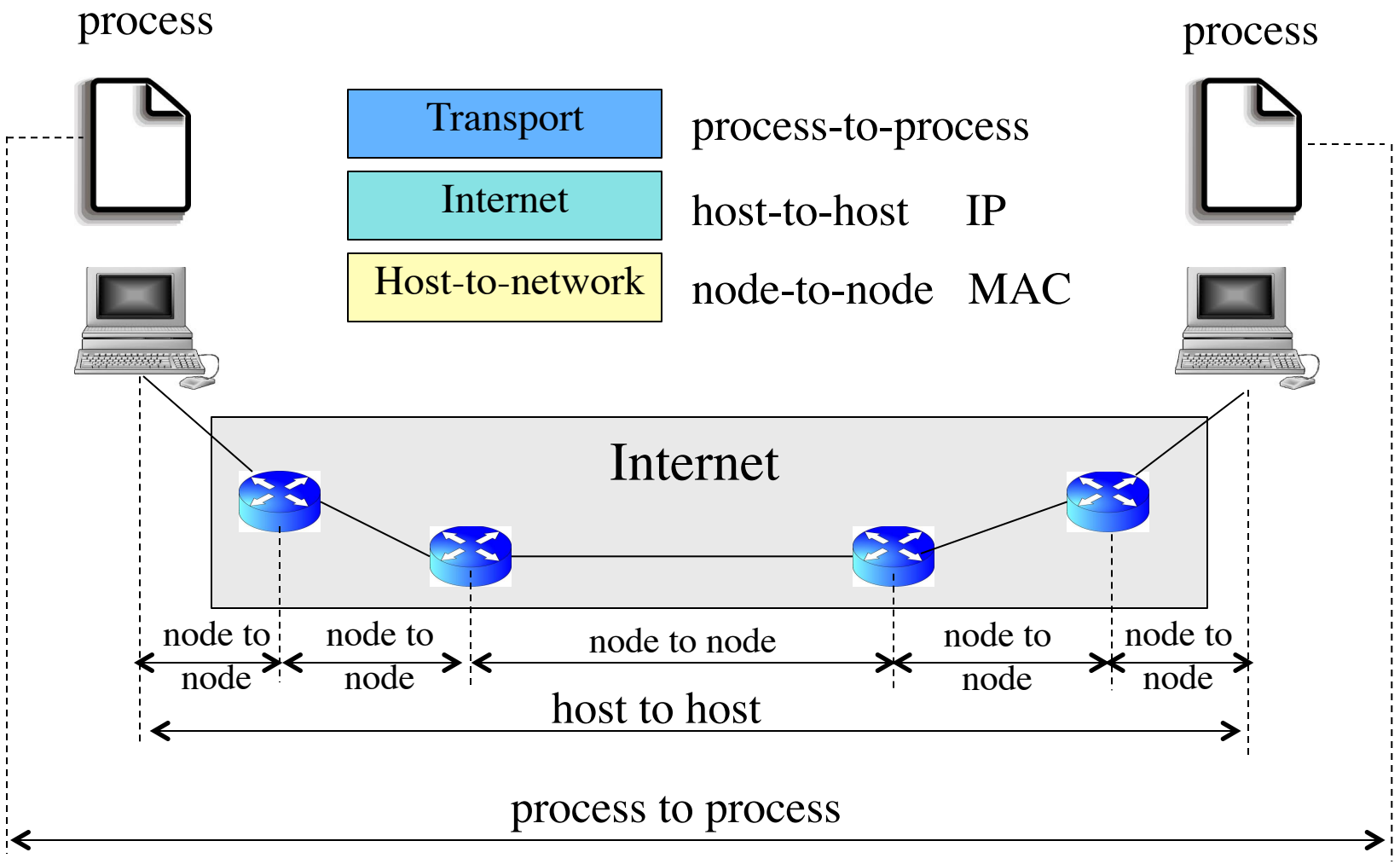
Lecture 19 Overview

- Last Lecture
  - Internet Protocol (2)
- This Lecture
  - Transport Control Protocol (1)
    - Generic transport layer
    - Connection management
  - Source: chapters 23, 24
- Next Lecture
  - Transport Control Protocol (2)
  - Source: chapters 23, 24

Type of Data Deliveries



Transport Layer

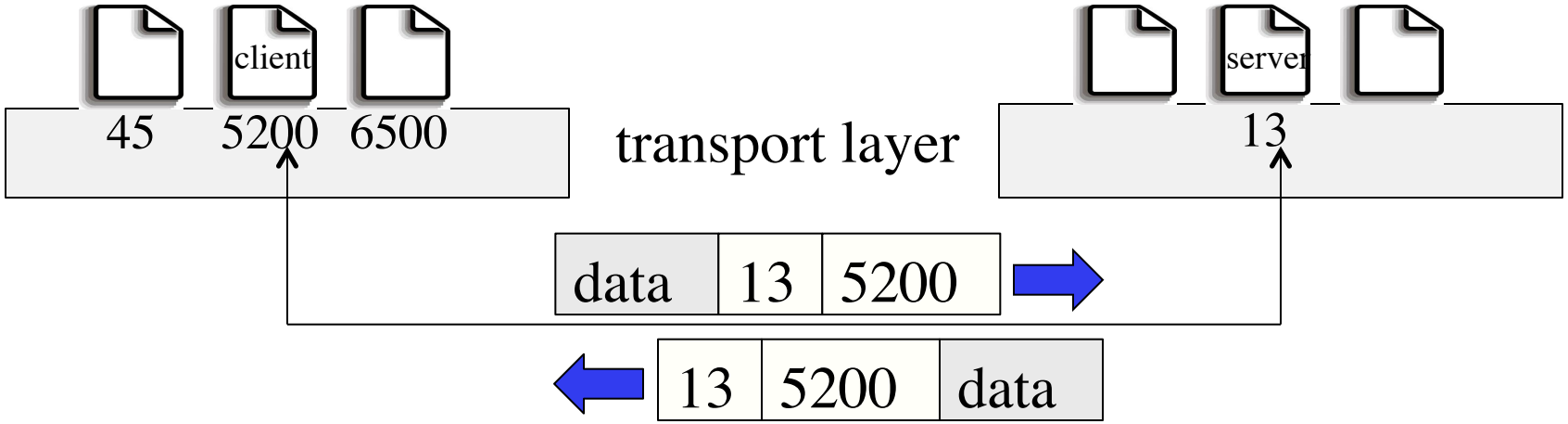
- The first to define end-to-end communications
  - Intermediate nodes deliver packets, but do not get involved in what goes on at the transport layer.
- Ensures that the whole message arrives intact and in order, overseeing both error control and flow control at the source-to-destination level.
- Types of transport services
  - connection-oriented/connectionless
  - reliable/unreliable

Transport Layer Functions

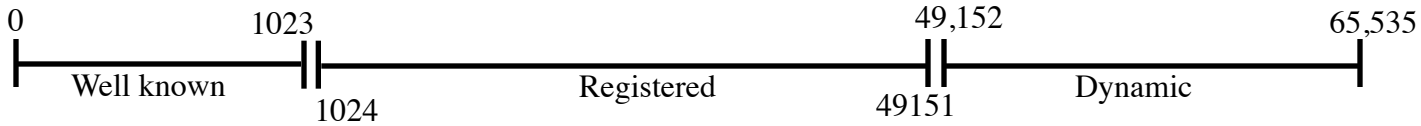
- Connection management
  - Setup/release connections
- Flow and congestion control
  - Provide flow and congestion control between end users
- Error detection
  - Guarantee error-free transmission between source and destination
- Quality-of-Service (QoS)
  - Reliability
  - Delay
  - Throughput

Transport Layer Address

- A transport layer address is called a **port number**
  - Used to choose among multiple processes on the same host
  - A port number is a 16-bit integer between 0 and 65535



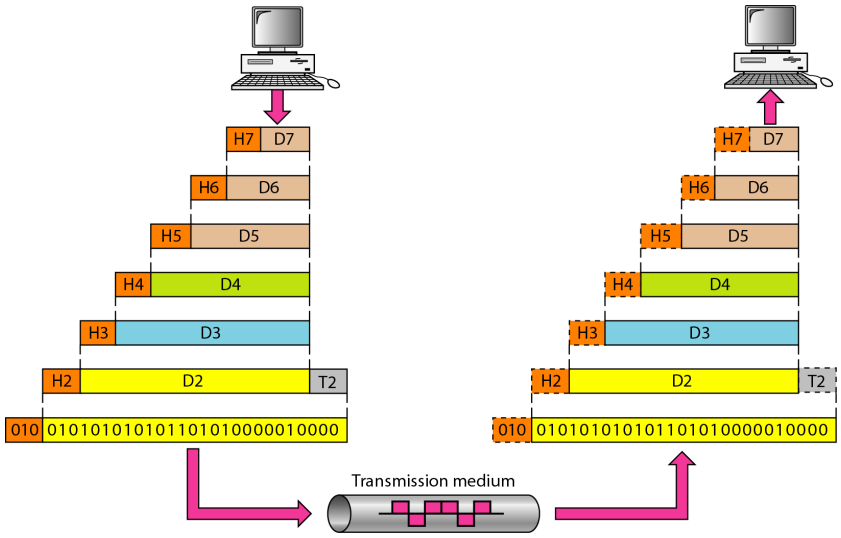
  - IANA (Internet Assigned Number Authority) has divided the port numbers into three ranges: well known, registered, and dynamic (private)

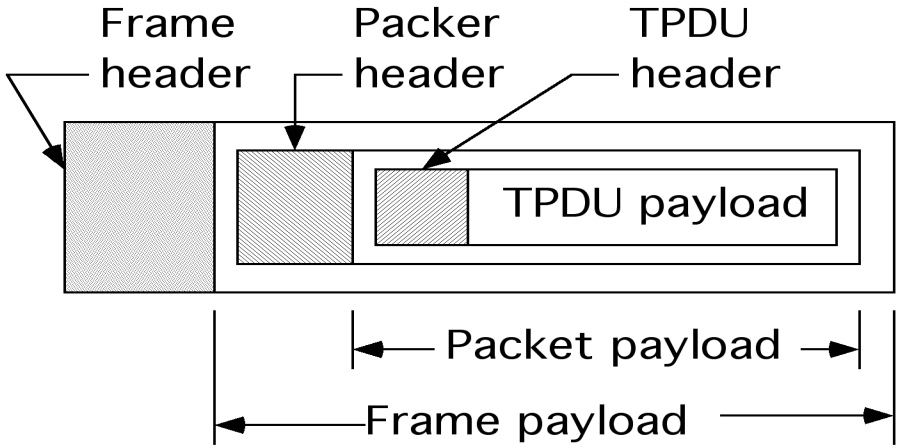


  - well-known numbers : SSH - 22, SMTP - 25, HTTP – 80, DNS – 53
    - See /etc/services on Linux

Overview of Transport Layer

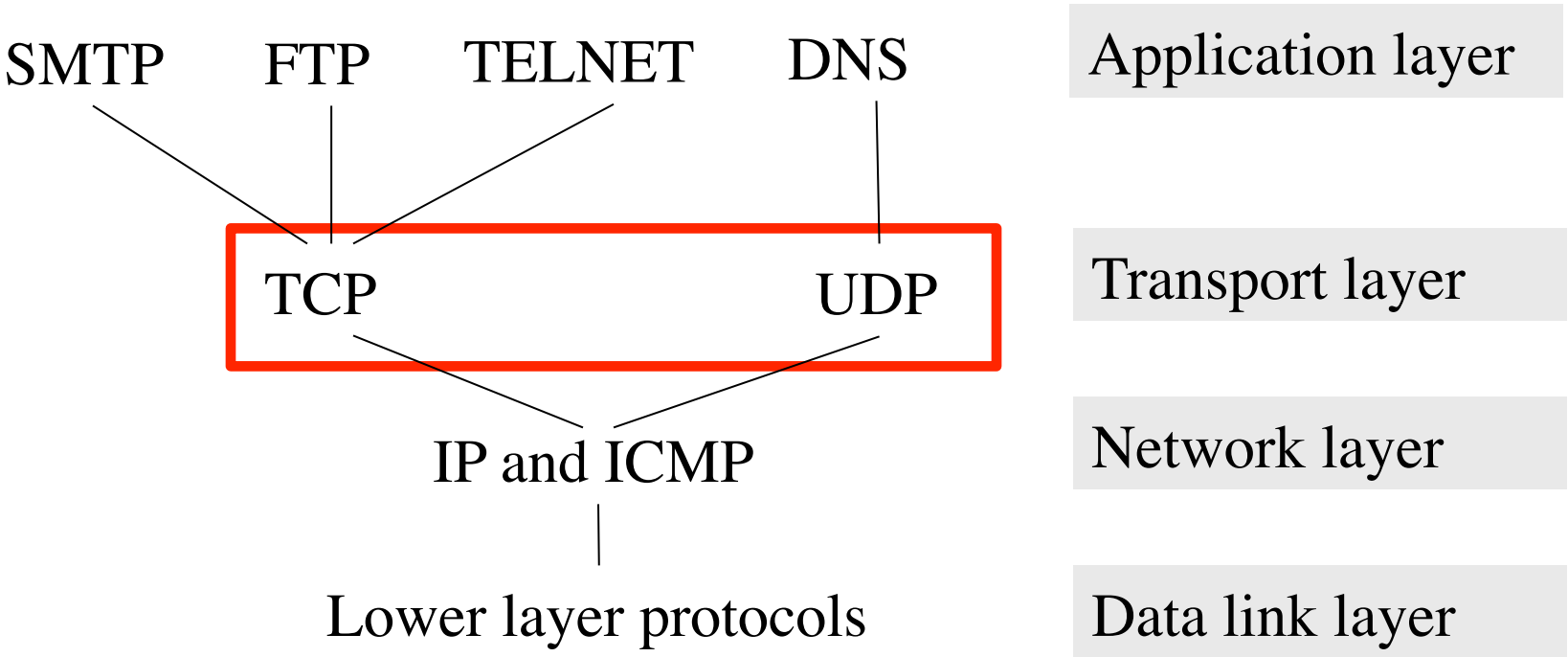
- TCP segment, IP packet, Data link frame





Transport Layer for the Internet

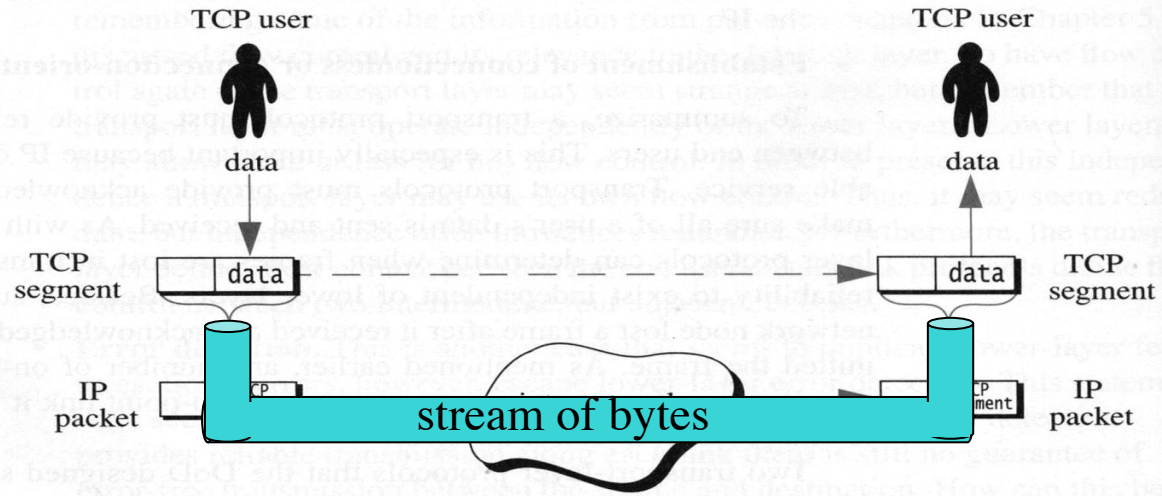
- Two protocols
  - TCP - Transport control protocol
  - UDP - User datagram protocol



Transport Control Protocol

- TCP is a **connection-oriented** protocol
  - TCP connections are full duplex and end-to-end.
  - TCP does not support multicasting or broadcasting.
- TCP is a **stream-oriented** protocol
  - Provides a reliable end-to-end byte stream over an unreliable network such as IP.
  - **Each byte** of data has a position in the stream.

Figure 7.42 TCP as a User-to-User Service

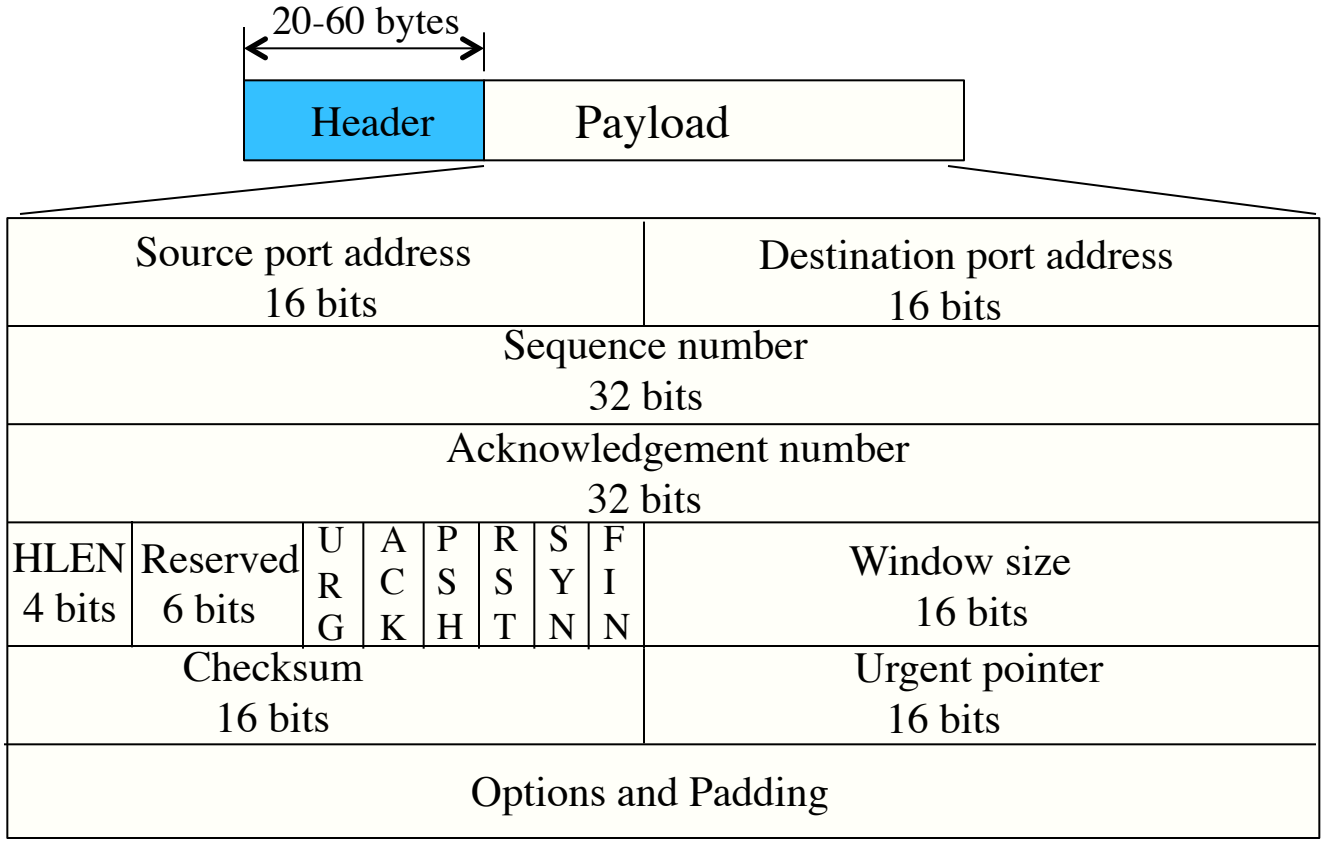


Transport Control Protocol (cont.)

- Each machine supporting TCP has a **TCP entity**, either a user process or part of the OS kernel that manages TCP streams and interfaces to the IP layer.
  - A TCP entity accepts user data streams from local processes, breaks them up into pieces not exceeding 64K bytes, puts each of them into a TCP segment which is put into the payload of an IP packet, which is then transmitted by IP network protocol.
  - When IP packets containing TCP segment arrive at a machine, they are given to the TCP entity, which reconstructs the original byte stream.

TCP Segment

- A packet in TCP is called a segment.
- A segment consists of a fixed 20- to 60-byte header, followed by zero or more data bytes

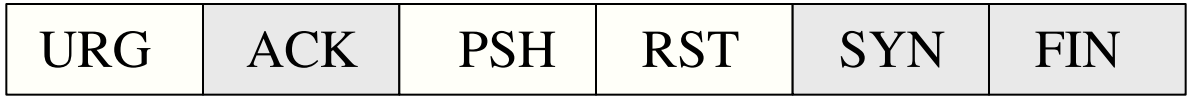


TCP Segment (cont.)

- Source port address (16 bits)
  - Specifies the application sending the segment.
- Destination port address (16 bits)
  - Identifies the application to which the segment is sent.
- Sequence number (32 bits)
  - In TCP each byte to be transmitted is numbered
  - This field defines contains the number assigned to the **first byte of data** in this segment.
- Acknowledgment number (32 bits)
  - Contains the byte number the receiving TCP entity **expects to receive next.**

TCP Segment (cont.)

- Header length (4 bits) - the size of the TCP header.
- Flags (6 bits)



URG: urgent pointer is valid
ACK: acknowledgement is valid
PHS: request for push
RST: reset the connection
SYN: synchronize sequence numbers
FIN: terminate the connection

- Window size (16 bits)
  - Tells the receiving TCP entity how many data bytes the sending TCP entity can accept.

TCP Segment (cont.)

- Checksum (16 bits)
  - Used for transport layer error detection.
- Urgent pointer (16 bits)
  - Signal the receiver to deliver the data to the higher layer as quickly as possible.
- Options/padding

How large can a TCP segment be?

Limited by

- 65,536 (64K) bytes of the IP payload
- MTU of the network

TCP Connection

- TCP is connected-oriented
  - TCP connection is virtual, not physical
  - All segments of the same message are send over the same virtual path that is chosen during connection set up.
- TCP connection-oriented communication requires three phases:
  - Connection establishment
  - Data transfer
  - Connection termination

Connection Establishment

- Simple connection (two-way handshaking)
  - Entity 1 send a TPDU to entity 2, saying "good morning, I would like to talk with the process with port number *pn*."
  - Entity 2 receives the TPDU and asks the process if it will accept the request.
  - If the process agrees, entity 2 sends a TPDU saying "ok, you can talk now", and connection is established.
- Problem with establishing a connection occurs when the network can lose, store, and duplicate packets.

Connection Establishment (cont.)

- Consider the following scenario
  - A user establishes a connection with a bank
  - Sends messages telling the bank to transfer a large amount of money to the account of a not entirely trustworthy person
  - And then releases the connection
- What happens if each packet in the above process is duplicated and stored in the network?

Connection Establishment (cont.)

- Three-way handshaking
  - Entity 1 sends a SYN segment, in which only the SYN flag is set, to synchronize with Entity 2.

The SYN segment cannot carry data, but consumes one sequence number.

  - Entity 2 sends a SYN+ACK segment, with 2 flag bit set: SYN and ACK. This segment has a dual purpose: acknowledge the received SYN segment and synchronize with Entity 1.

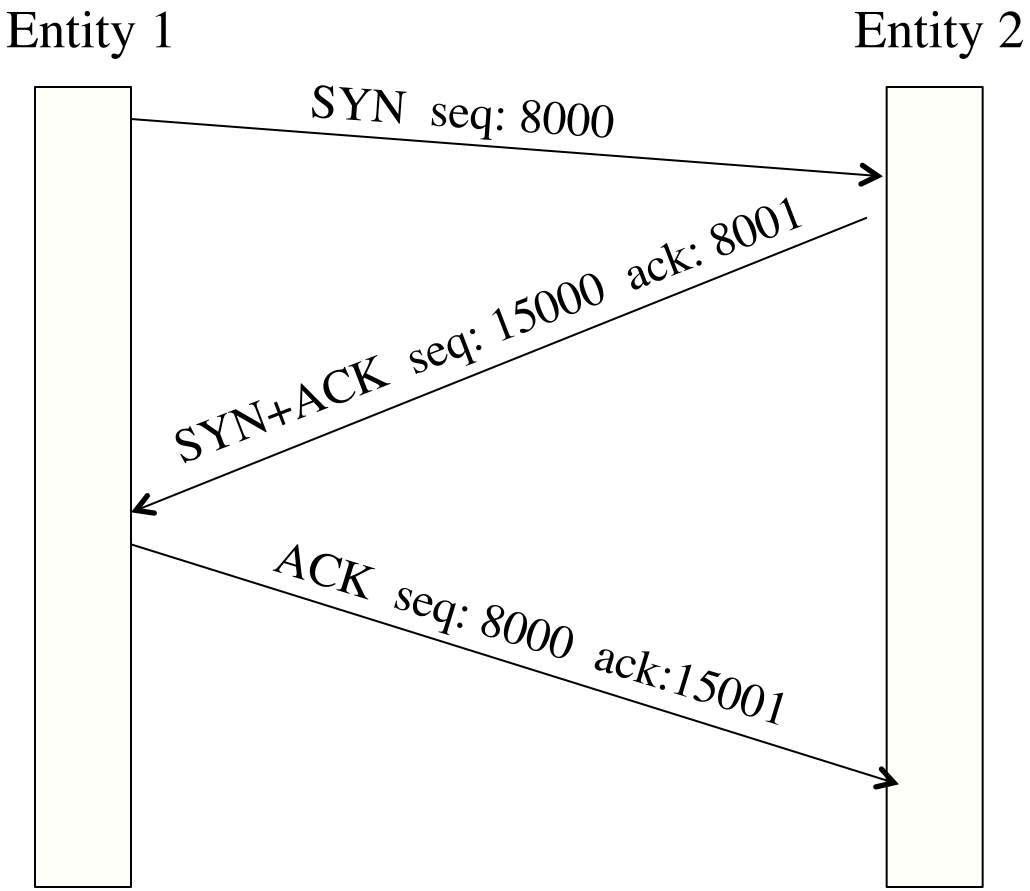
The SYN +ACK segment cannot carry data, but consumes one sequence number.

  - Entity 1 sends an ACK segment to acknowledge the receipt of the SYN+ACK segment.

An ACK segment, if carrying no data, consumes no sequence number.

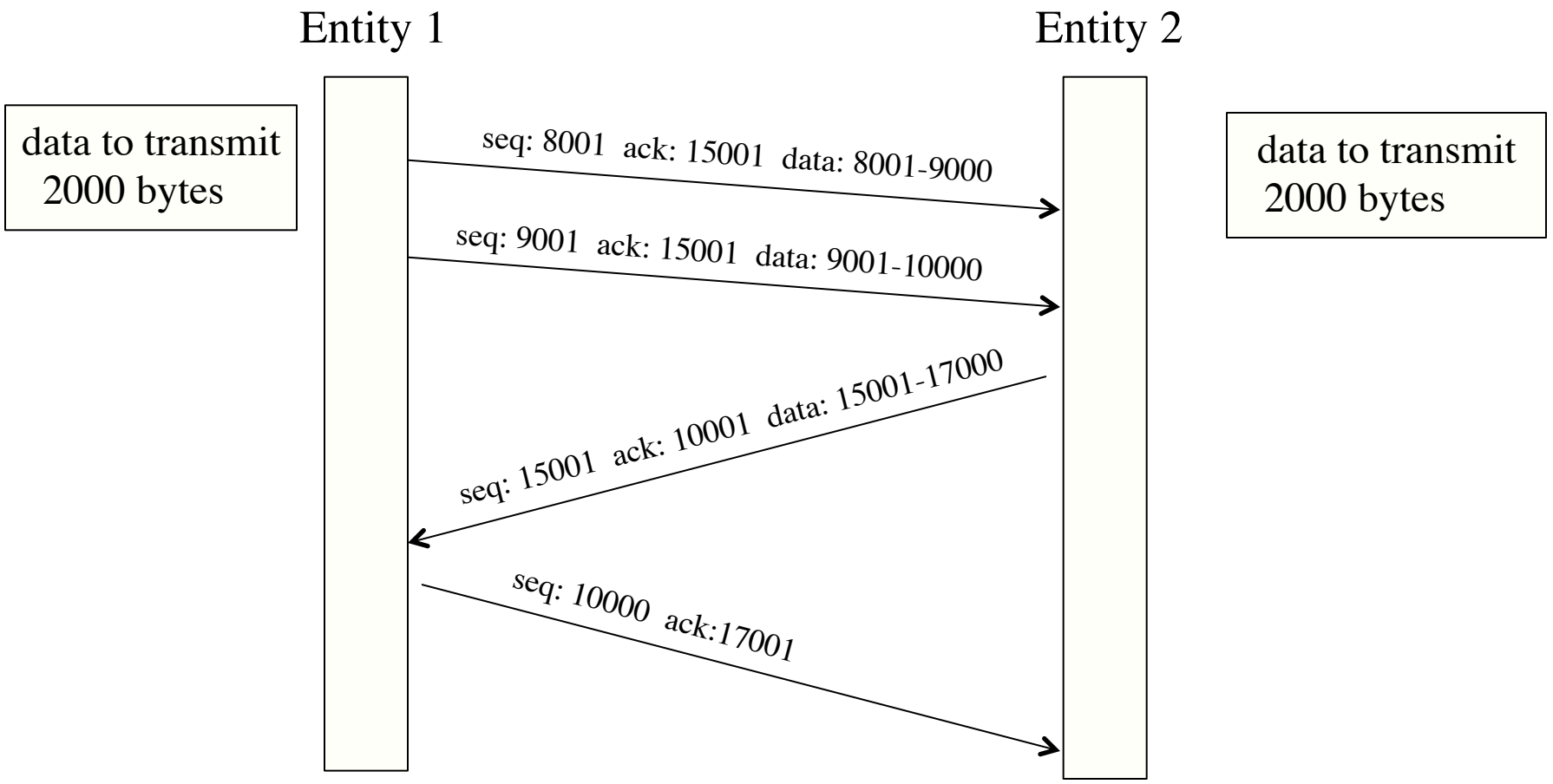
Connection Establishment (cont.)

Three-way handshaking



Data Transfer

- Acknowledgement can be piggybacked with the data.



Connection Termination

- Three-way handshaking
  - Entity 1 wants to terminate the connection, and sends a FIN segment. A FIN segment can include the last chunk of data sent by Entity 1.

The FIN segment consumes one sequence number if it does not carry data.

  - After receiving the FIN segment, Entity 2 sends a FIN+ACK segment to confirm the receipt of the FIN segment and to announce the close of the connection from its side. This segment can also contain the last chunk of data sent by Entity 2.

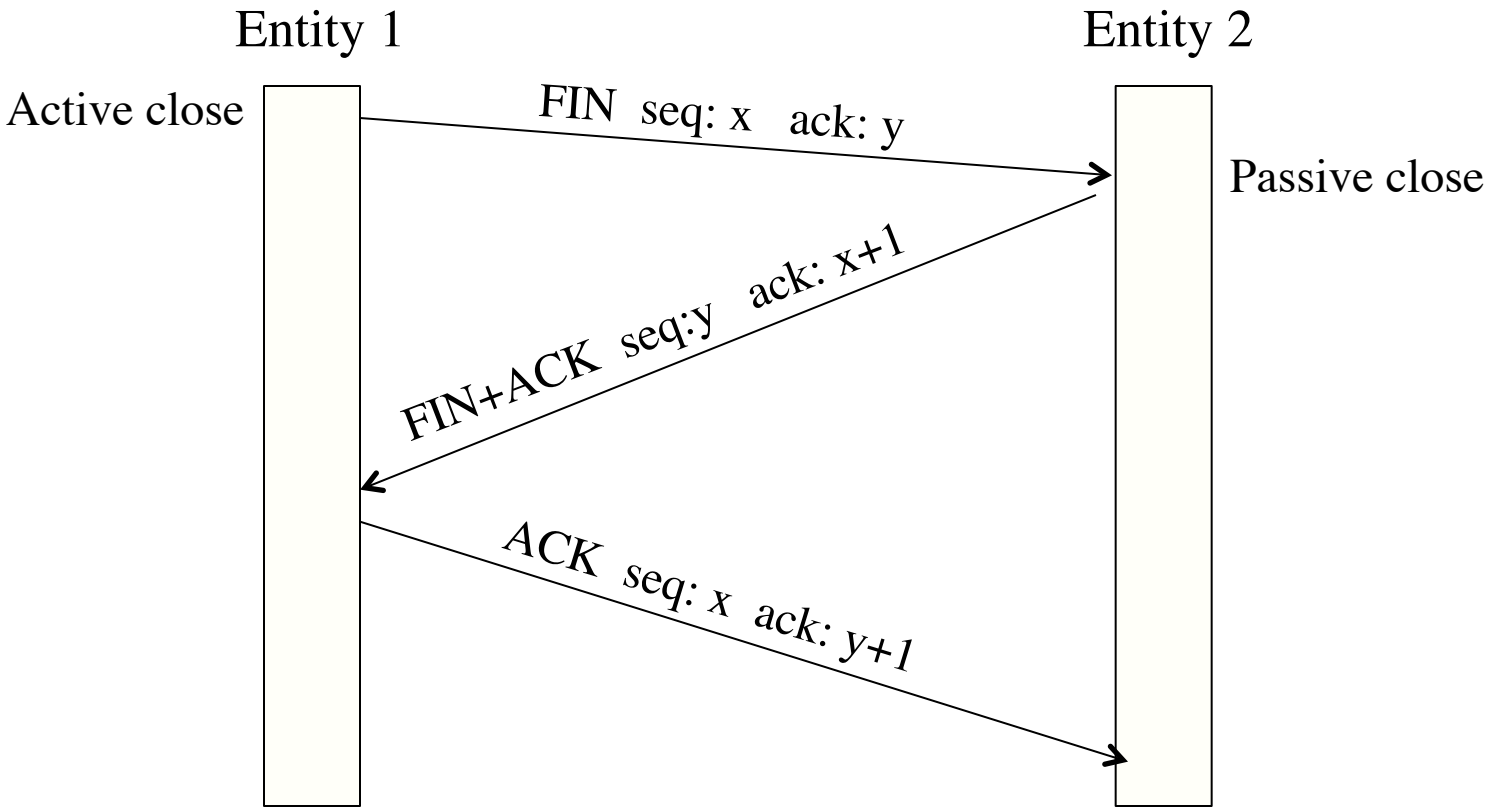
The FIN +ACK segment consumes one sequence number if it does not carry data.

  - Entity 1 sends an ACK segment to acknowledge the receipt of the FIN+ACK segment.

An ACK segment does not consume any sequence number.

Connection Termination

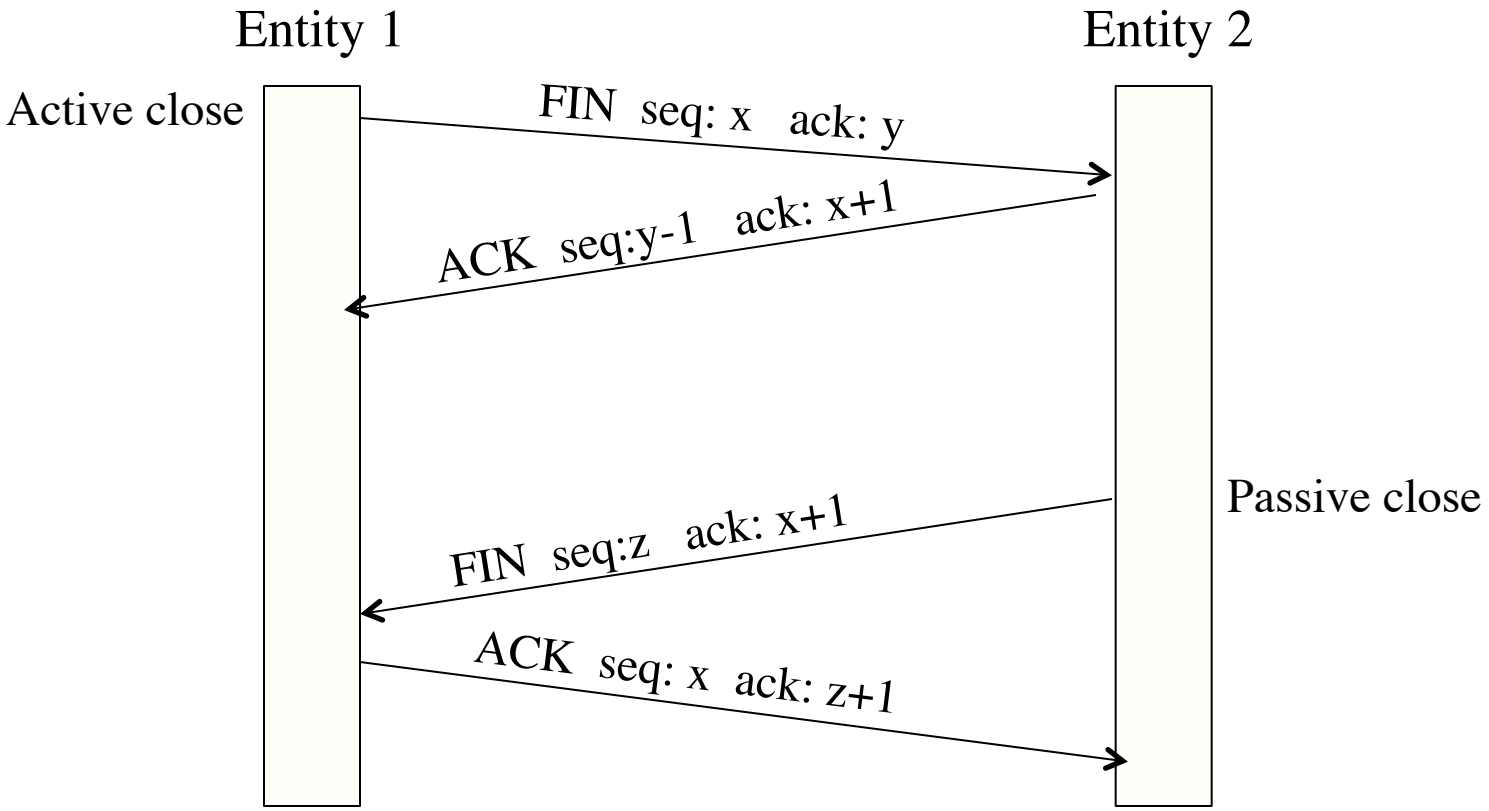
Three-way handshaking connection termination



Problem: This approach can result in data loss.

Connection Termination (cont.)

- Half-close: the connection in each direction is released independently of the other.



Questions to ponder

- Can the three-way handshaking scheme solve the problem caused by duplicate segments?

- Does the half-close approach always work? What about the FIN and ACK segments for disconnection request are lost?

Summary

- Concepts
  - Transport layer, Transport entity, Transport address, TPDU, QoS, Two-army problem
- Differences between transport layer and network layer
- Connection establishment and release
  - three-way handshake protocol for connection establishment
  - Three-way handshake combined with timer for connection release