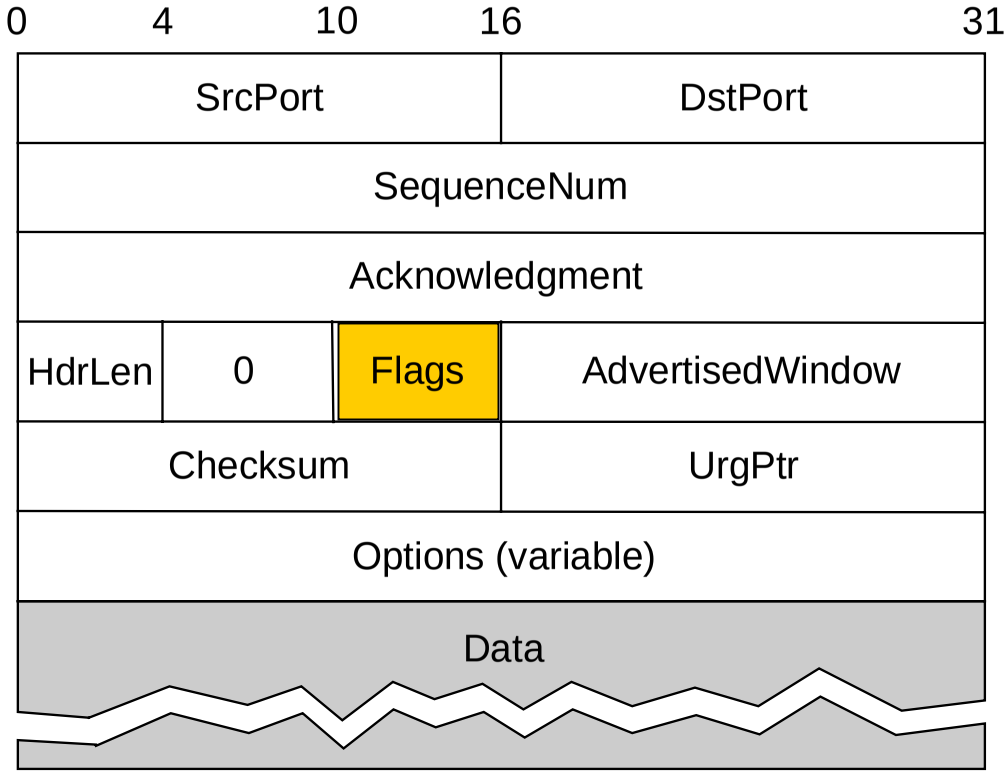
TCP Header Format

- Sequence and Ack numbers used for the sliding window
  - Congestion control works by controlling the window size



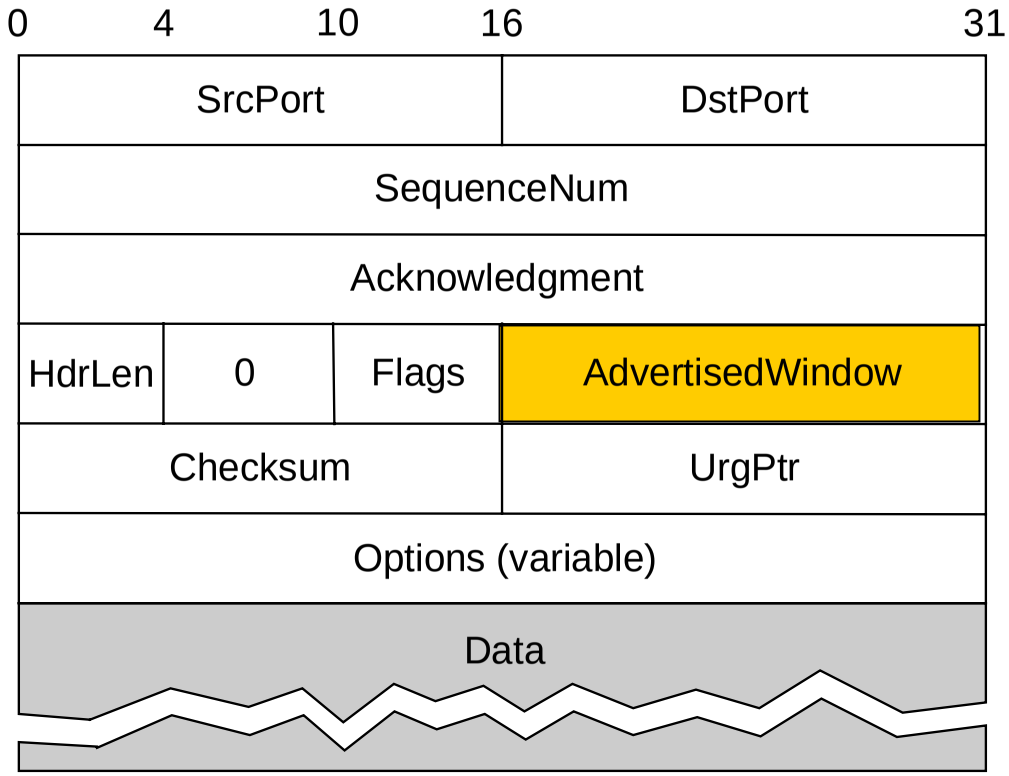
TCP Header Format

- Flags may be URG, ACK, PSH, RST, SYN, FIN



TCP Header Format

- Advertised window is used for flow control



Other TCP Header Fields

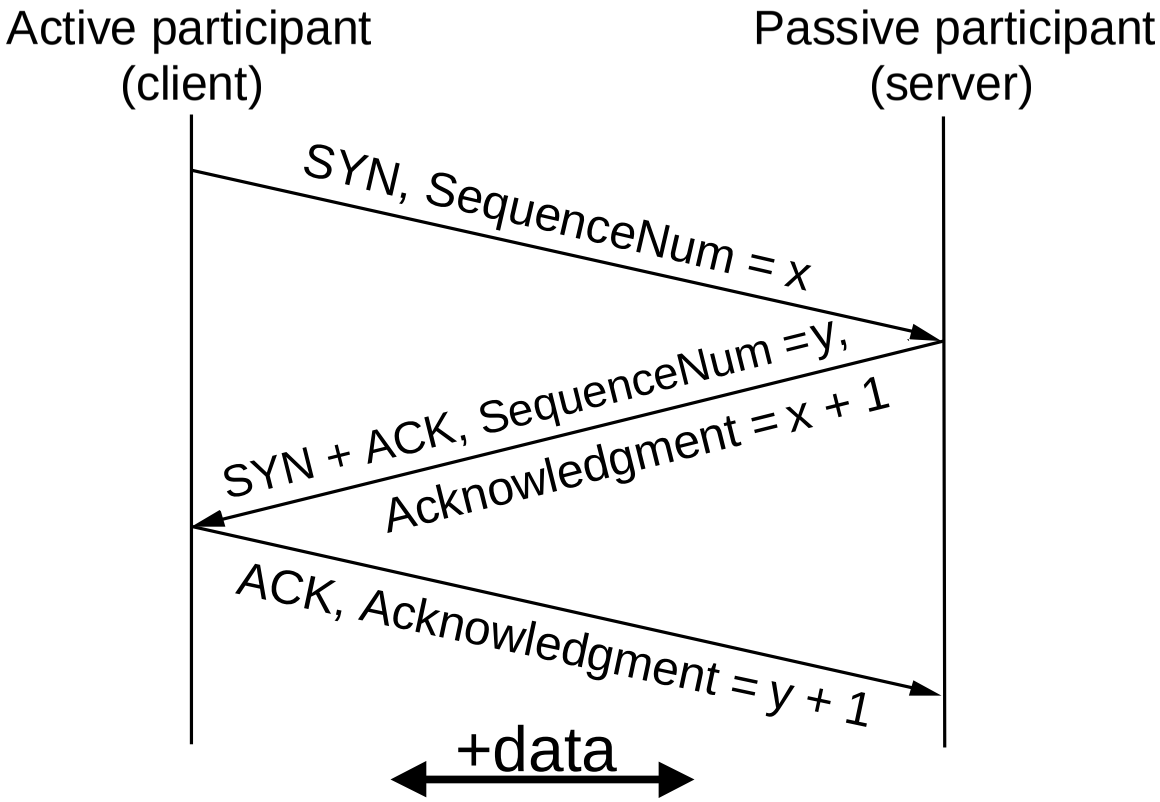
- Header length allows for variable length TCP header with options for extensions such as timestamps, selective acknowledgements, etc.
- Checksum is analogous to that of UDP
- Urgent pointer/data not used in practice
- Very few bits not assigned ...

Connection Establishment

- Both sender and receiver must be ready before we start to transfer the data
  - Sender and receiver need to agree on a set of parameters, e.g., the Maximum Segment Size (MSS)
- This is signaling
  - It sets up state at the endpoints
  - Compare to “dialing” in the telephone network
- In TCP a Three-Way Handshake is used

Three-Way Handshake

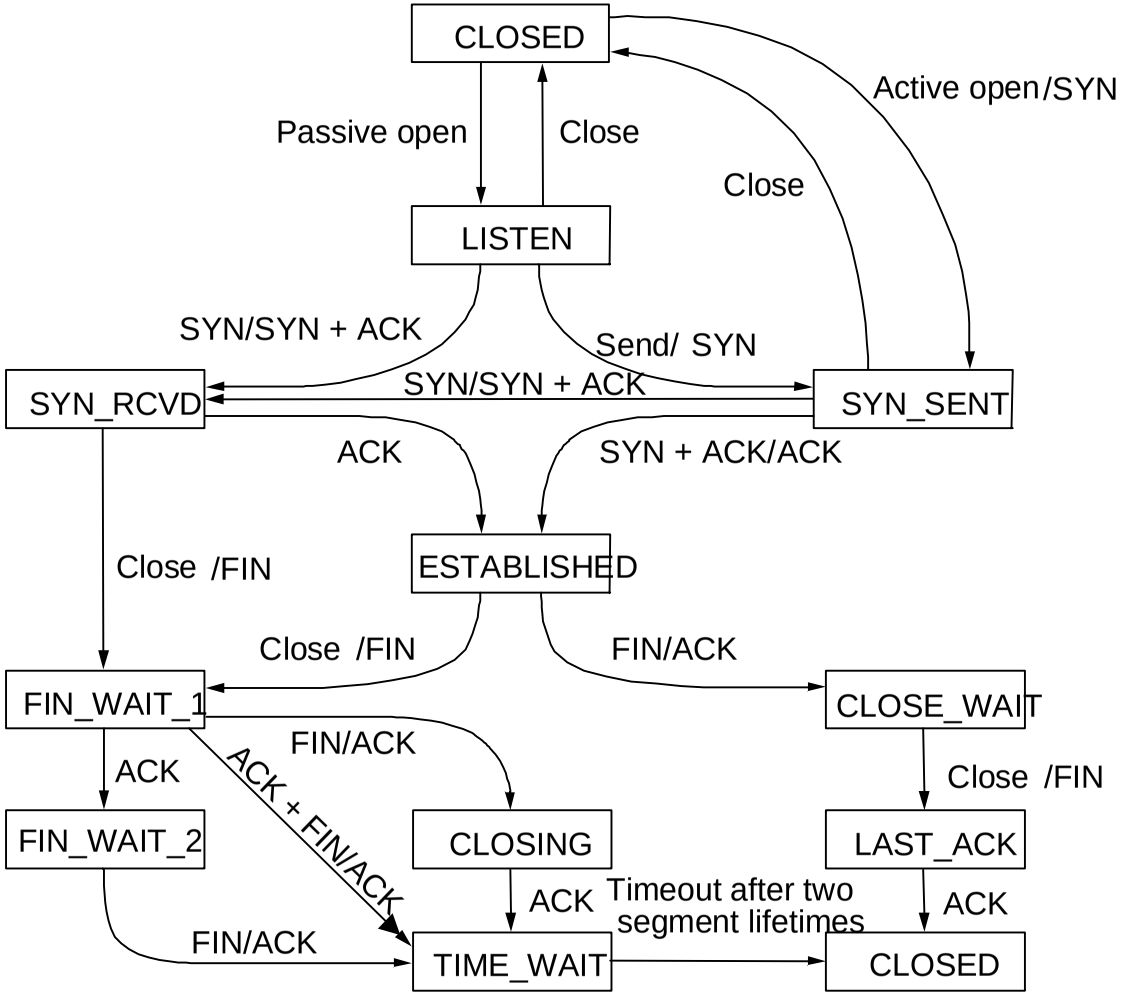
- Opens both directions for transfer



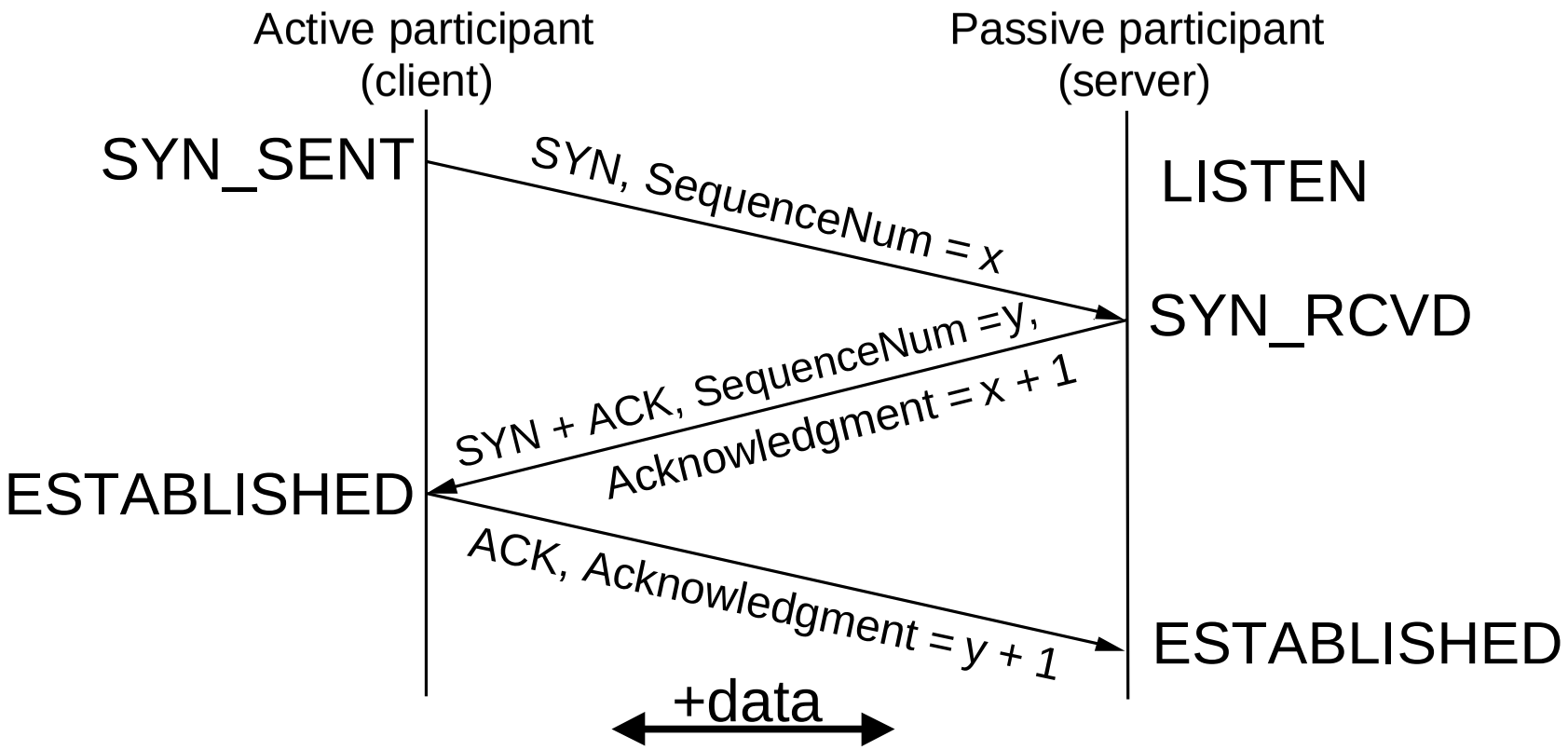
Some Comments

- We could abbreviate this setup, but it was chosen to be robust, especially against delayed duplicates
  - Three-way handshake from Tomlinson 1975
- Choice of changing initial sequence numbers (ISNs) minimizes the chance of hosts that crash getting confused by a previous incarnation of a connection
- But with random ISN it actually proves that two hosts can communicate
  - Weak form of authentication

TCP State Transitions



Again, with States

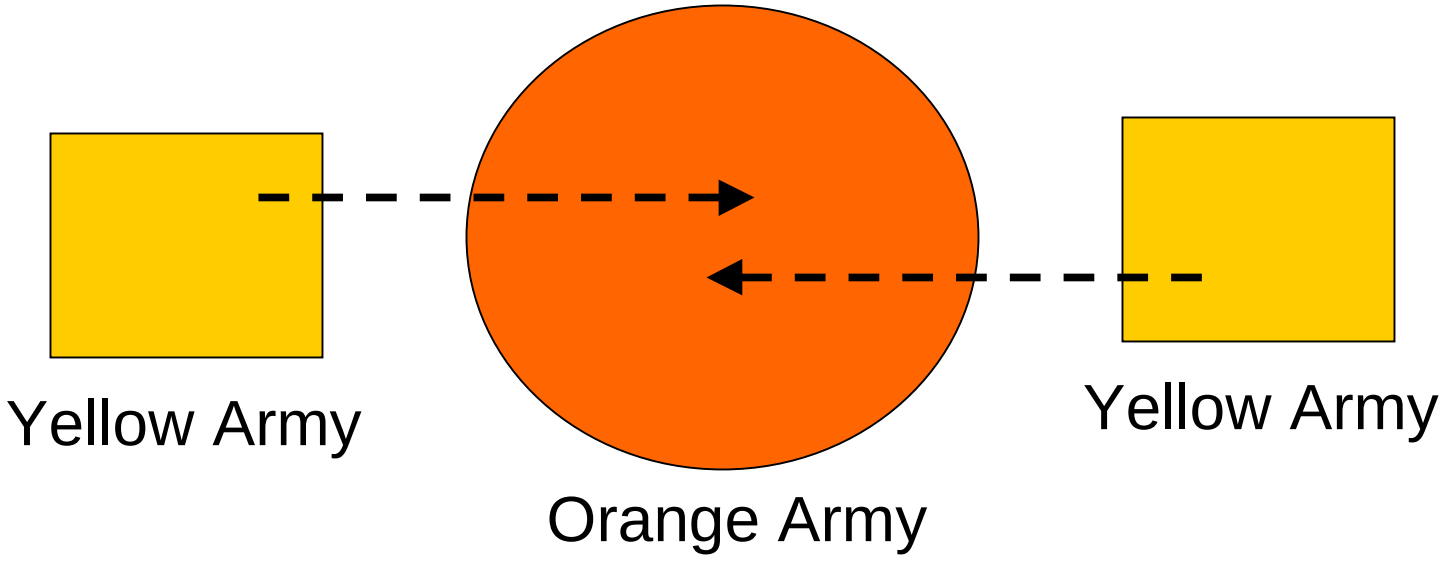


Connection Teardown

- Orderly release by sender and receiver when done
  - Delivers all pending data and “hangs up”
- Cleans up state in sender and receiver
- TCP connection teardown follows, but first an aside ...

The Two-Army Problem

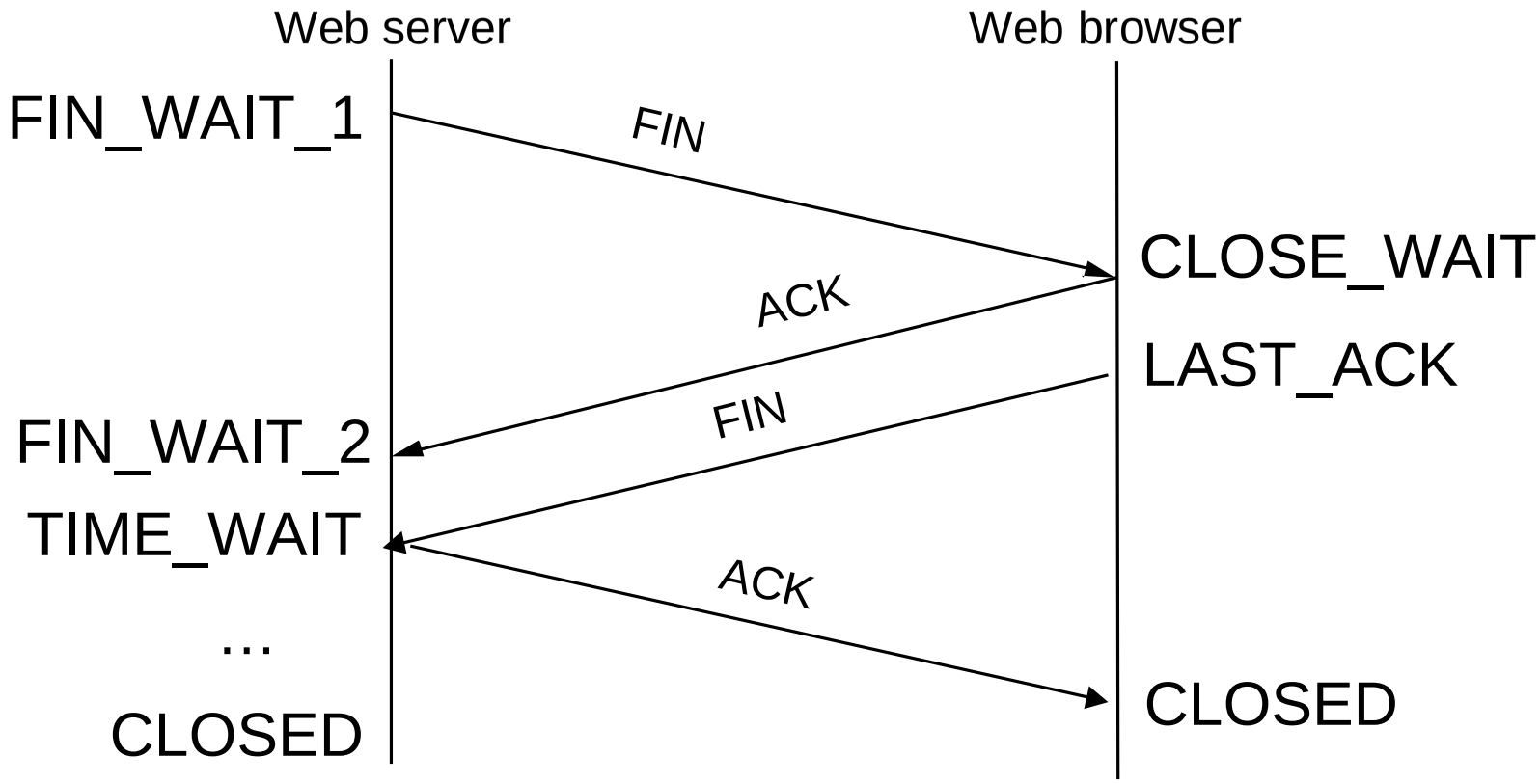
- Yellow armies want to synchronize their attacks to win
  - But their messengers might be captured by the orange army



It is impossible for both Yellow armies guarantee a joint attack!

TCP Connection Teardown

- Symmetric close – both sides shutdown independently



The TIME_WAIT State

- We wait 2MSL (two times the maximum segment lifetime of 60 seconds) before completing the close

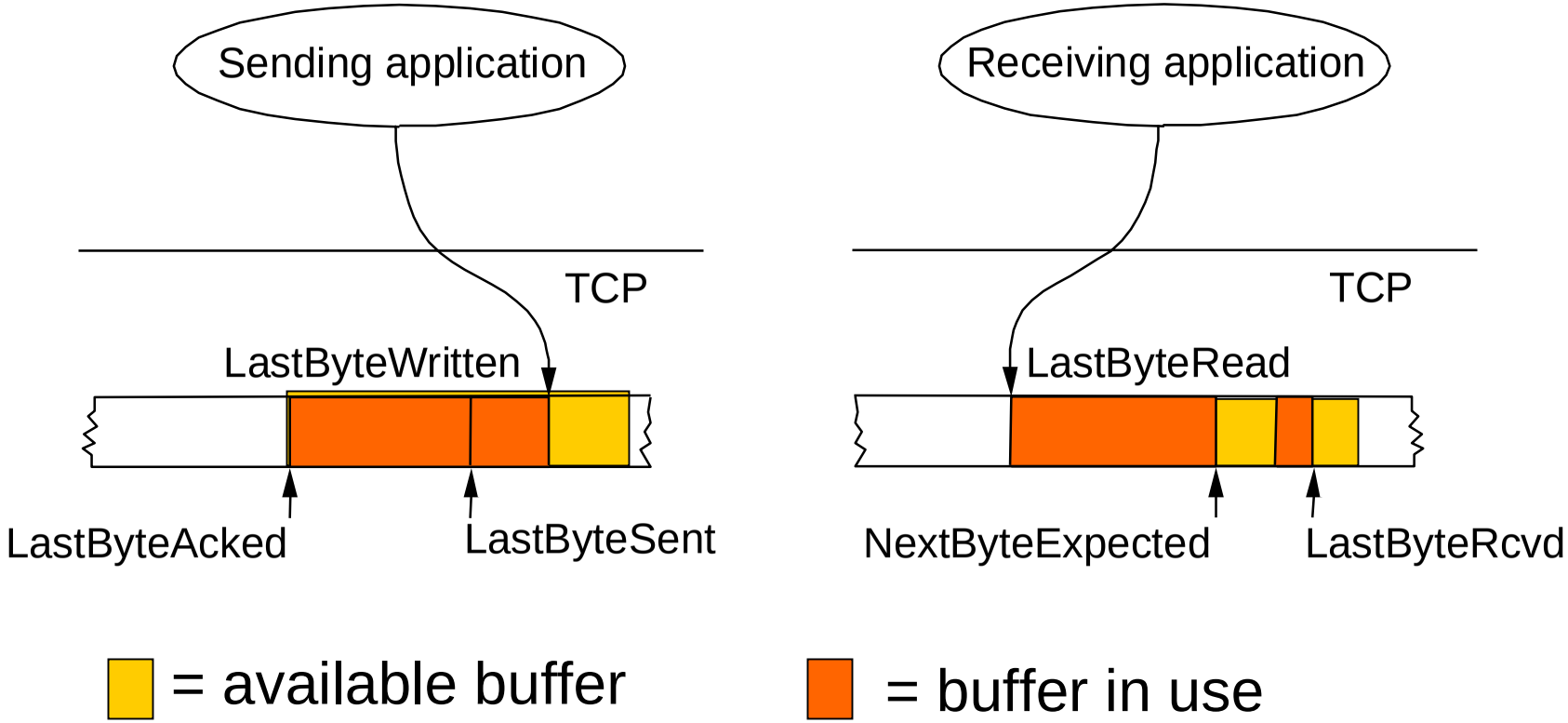
- Why?

- ACK might have been lost and so FIN will be resent
- Could interfere with a subsequent connection

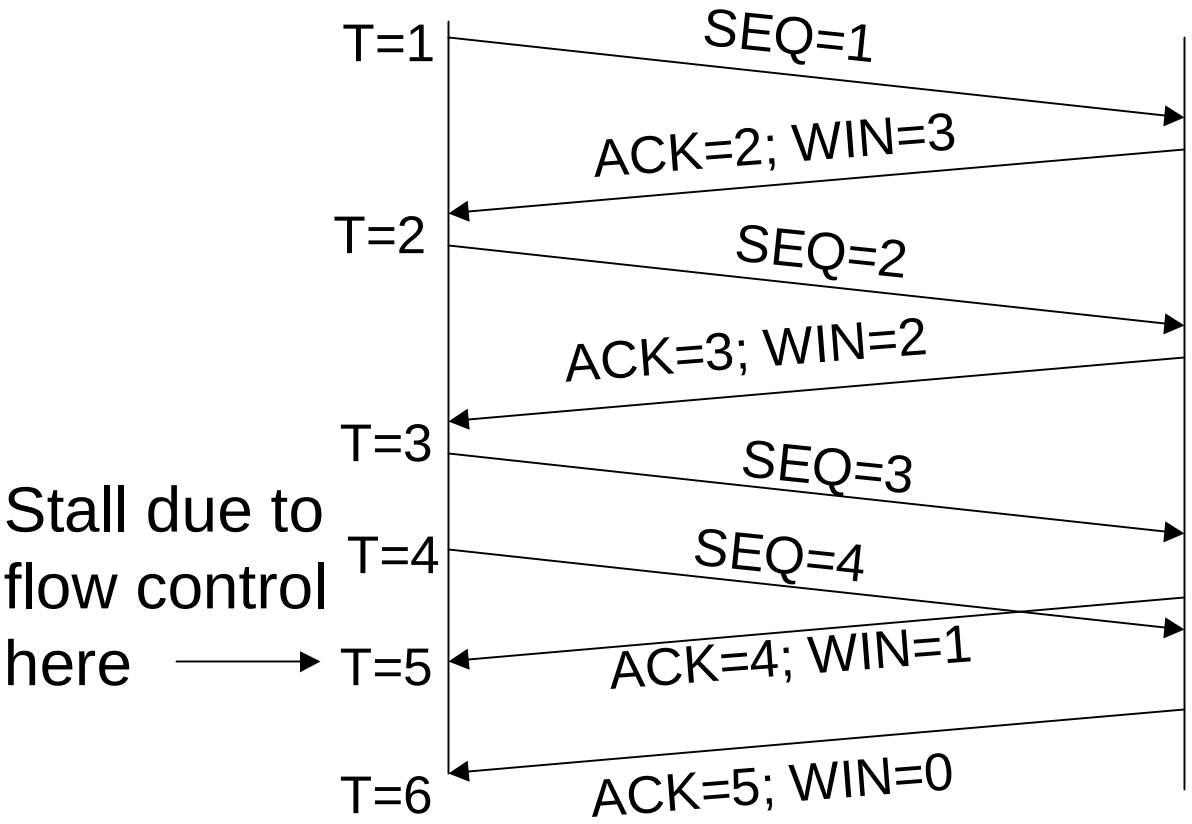
Flow Control

- Sender must transmit data no faster than it can be consumed by the receiver
  - Receiver might be a slow machine
  - App might consume data slowly
- Implement by adjusting the size of the sliding window used at the sender based on receiver feedback about available buffer space
  - This is the purpose of the Advertised Window field

Sender and Receiver Buffering

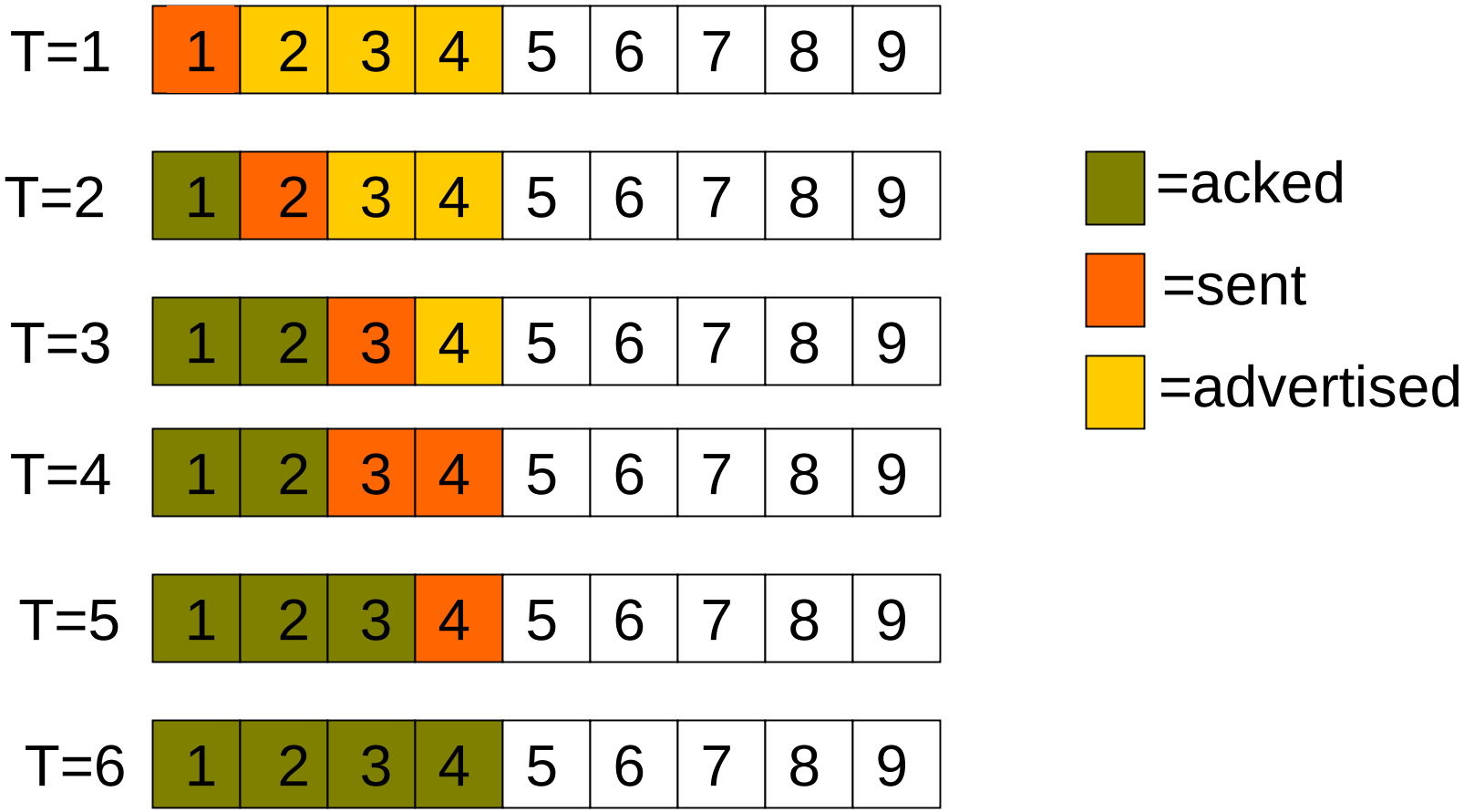


Example – Exchange of Packets



Receiver has buffer of size 4 and application doesn't read

Example – Buffer at Sender



Key Concepts

- We use ports to name processes in TCP/UDP
  - "Well-known" ports are used for popular services
- Connection setup and teardown complicated by the effects of the network on messages
  - TCP uses a three-way handshake to set up a connection
  - TCP uses a symmetric disconnect
- Flow control prevents sender over-running receiver
  - Implemented using an advertised window