

- **Computer Networks**
- **Al-Mustansiryah University**
- **Elec. Eng. Department College of Engineering**  
**Fourth Year Class**

# Chapter 8

## Transport Layer: UDP and TCP

## 8-1 PROCESS-TO-PROCESS DELIVERY

The transport layer is responsible for process-to-process delivery—the delivery of a packet, part of a message, from one process to another. Two processes communicate in a client/server relationship, as we will see later.

---

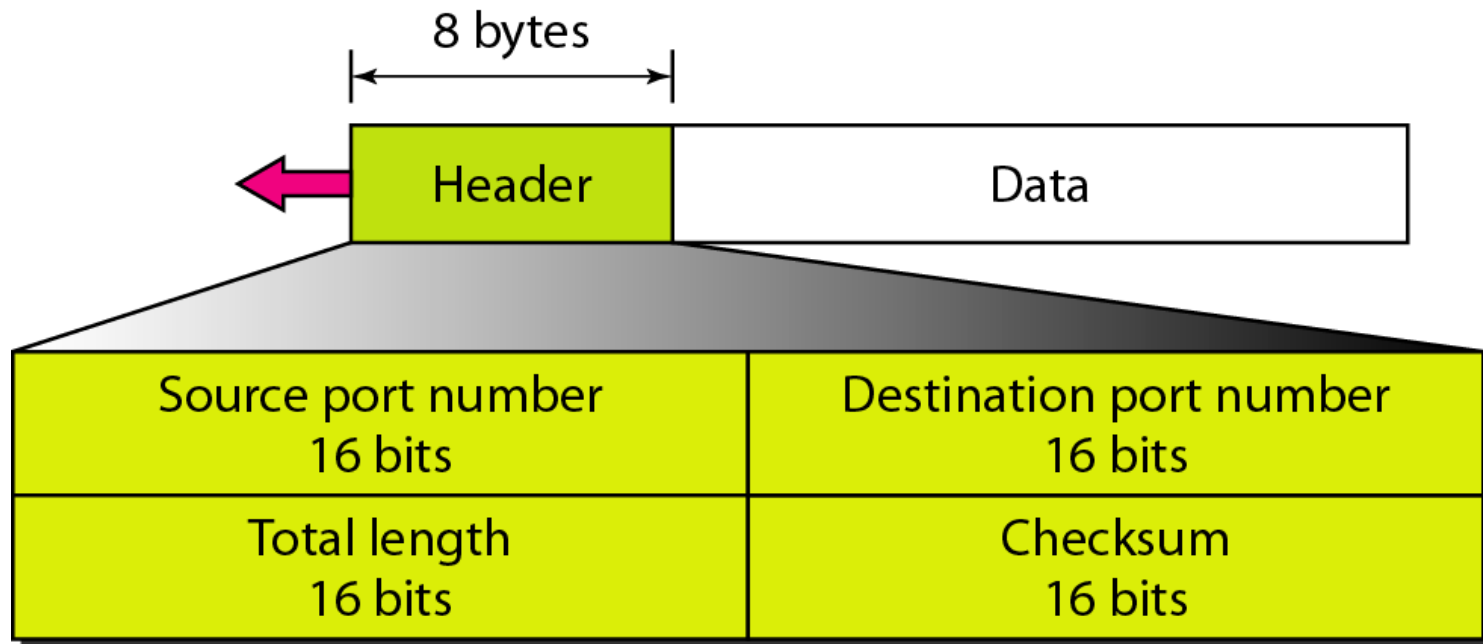
- **Connection-oriented** Requires a session connection (analogous to a phone call) be established before any data can be sent. This method is often called a "reliable" network service. It can guarantee that data will arrive in the same order. Connection-oriented services set up virtual links between end systems through a network.

- **Connectionless** Does not require a session connection between sender and receiver. The sender simply starts sending packets (called datagrams) to the destination. This service does not have the reliability of the connection-oriented method, but it is useful for periodic burst transfers.

## 8-2 USER DATAGRAM PROTOCOL (UDP)

The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication.

**Figure 8.9** *User datagram format*



---

**Example 1** :The following is a dump of a UDP header in hexadecimal format.

**CB84000D001C001C**

- a. What is the source port number?
- b. What is the destination port number?
- c. What is the total length of the user datagram?
- d. What is the length of the data?

---

**Solution:**

- a. The source port number is the first four hexadecimal digits (CB84), which means that the source port number is 52100.
- b. The destination port number is the second four hexadecimal digits (000D), which means that the destination port number is 13.
- c. The third four hexadecimal digits (001C) define the length of the whole UDP packet as 28 bytes.
- d. The length of the data is the length of the whole packet minus the length of the header, or  $28 - 8 = 20$  bytes.



---

*Note*

---

**UDP length  
= IP length – IP header's length**

---



## 8-3 TCP

TCP is a connection-oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level.




---

*Note*

---

**The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number.**

---



**Example 2 :** Suppose a TCP connection is transferring a file of 5,000 bytes. The first byte is numbered 10,001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1,000 bytes?

<b>Segment 1</b>	<b>➔</b>	<b>Sequence Number: 10,001 (range: 10,001 to 11,000)</b>
<b>Segment 2</b>	<b>➔</b>	<b>Sequence Number: 11,001 (range: 11,001 to 12,000)</b>
<b>Segment 3</b>	<b>➔</b>	<b>Sequence Number: 12,001 (range: 12,001 to 13,000)</b>
<b>Segment 4</b>	<b>➔</b>	<b>Sequence Number: 13,001 (range: 13,001 to 14,000)</b>
<b>Segment 5</b>	<b>➔</b>	<b>Sequence Number: 14,001 (range: 14,001 to 15,000)</b>



---

*Note*

---

**The value in the sequence number field of a segment defines the number of the first data byte contained in that segment.**

---



**Note**

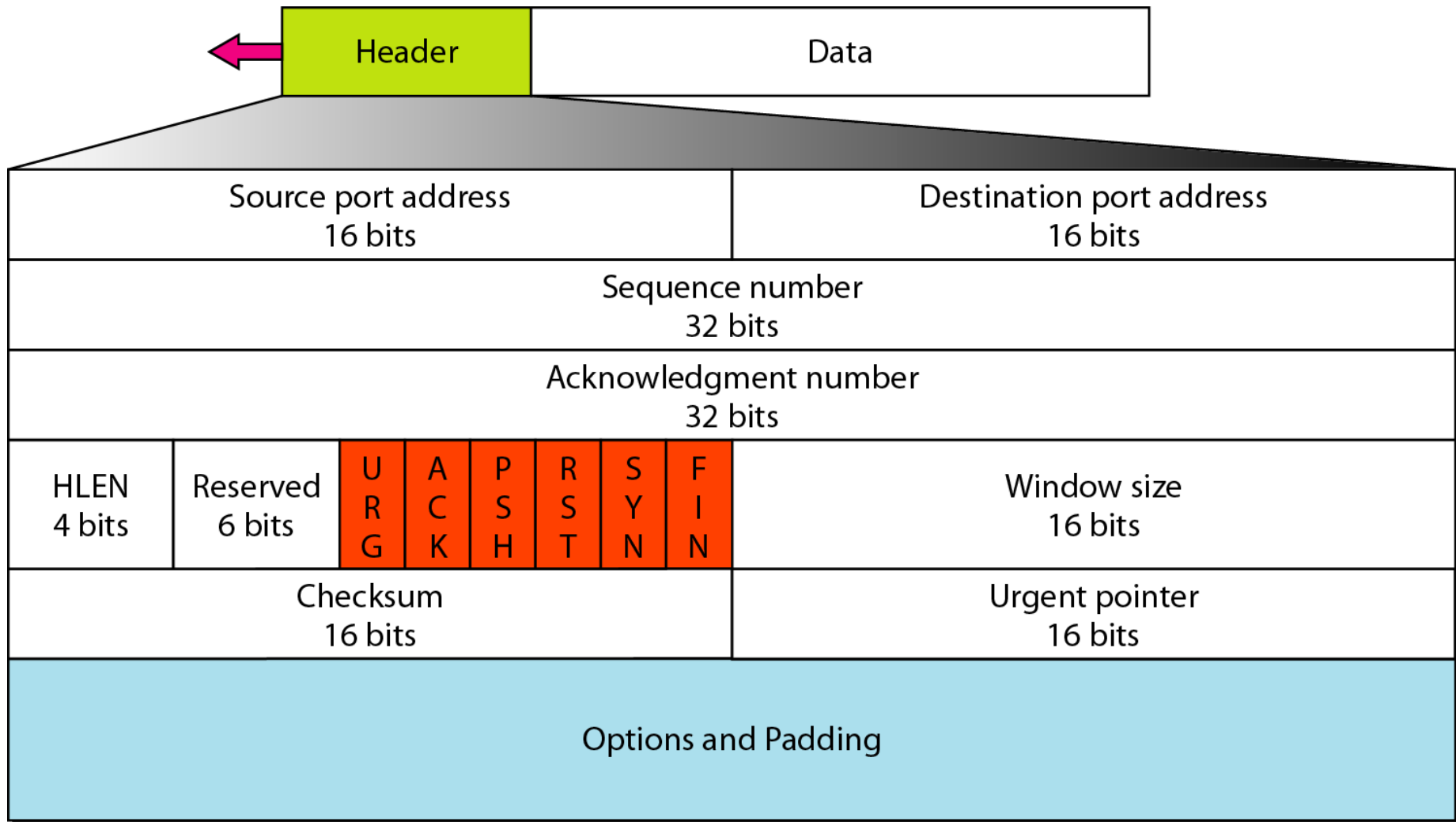
---

**The value of the acknowledgment field  
in a segment defines  
the number of the next byte a party  
expects to receive.**

**The acknowledgment number is  
cumulative.**

---

**Figure 8.16** *TCP segment format*



---

**Source port address.** This is a 16-bit field that defines the port number of the application program in the host that is sending the segment

**Destination port address.** This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment

**Sequence number.** This 32-bit field defines the number assigned to the first byte of data contained in this segment

**Acknowledgment number.** This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number  $x$  from the other party, it returns  $x + 1$  as the acknowledgment number.

---

**Header length.** This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field is always between 5 ( $5 * 4 = 20$ ) and 15 ( $15 * 4 = 60$ ).

**Reserved.** This is a 6-bit field reserved for future use.

**Control.** This field defines 6 different control bits or flags

**Window size.** This field defines the window size of the sending TCP in bytes. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes.

**Checksum.** This 16-bit field contains the checksum

**Urgent pointer.** This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data



---

## Figure 8.18 *Control field*

---

URG: Urgent pointer is valid  
ACK: Acknowledgment is valid  
PSH: Request for push

RST: Reset the connection  
SYN: Synchronize sequence numbers  
FIN: Terminate the connection



**Table 8.3** *Description of flags in the control field*

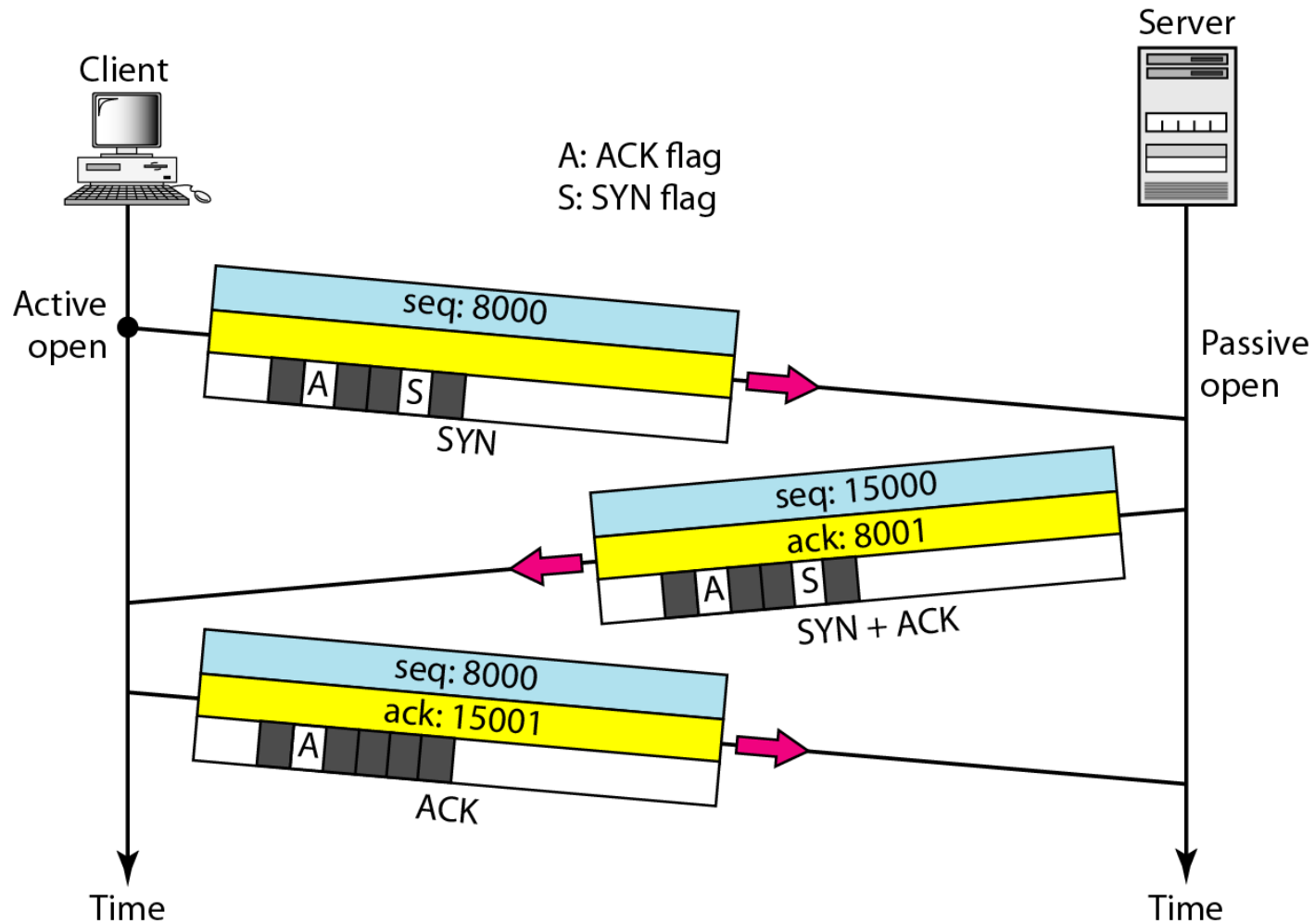
<i>Flag</i>	<i>Description</i>
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

---

## **Flow Control**

Flow control assists the reliability of TCP transmission by adjusting the effective rate of data flow between the two services in the session. Window Size field in the TCP header specifies the amount of data that can be transmitted before an acknowledgement must be received.

**Figure 8.18** *Connection establishment using three-way handshaking*





---

**A SYN segment cannot carry data, but it consumes one sequence number.**

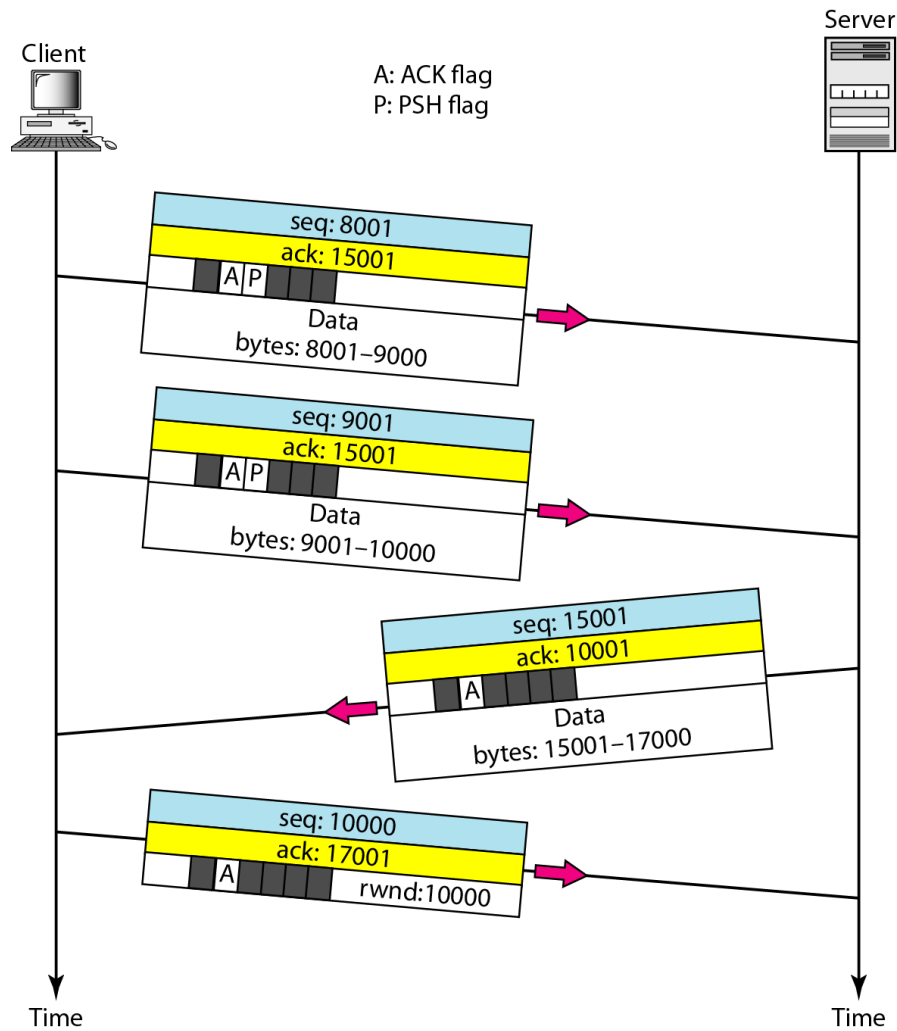
---

**A SYN + ACK segment cannot carry data, but does consume one sequence number.**

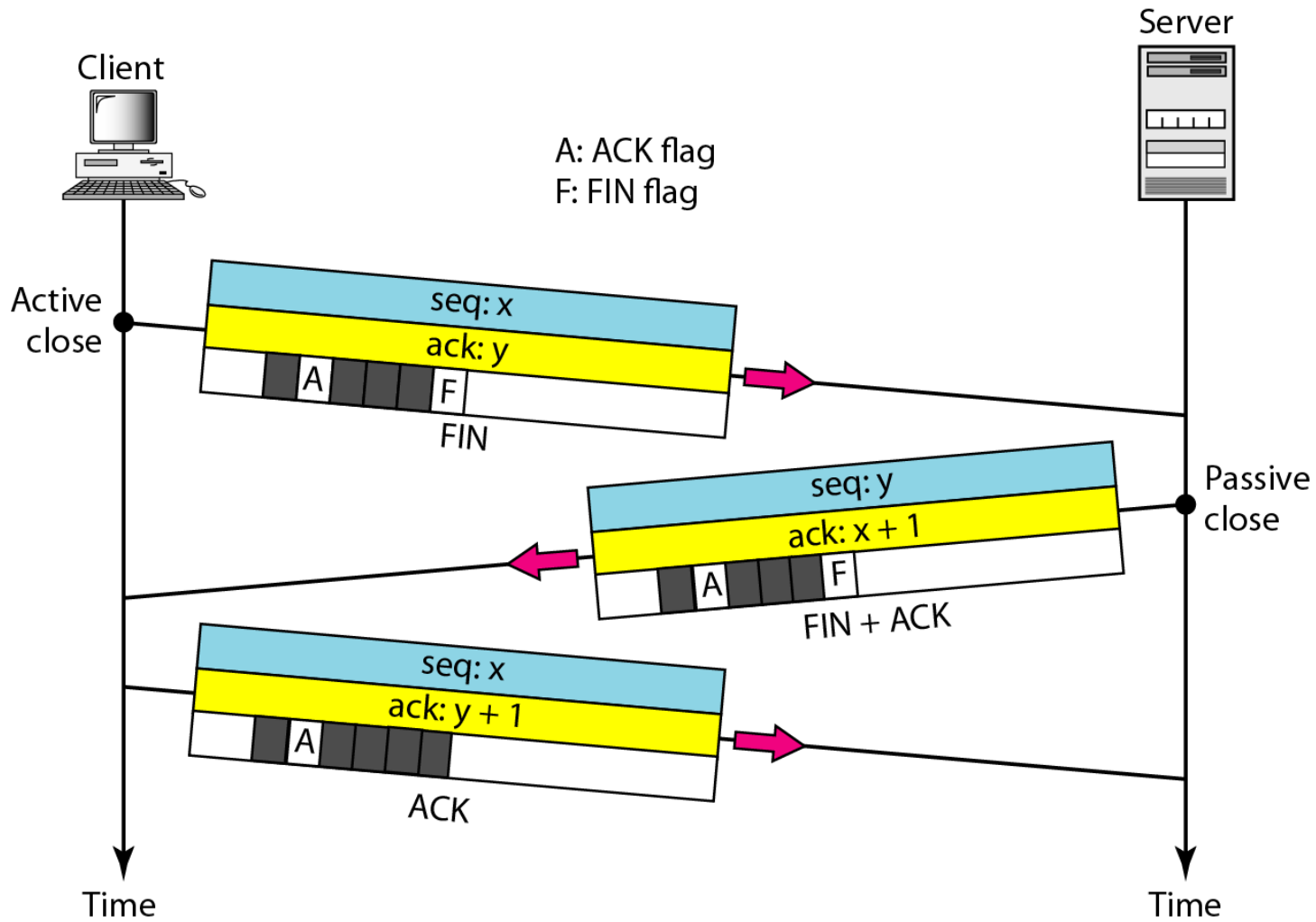
---

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

**Figure 8.19** *Data transfer*



**Figure 8.20** *Connection termination using three-way handshaking*





---

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

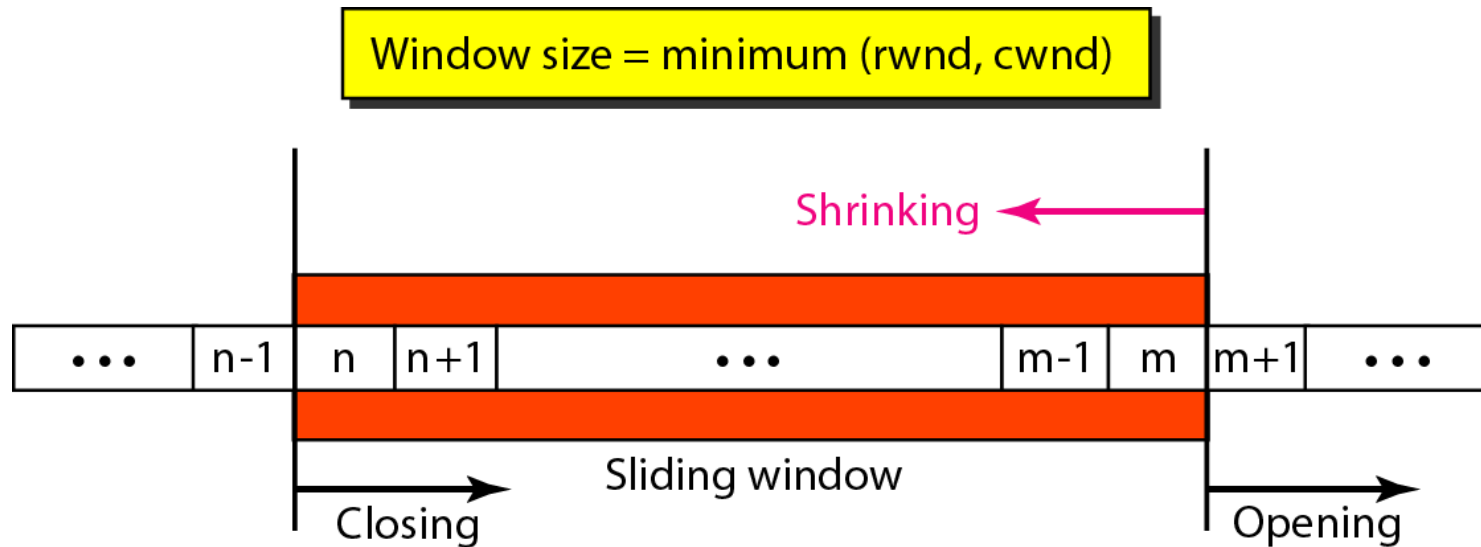
---

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



## Figure 8.22 *Sliding window*

$rwnd = \text{buffer size} - \text{number of waiting bytes to be pulled}$





*Note*

---

**A sliding window is used to make transmission more efficient as well as to control the flow of data so that the destination does not become overwhelmed with data.  
TCP sliding windows are byte-oriented.**

---



## Example 8.4

**What is the value of the receiver window (rwnd) for host A if the receiver, host B, has a buffer size of 5000 bytes and 1000 bytes of received and unprocessed data?**

### **Solution**

**The value of  $rwnd = 5000 - 1000 = 4000$ . Host B can receive only 4000 bytes of data before overflowing its buffer. Host B advertises this value in its next segment to A.**



## *Example 8.5*

---

*What is the size of the window for host A if the value of  $rwnd$  is 3000 bytes and the value of  $cwnd$  is 3500 bytes?*

### *Solution*

*The size of the window is the smaller of  $rwnd$  and  $cwnd$ , which is 3000 bytes.*

## Some points about TCP sliding windows:

- ❑ The size of the window is the lesser of `rwnd` and `cwnd`.
- ❑ The source does not have to send a full window's worth of data.
- ❑ The window can be opened or closed by the receiver, but should not be shrunk.
- ❑ The destination can send an acknowledgment at any time as long as it does not result in a shrinking window.
- ❑ The receiver can temporarily shut down the window; the sender, however, can always send a segment of 1 byte after the window is shut down.



---

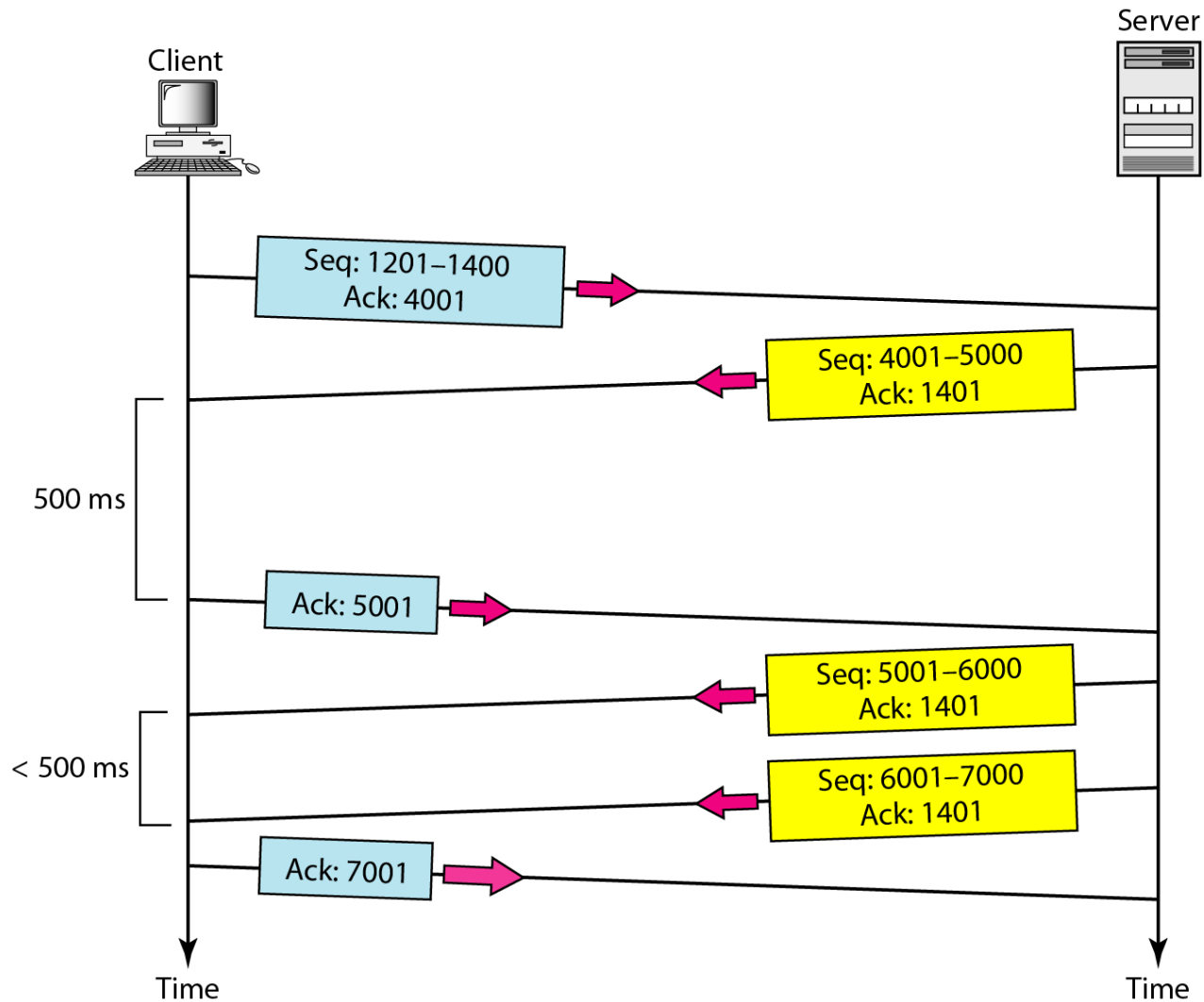
*Note*

---

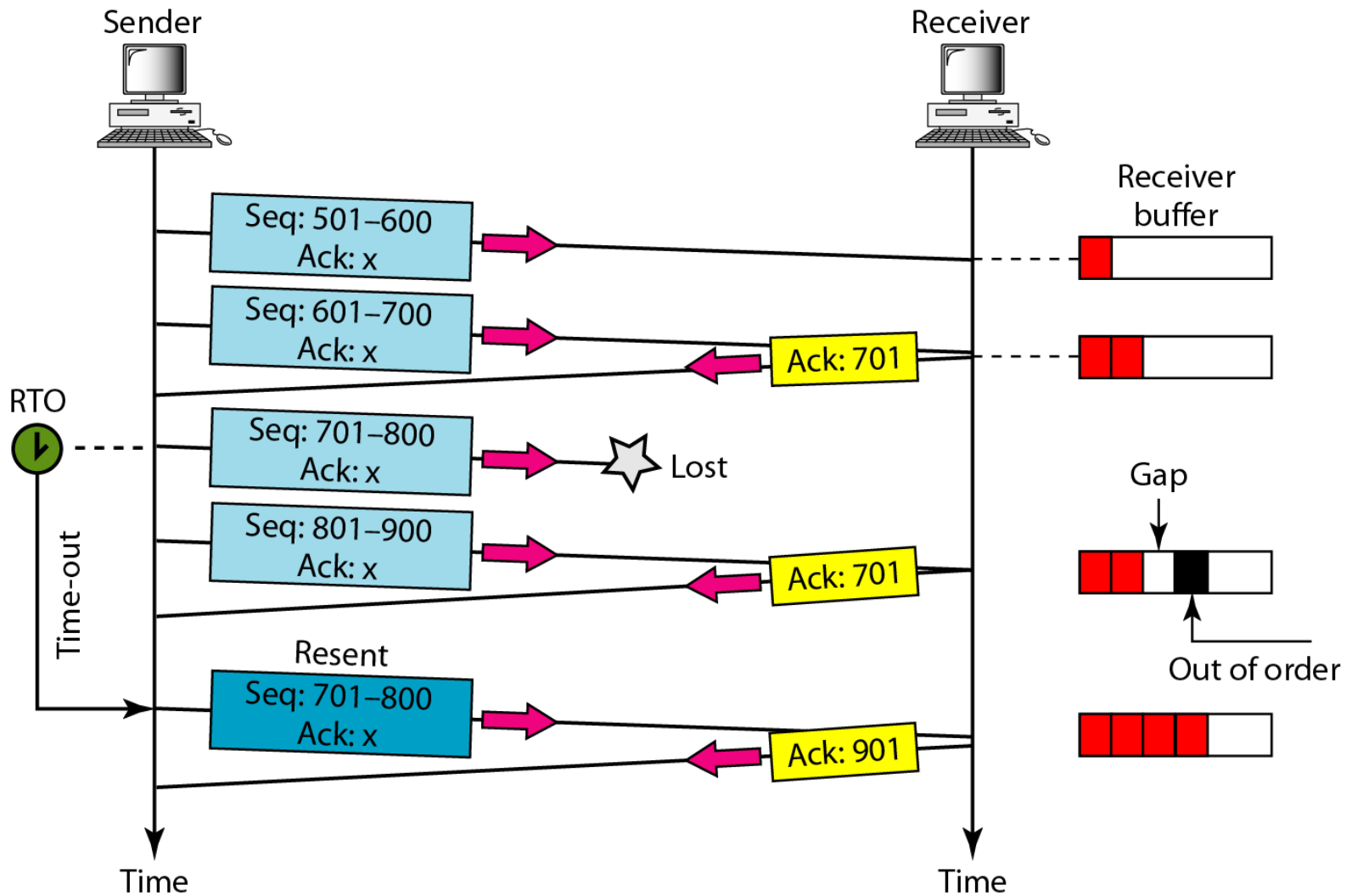
**Data may arrive out of order and be temporarily stored by the receiving TCP, but TCP guarantees that no out-of-order segment is delivered to the process.**

---

**Figure 8.24** *Normal operation*



**Figure 8.25** *Lost segment*







---

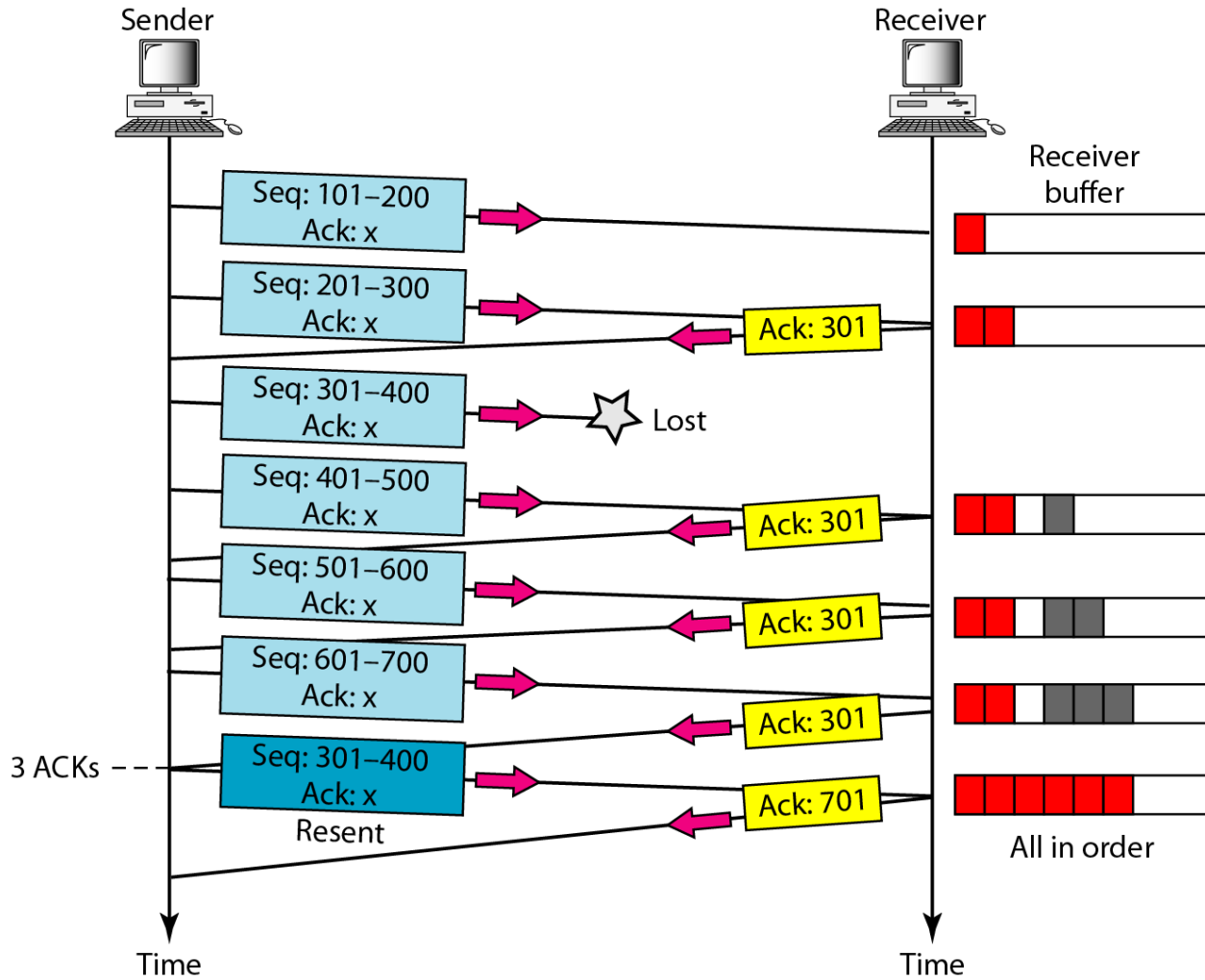
*Note*

---

**The receiver TCP delivers only ordered data to the process.**

---

**Figure 8.26** *Fast retransmission*



	<b>TCP</b>	<b>UDP</b>
<b>Connection</b>	TCP is a connection-oriented protocol.	UDP is a connectionless protocol.
<b>Ordering of data packets</b>	TCP rearranges data packets in the order specified.	UDP has no inherent order as all packets are independent of each other.
<b>Speed of transfer</b>	The speed for TCP is slower than UDP.	UDP is faster because there is no error-checking for packets.
<b>Reliability</b>	There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent.	There is no guarantee that the messages or packets sent would reach at all.
<b>Header Size</b>	TCP header size is 20 bytes	UDP Header size is 8 bytes.

	<b>TCP</b>	<b>UDP</b>
<b>Streaming of data</b>	Data is read as a byte stream,	Packets are sent individually
<b>Data Flow Control</b>	TCP does Flow Control. TCP requires three packets to set up a socket connection,	UDP does not have an option for flow control
<b>Acknowledgement</b>	Acknowledgement segments	No Acknowledgment