# Transport Layer

# Process-to-Process Delivery: UDP, TCP

# our goals:

- Understand principles behind transport layer services:
  - Process to Process delivery
  - multiplexing, demultiplexing
  - Error control
  - congestion control
- learn about Internet transport layer protocols:
  - UDP: connectionless transport
  - TCP: connection-oriented reliable transport
  - TCP congestion control

# PROCESS-TO-PROCESS DELIVERY

***The transport layer is responsible for realiable process-to-process delivery (the delivery of a packet, part of a message, from one process to another)***

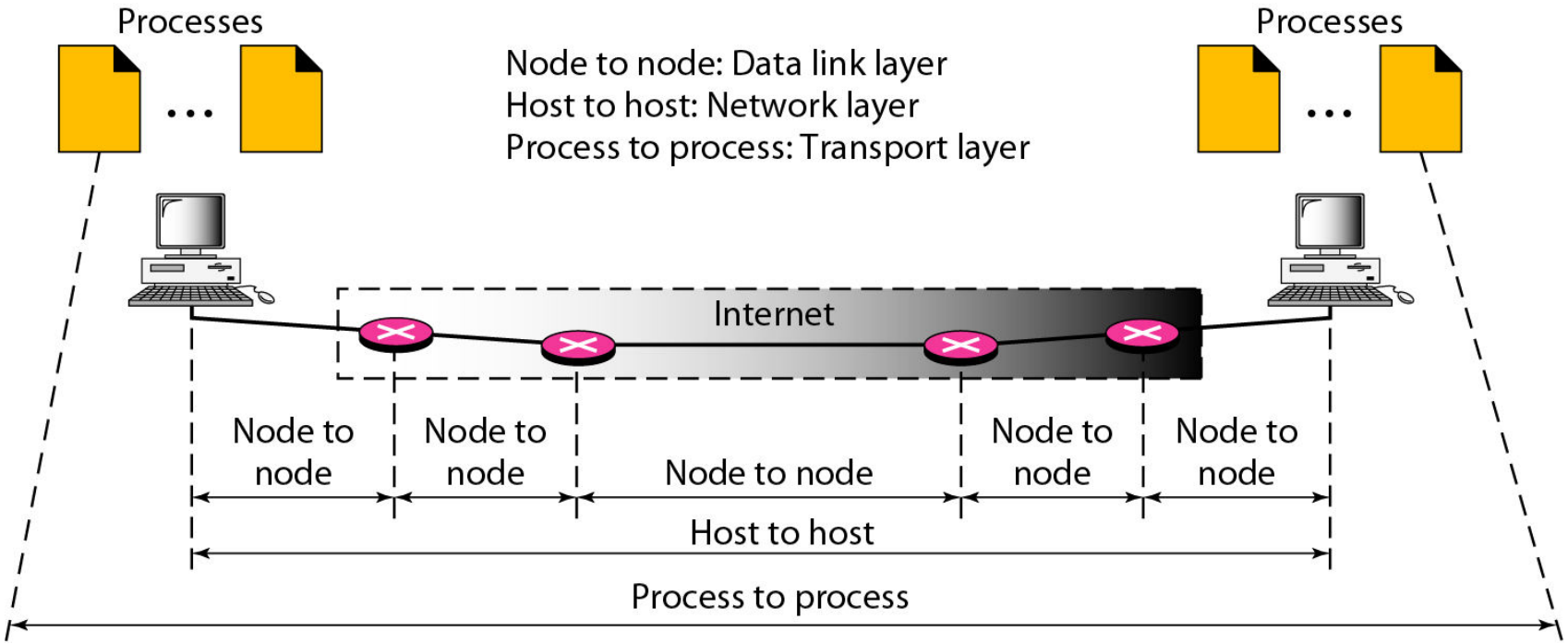
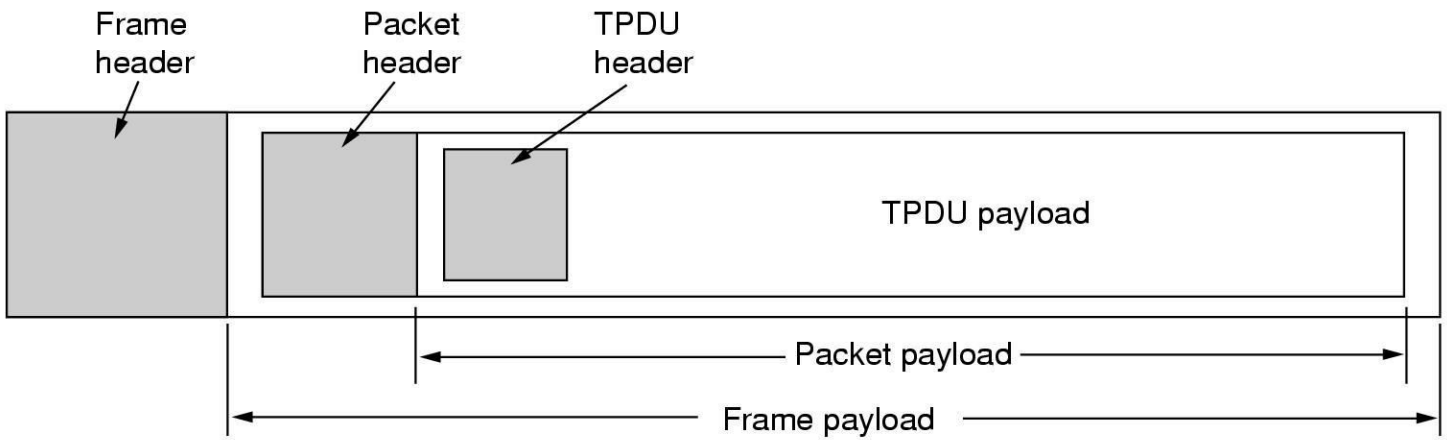



**Figure** *Types of data deliveries*

# Nesting of TPDU, Packets, and Frames




- Transport Protocol Data Unit (TPDU)
  - Term used for transport entity to transport entity messages
- TPDUs are contained in packets exchanged by network layer
- Packets are contained in frames exchanged by data link layer

# Processes:

- Process is an instance of a program in execution.
- Processes on two hosts communicate with each other by **sending and receiving messages.**
- The process receives messages from, and sends messages into the network through its **socket.**

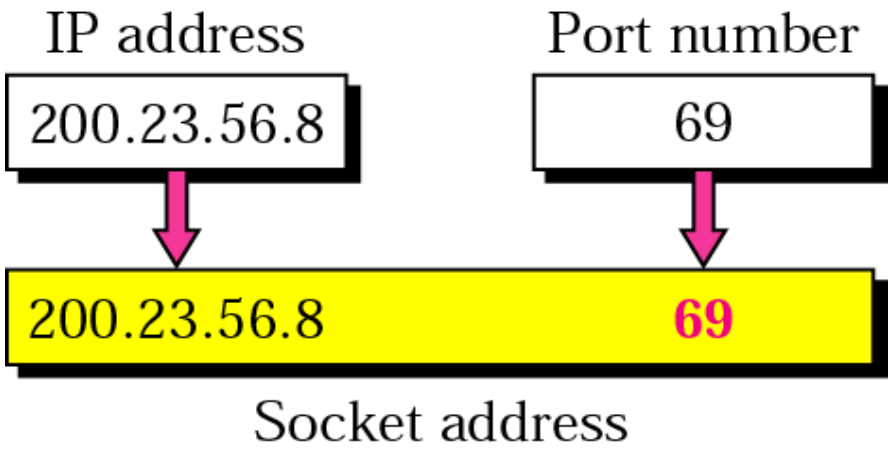
# Socket address

Transport layer at the receiving host delivers data to **the socket**

There should be **a unique identifier** for each socket.

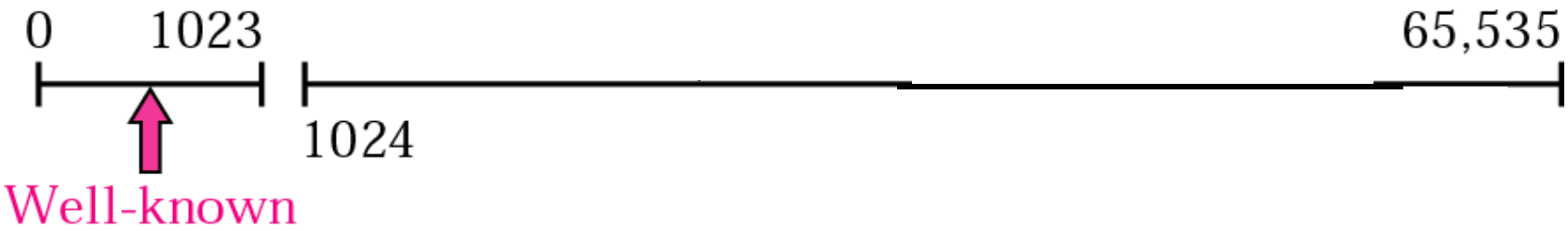
Socket identifier is called **socket address**

Socket address = IP address & Port number



## Figure 22.4 IANA ranges

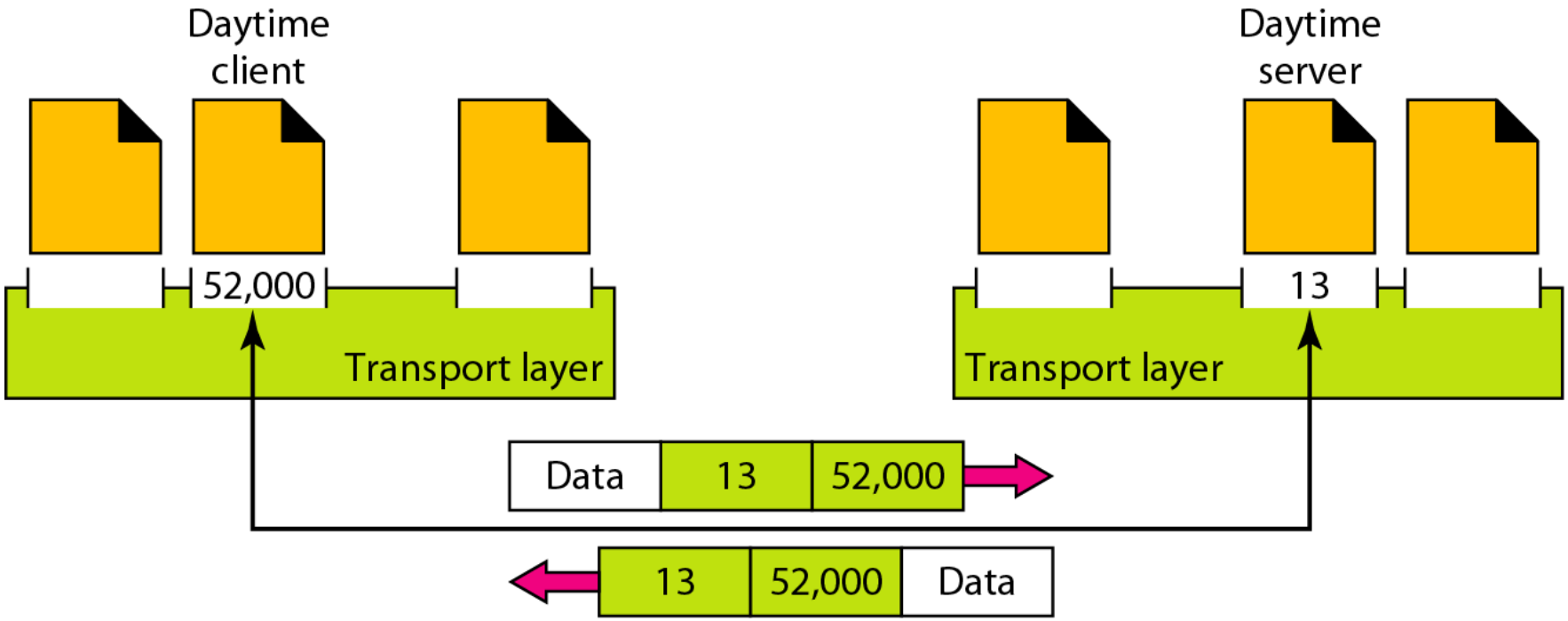
**Port numbers are 16-bit integers between 0 - 65535**




**The IANA divide the port numbers into three ranges**

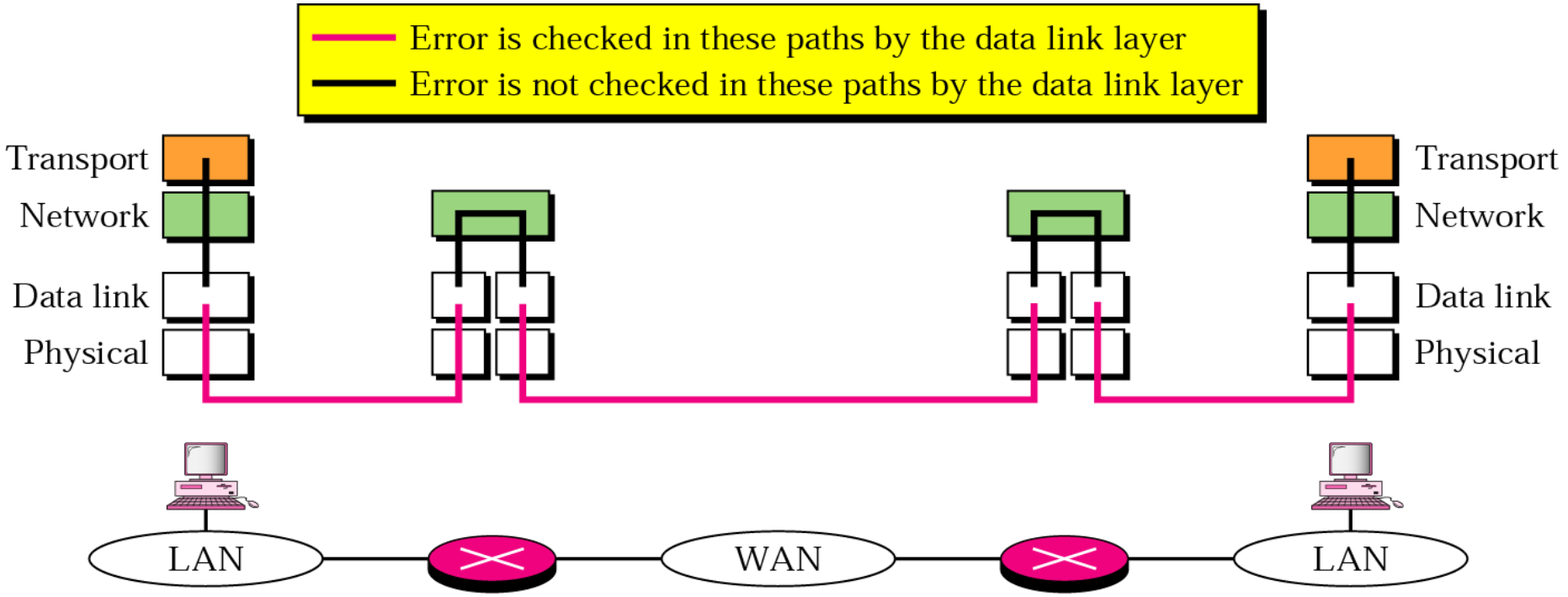
- **Well-known**: Assigned and controlled by **Internet Assigned Numbers Authority IANA** for example: **FTP 20,21, TELNET 23, SMTP 25, HTTP 80**
- **Registered ports:** The ports ranging from 1024 to 49,151 are not assigned or controlled by lANA.
- **Dynamic ports:** The ports ranging from 49,152 to 65,535.

## Figure 23.2 *Port numbers*



## Figure 23.7 Error control

**Why we need error control at the transport layer?**



# Transport Layer Protocols

- The Internet supports two transport layer protocols:
  - The Transport Control Protocol (**TCP**) for reliable service
  - The Unreliable (User) Datagram Protocol (**UDP**)

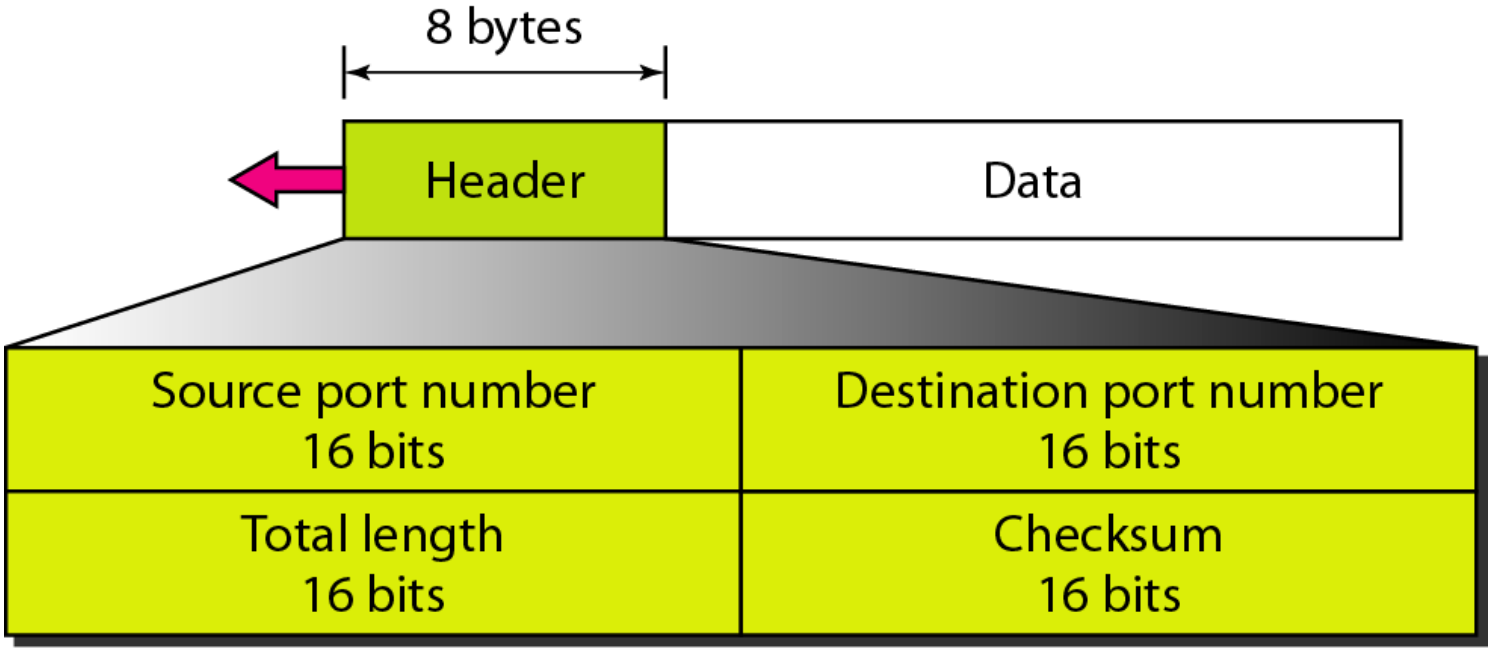
# 23-2 User Datagram Protocol (UDP)

- **Connectionless**
  - **No handshaking** between UDP sender, receiver
  - Each UDP segment handled **independently** of others
- A **server application** that uses UDP serves only **ONE request** at a time. All other requests are stored in a **queue** waiting for service.
- **Unreliable protocol has no flow and error control**
  - A UDP segment can be **lost**, **arrive out of order**, **duplicated**, or **corrupted**
  - **Checksum field checks error in the entire UDP segment. It is Optional**
  - UDP **doe not do anything to recover** from an error it simply **discard** the segment ➔ Application accepts full responsibility for errors
- It **uses port numbers** to multiplex/demultiplex data from/to the application layer.
- Advantages: Simple, **minimum overhead**, **no connection delay**
- **Services provided by UDP:**
  - Process-to-Process delivery
  - Error checking (however, if there is an error UDP does NOT do anything to recover from **error. It will just** discard the message

# UDP Applications

- Used for applications that can tolerate small amount of packet loss:
  - Multimedia applications,
  - Internet telephony,
  - real-time-video conferencing
  - Domain Name System messages
  - Audio
  - Routing Protocols

## Figure 23.9 *User datagram format*




**Header size = 8 bytes**

Minimum UDP **process data** size 0 bytes

Maximum UDP **process data** size=

65535 – 20 (network layer headers) - 8 (UDP headres)= **65507** bytes

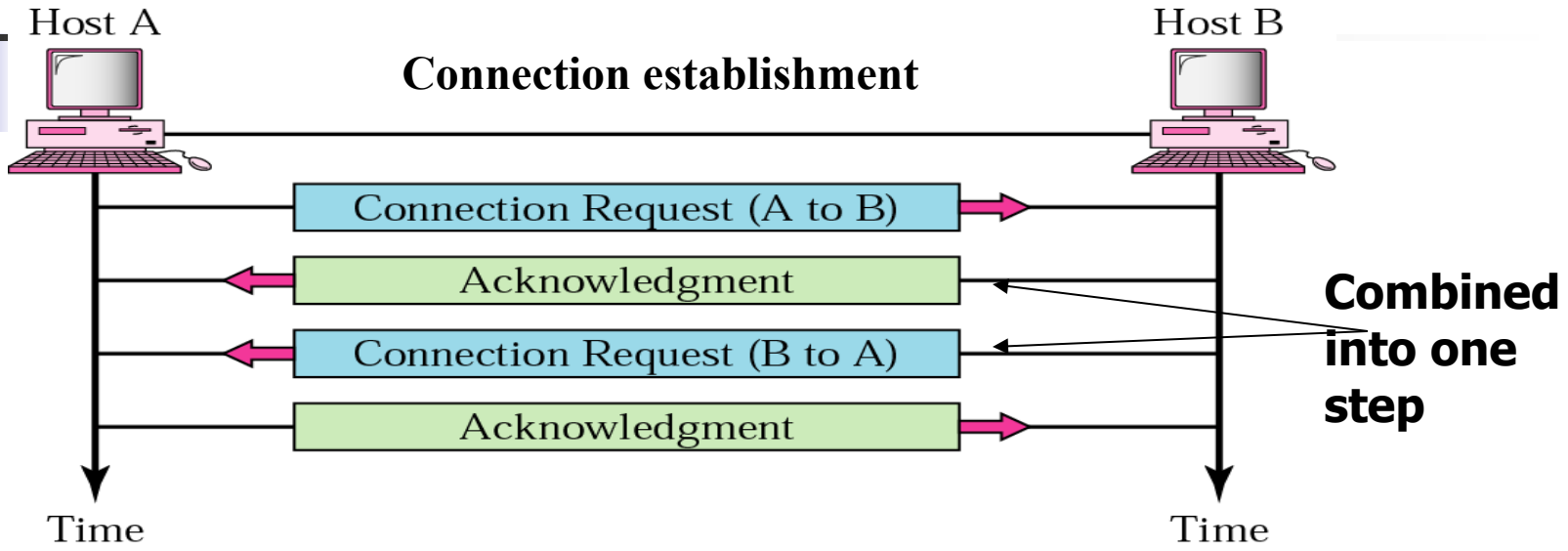
# 23-3 Transmission Control Protocol (TCP)

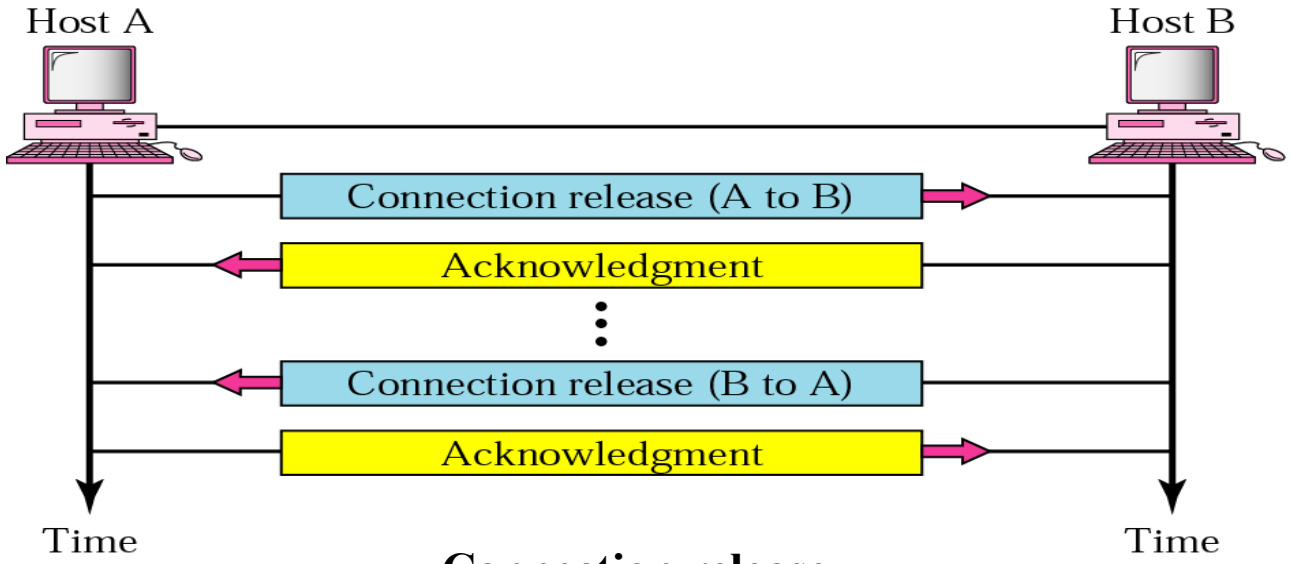
- Transmission Control Protocol properties:
  - Connection-oriented (establishment & termination)
  - Reliable
  - Full-duplex

# Connection-Oriented

- *Connection oriented* means that a virtual connection is established before any data is transferred.
- Connection ensures that the receiving process is available and ready **before** the data is sent
- **Three-way handshaking connection** establishment procedure because TCP is full-duplex both side must initialize communication and get approval from the other side before any data transfer,
- Virtual connection since TCP protocol will make sure that segments are given to the receiver application in the same order as they were sent by the sender even if they travel through different physical paths
- A server application that uses TCP can handle **many client** requests at the **same time** each has **its own connection**.

# Connection establishment and termination

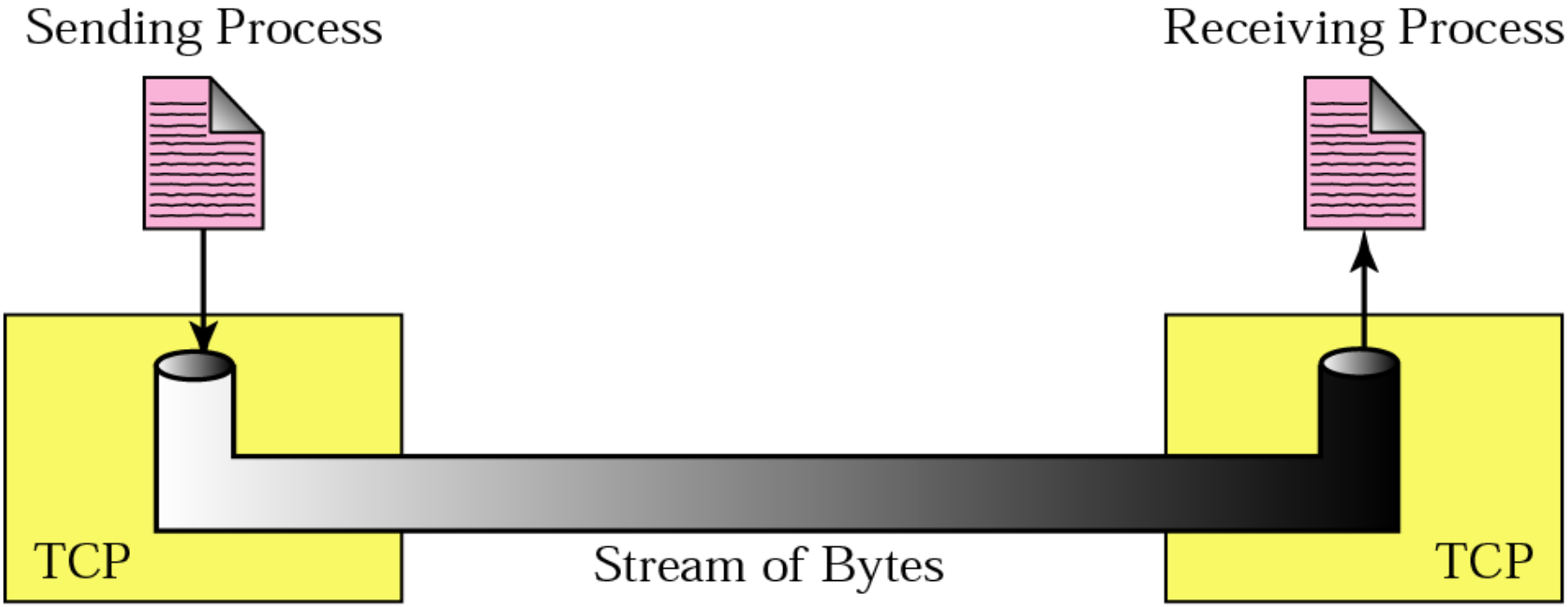



Connection release

**Figure 23.13** **Stream delivery**

# TCP establishes a virtual connection




**TCP will deliver segments to the applications in order and without error, lost, or duplicates**

# Full Duplex

- Data segments can flow in both directions at the same time.
- Each TCP connection has its own sending and receiving buffers.

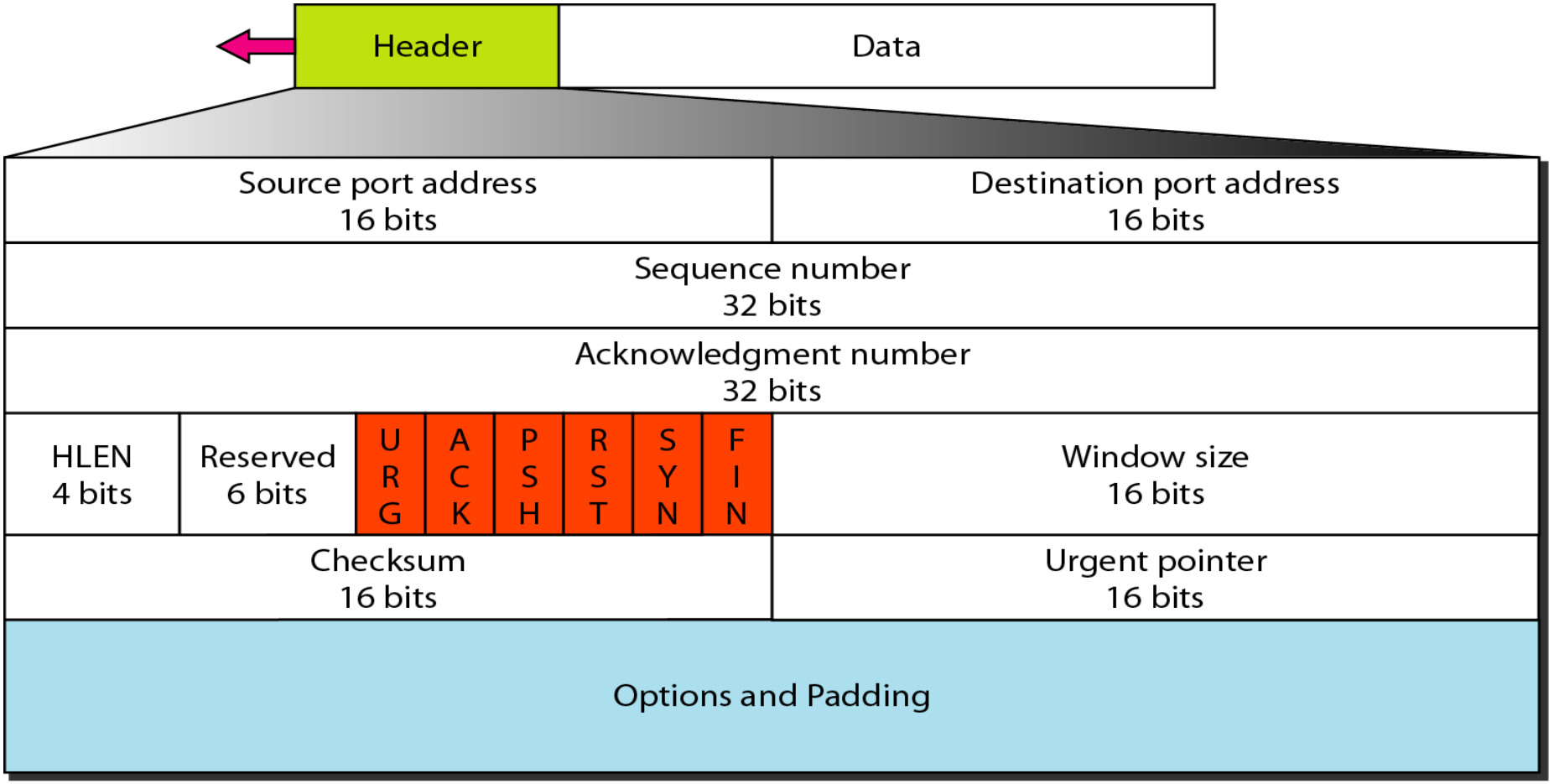
# Flow control and Reliability

- **Flow control** (process-to-process): TCP makes sure that the sender does not cause the receiver buffer to overflow
  - By defining the amount of data that can be sent before receiving an acknowledgement from the receiver **(sliding – window protocols**)
- **Error control** (process-to-process): **entire message** arrives at the receiving transport layer **without error, loss, duplication and in the same order they were sent**
  - Error detection is done using checksum and correction by retransmission
  - Implemented by a **sliding window ARQ**
  - **Every transmission** of data is **acknowledged** by the receiver.
  - Acknowledgements are **cumulative.**
  - If the sender does not receive ACK within a specified amount of time, the sender **retransmits** the data.
  - **Accepts out of order but does Not send negative acknowledgements**,
  - if a segment is not acknowledged before time-out, it is considered to be either **corrupted or lost** and the sender will **retransmit the segment only when it times-out**

# TCP Applications

- Following applications require reliable data transfer through TCP:
  - WWW using HTTP
  - Electronic mail using SMTP
  - Telnet
  - File transfer using FTP

# Figure 23.16 *TCP segment format*




**Minimum header length is 20 bytes and the maximum is 60 bytes when there are options**

# Transport layer duties

- **Packetizing**
  - Sender side: breaks **application messages** into segments, passes them to network layer
  - Transport layer at the receiving host deliver data to the receiving process
- **Connection control**
  - Connection-oriented
  - Connectionless
- **Addressing**
  - Port numbers to identify which network application
- **Reliability**
  - Flow control
  - Error Control
  - Congestion control