# Transmission Control Protocol(TCP)

*Kumkum Saxena*

# 3-4 TRANSMISSION CONTROL PROTOCOL

***Transmission Control Protocol (TCP) is a connection-oriented, reliable protocol. TCP explicitly defines connection establishment, data transfer, and connection teardown phases to provide a connection-oriented service. TCP uses a combination of GBN and SR protocols to provide reliability.***

# 3.4.1 TCP Services

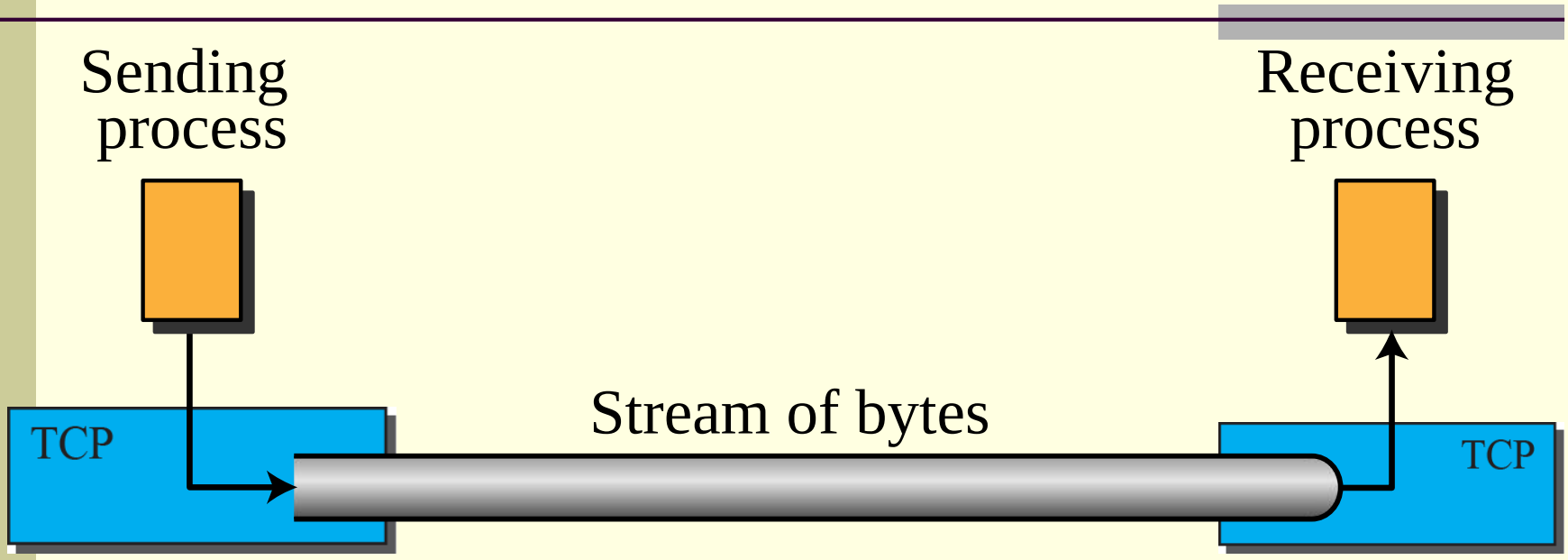
***Before discussing TCP in detail, let us explain the services offered by TCP to the processes at the application layer.***

- ***Process-to-Process Communication***
- ***Stream Delivery Service***
  - ***Sending and Receiving Buffers***
  - ***Segments***

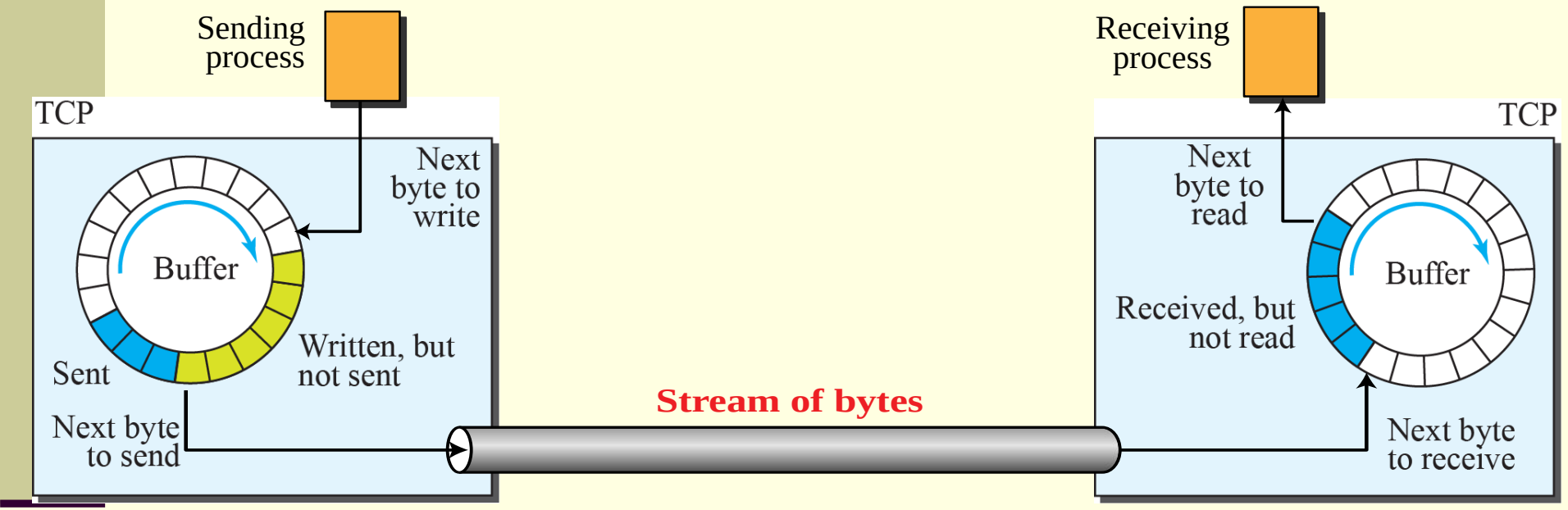
# 3.4.1 (continued)

- *Full-Duplex Communication*
- *Multiplexing and Demultiplexing*
- *Connection-Oriented Service*
- *Reliable Service*

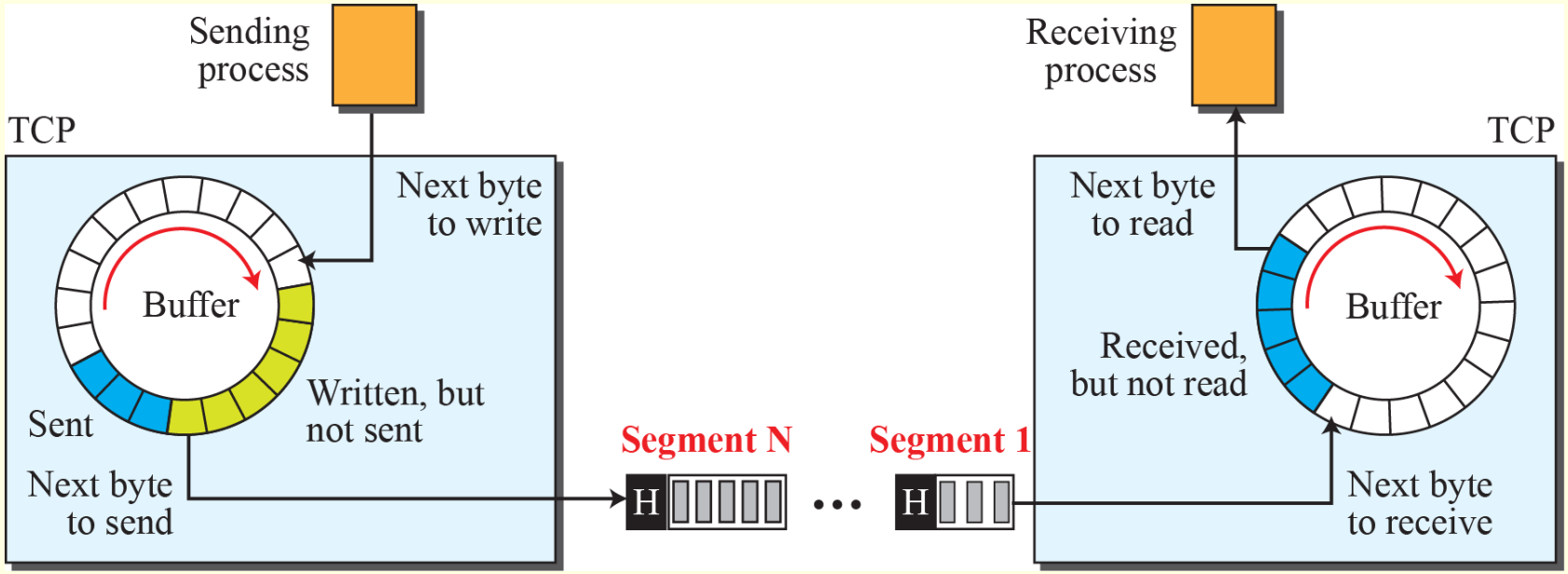
## *Figure 3.41: Stream delivery*



# Figure 3.42: Sending and receiving buffers



# Figure 3.43: TCP segments



# 3.4.2 TCP Features

***To provide the services mentioned in the previous section, TCP has several features that are briefly summarized in this section and discussed later in detail.***

- ***Numbering System***
  - ***Byte Number***
  - ***Sequence Number***
  - ***Acknowledgment Number***

Note

*The bytes of data being transferred in each connection are numbered by TCP.*

*The numbering starts with an arbitrarily generated number.*

# Sequence Number

- The sequence number in each direction is defined as follows:-
  - The sequence number of the first segment is the ISN(Initial sequence number) which is a random number.
  - The sequence number of any other segment is the sequence number of the previous segment plus the number of bytes carried by the previous segment.

Note

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

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?

Solution

The following shows the sequence number for each segment:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Segment 1 | → | Sequence Number: | 10,001 | **Range:** | 10,001 | to | 11,000 |
| Segment 2 | → | Sequence Number: | 11,001 | **Range:** | 11,001 | to | 12,000 |
| Segment 3 | → | Sequence Number: | 12,001 | **Range:** | 12,001 | to | 13,000 |
| Segment 4 | → | Sequence Number: | 13,001 | **Range:** | 13,001 | to | 14,000 |
| Segment 5 | → | Sequence Number: | 14,001 | **Range:** | 14,001 | to | 15,000 |

Note

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

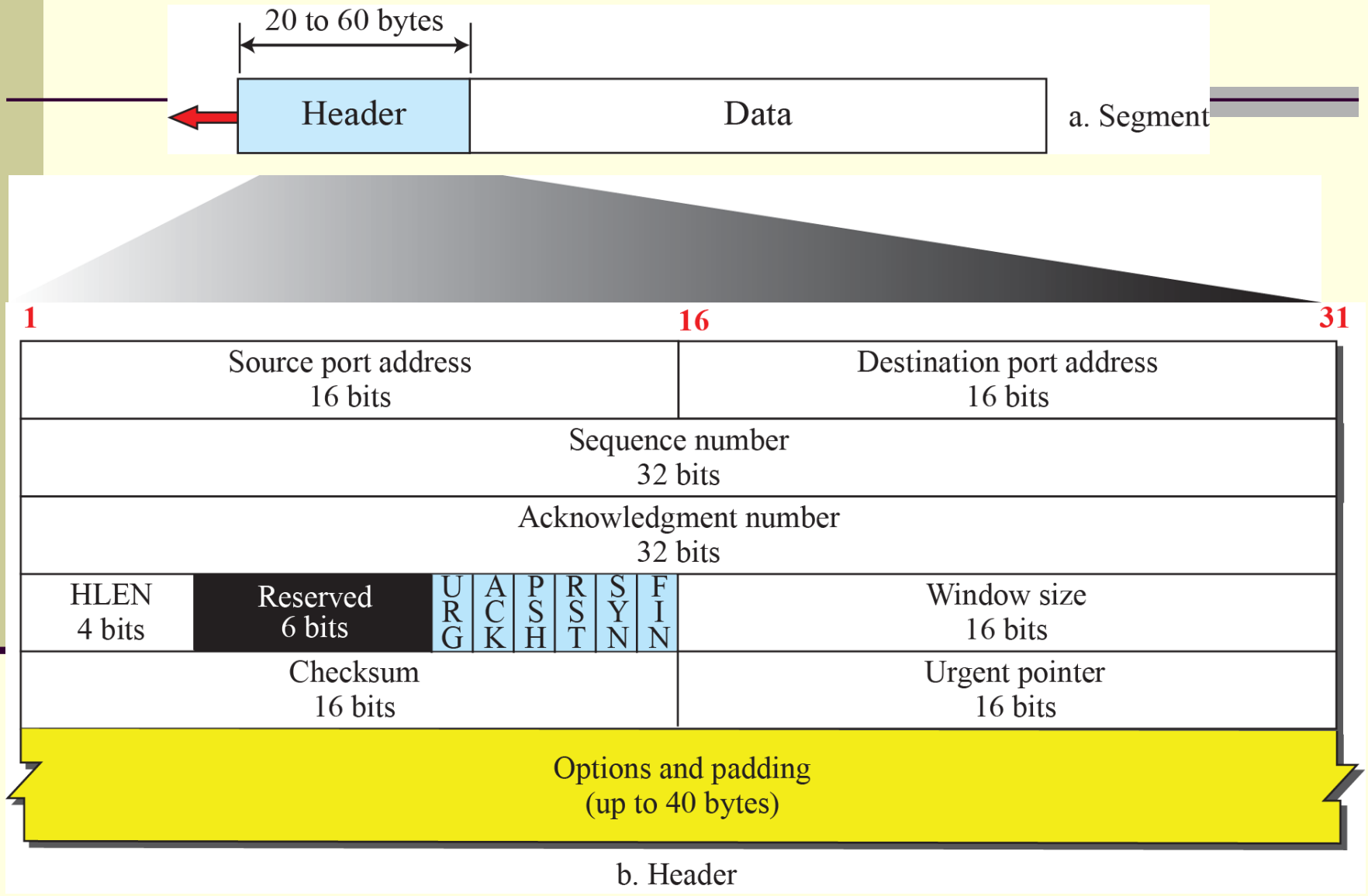
*The acknowledgment number is cumulative.*

# 3.4.3 Segment

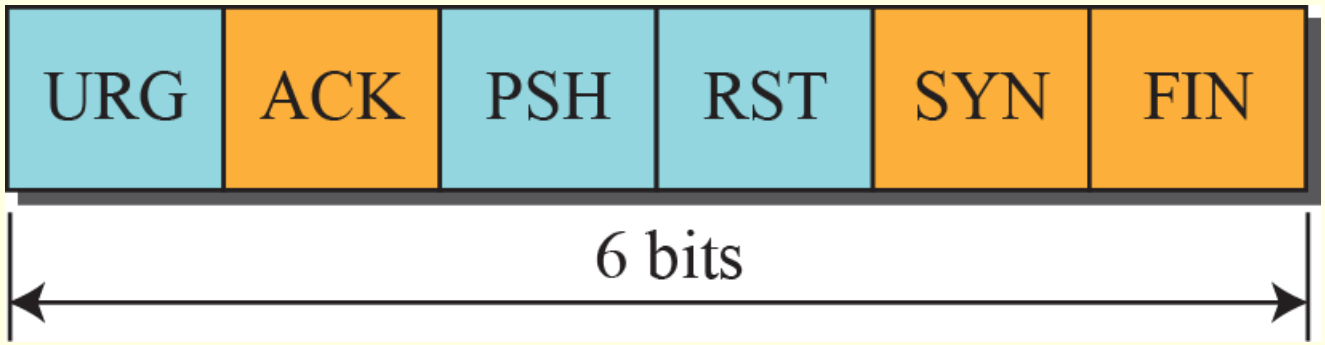
***Before discussing TCP in more detail, let us discuss the TCP packets themselves. A packet in TCP is called a segment.***

- ***Format***
- ***Encapsulation***

# Figure 3.44: TCP segment format



## *Figure 3.45: Control field*

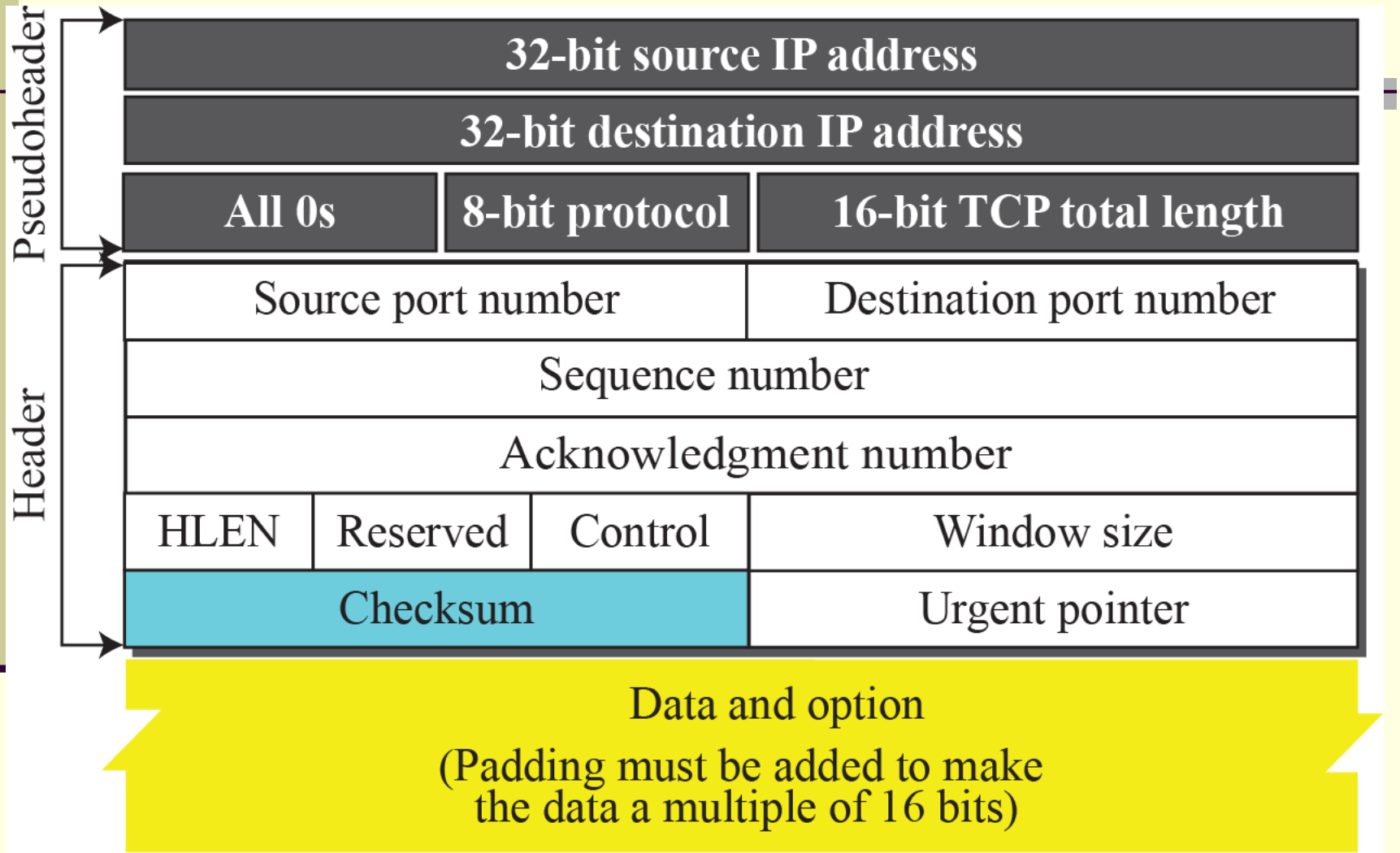



URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection

Note

*The use of the checksum in TCP is mandatory.*

## *Figure 3.46: Pseudoheader added to the TCP datagram*



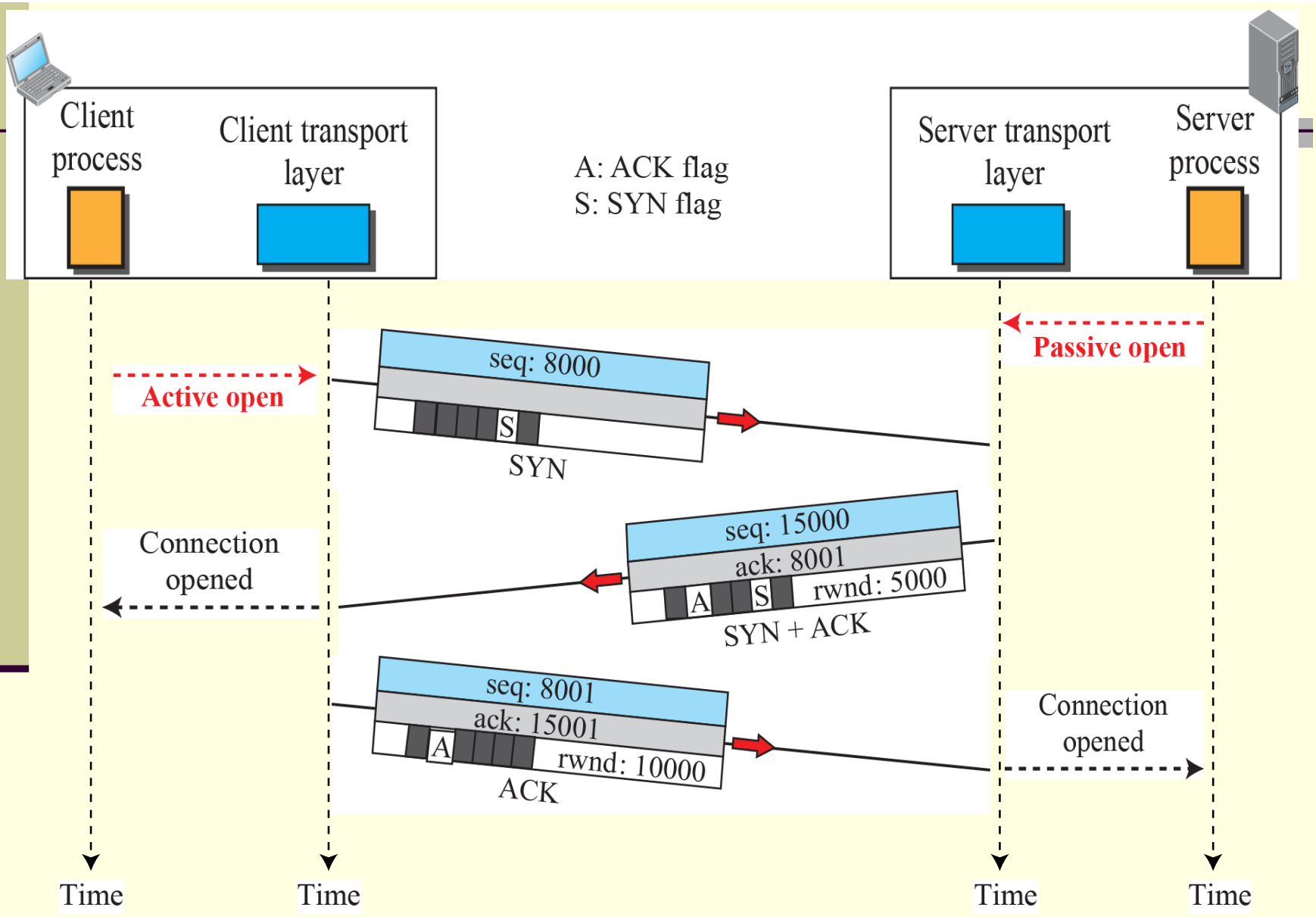
# *3.4.4 A TCP Connection*

***TCP is connection-oriented. As discussed before, a connection-oriented transport protocol establishes a logical path between the source and destination. All of the segments belonging to a message are then sent over this logical path. Using a single logical pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames.***

# 3.4.4 (continued)

- **Connection Establishment**
  - Three-Way Handshaking
  - SYN Flooding Attack
- **Data Transfer**
  - Pushing Data
  - Urgent Data
- **Connection Termination**
  - Three-Way Handshaking
  - Half-Close
- **Connection Reset**

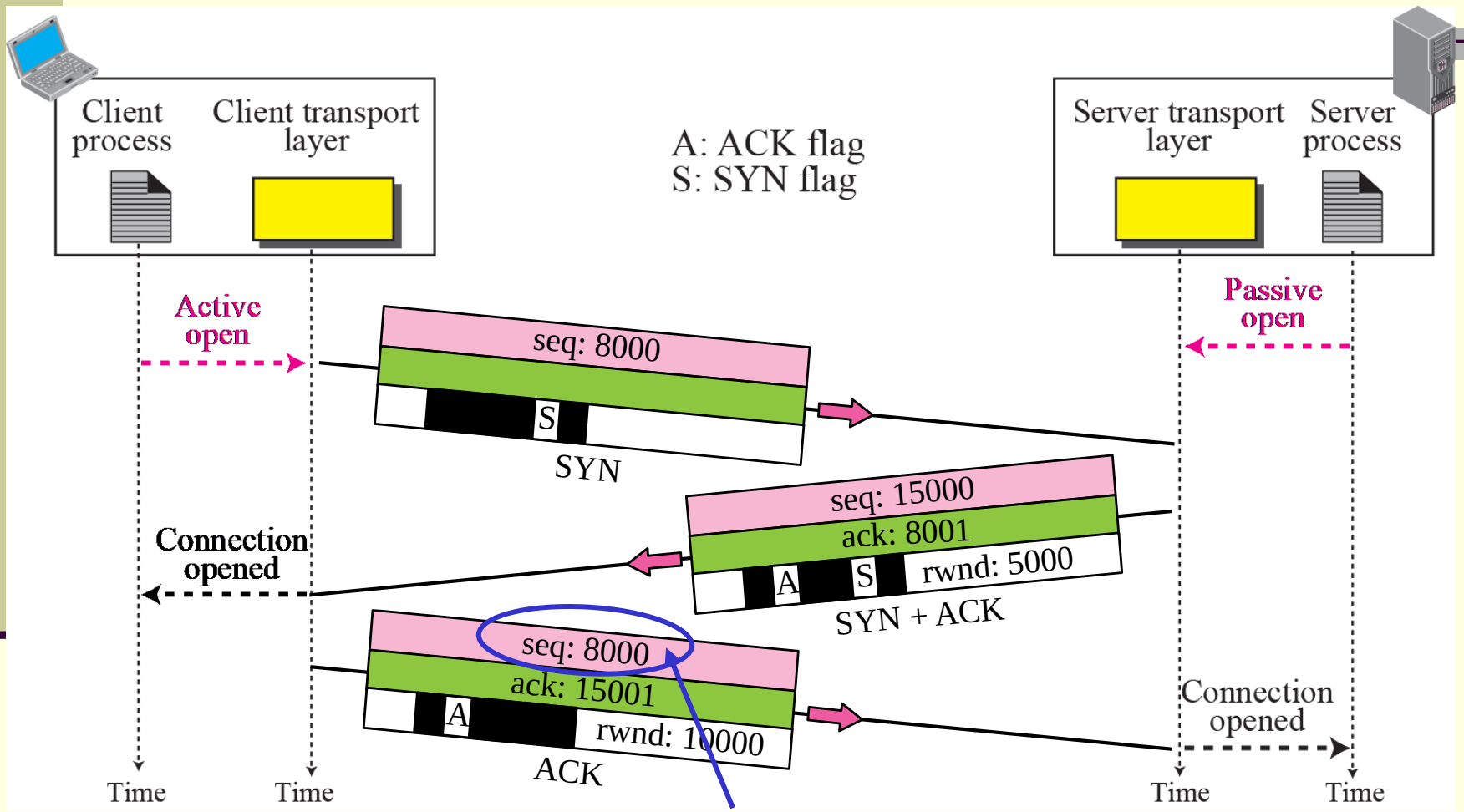
# Figure 3.47: Connection establishment using three-way handshaking



Note

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

## *Connection establishment using three-way handshake*




*Means "no data" !*

seq: 8001 if piggybacking

Note

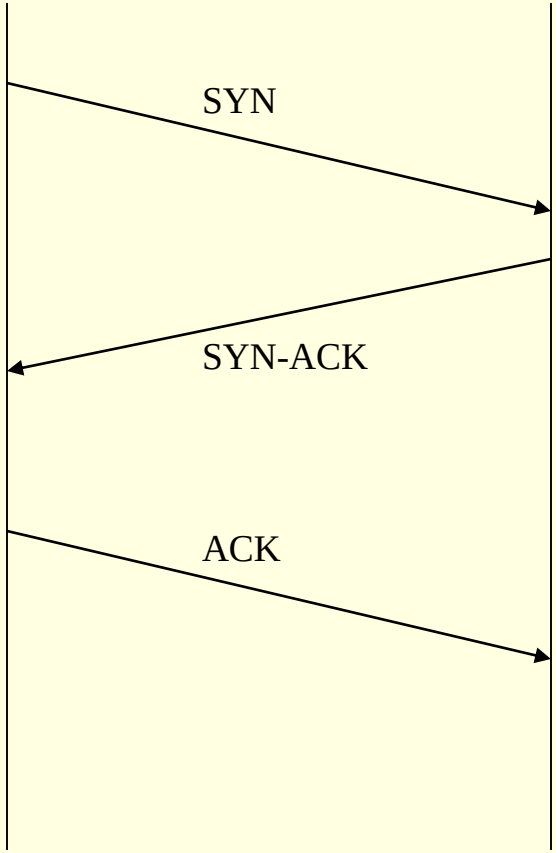
*A SYN + ACK segment cannot carry data, but does consume one sequence number.*

Note

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

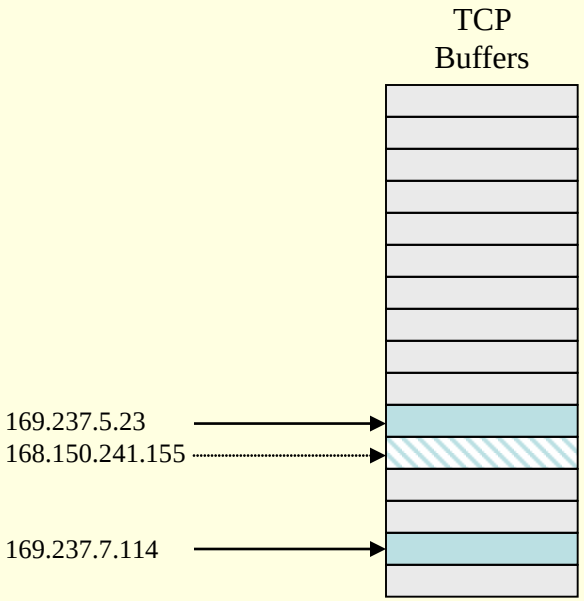
# SYN Flood Attack

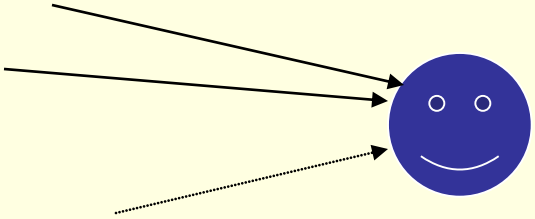
- TCP Handshake Review
  - client
    - sends SYN packet to server
    - waits for SYN-ACK from server
  - server
    - responds with SYN-ACK packet
    - waits for ACK packet from client
  - client
    - sends ACK to server

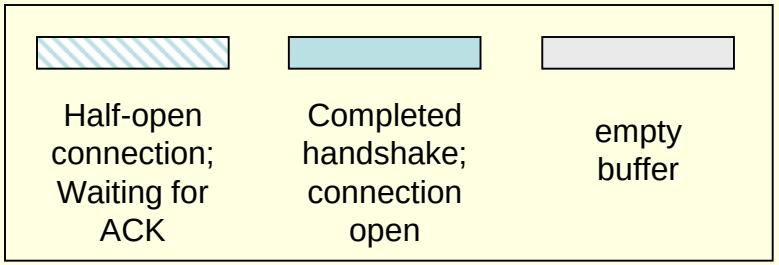


# SYN Flood Attack

- Attacker causes TCP buffer to be exhausted with half-open connections
- No reply from target needed, so source may be spoofed.
- Claimed source must not be an active host.

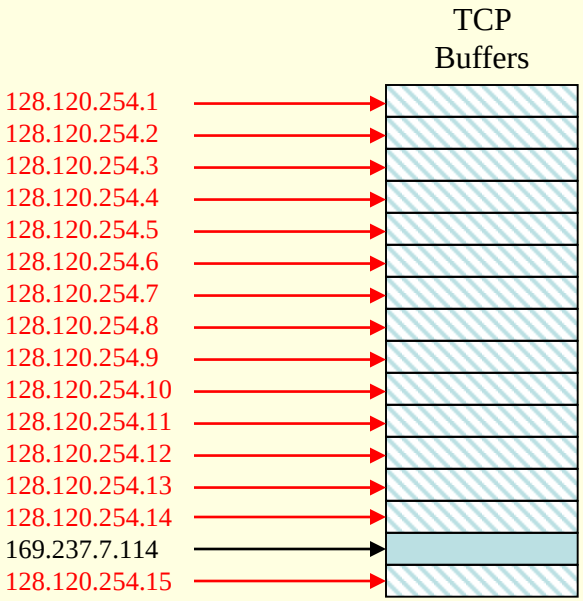


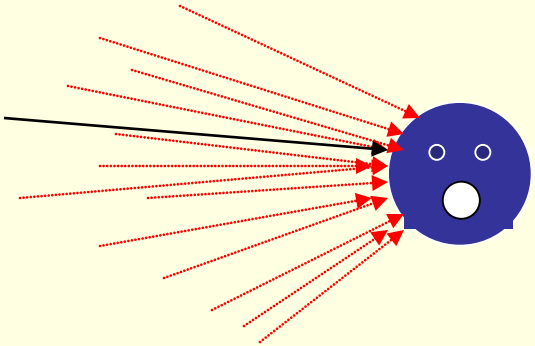



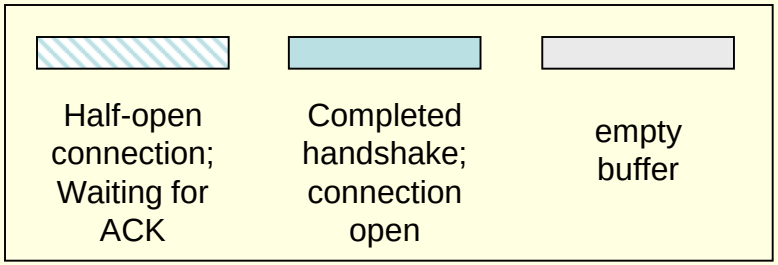


# SYN Flood Attack

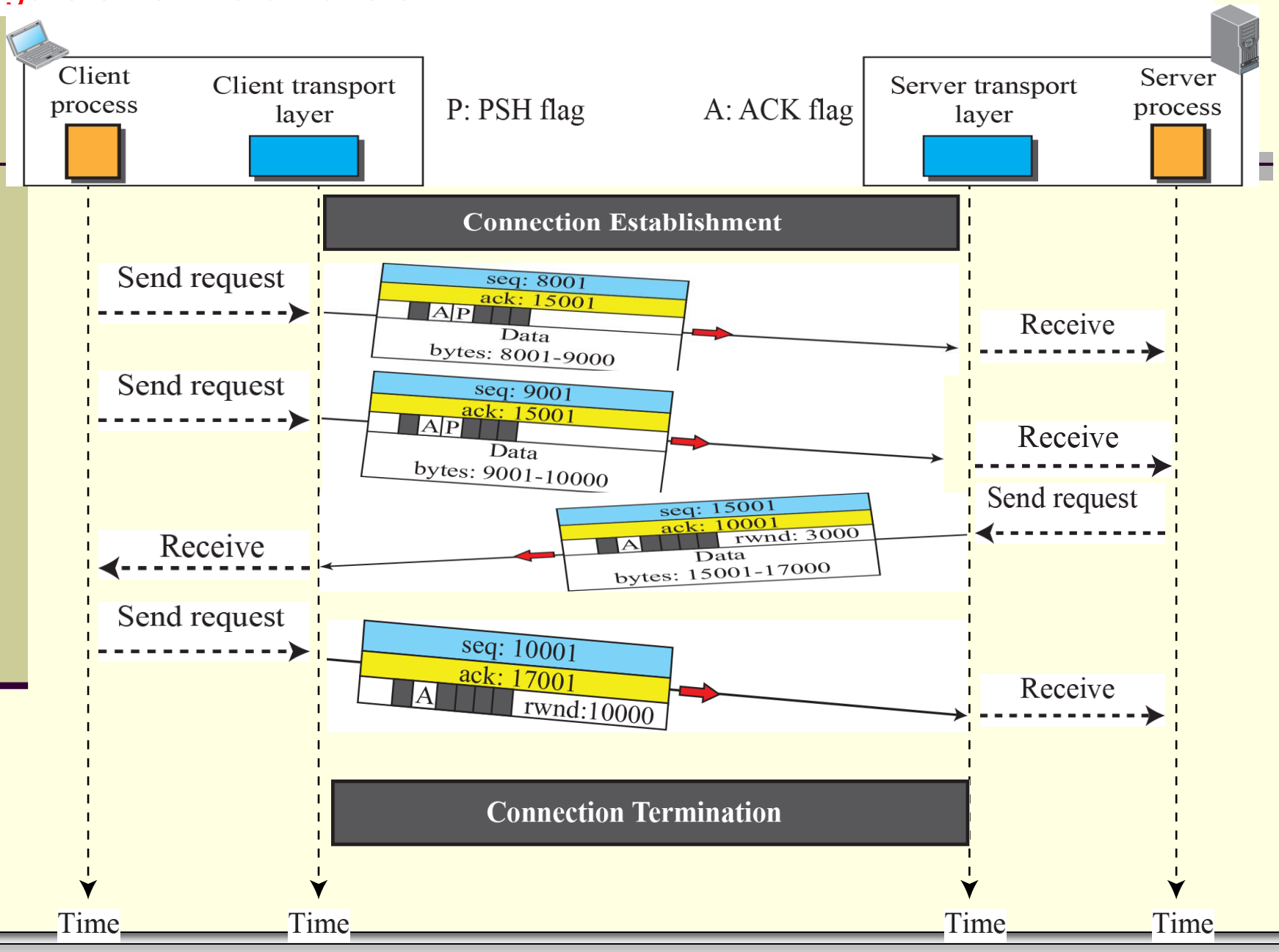
- Attacker causes TCP buffer to be exhausted with half-open connections
- No reply from target needed, so source may be spoofed.
- Claimed source must not be an active host.



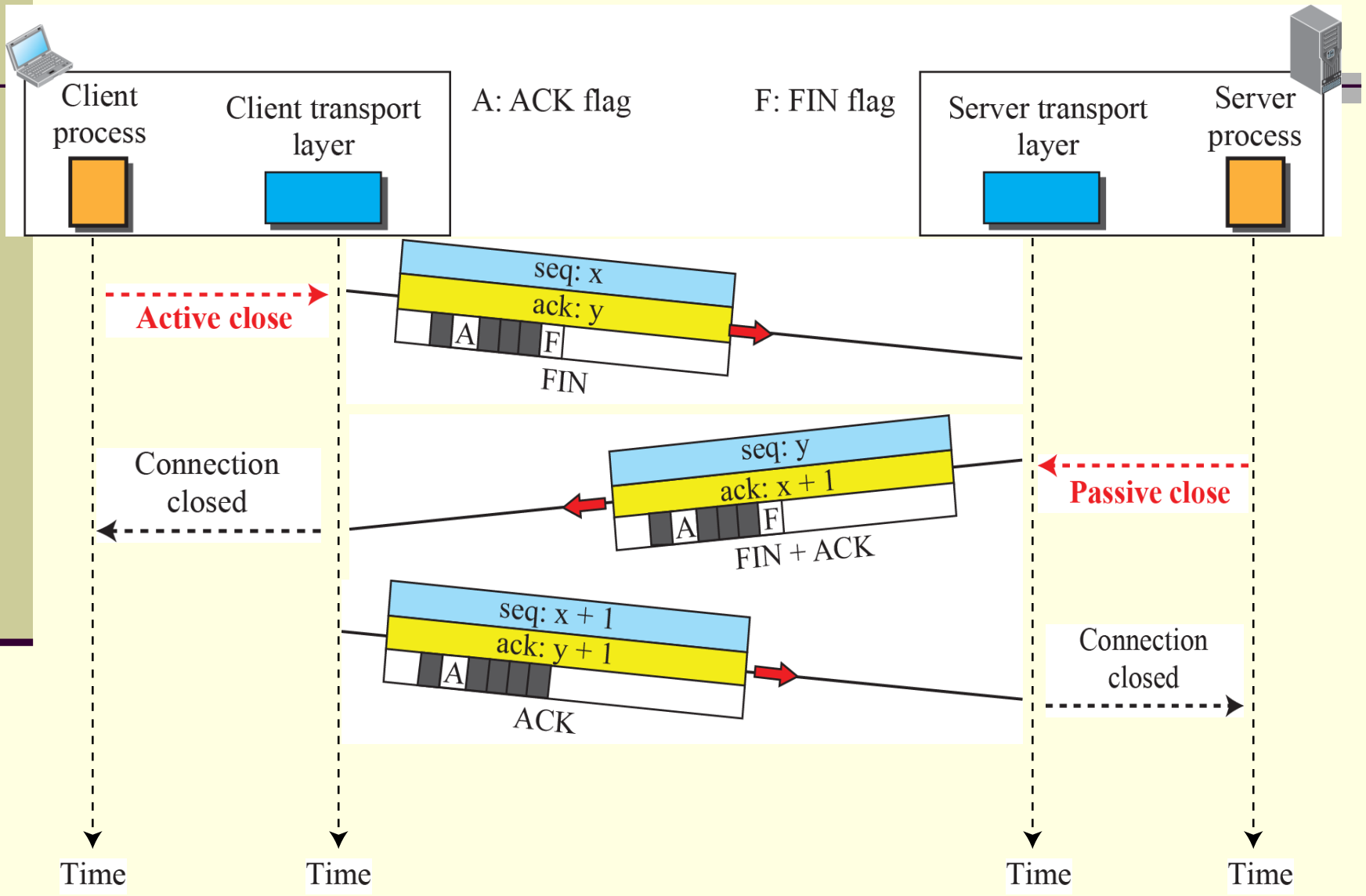





# Figure 3.48: Data transfer



# Figure 3.49: Connection termination using three-way handshaking



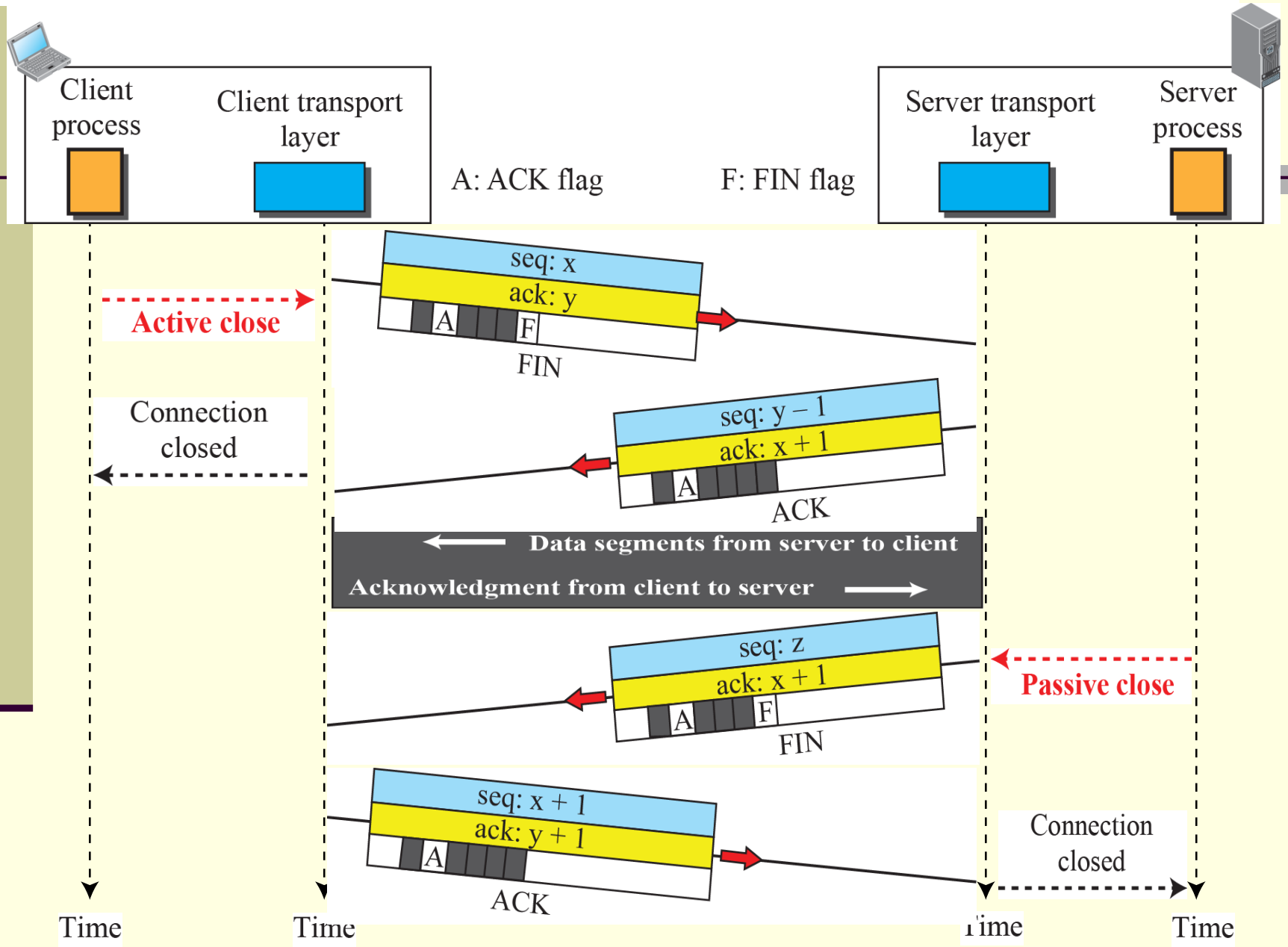
Note

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

Note

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

# Figure 3.50: Half-close

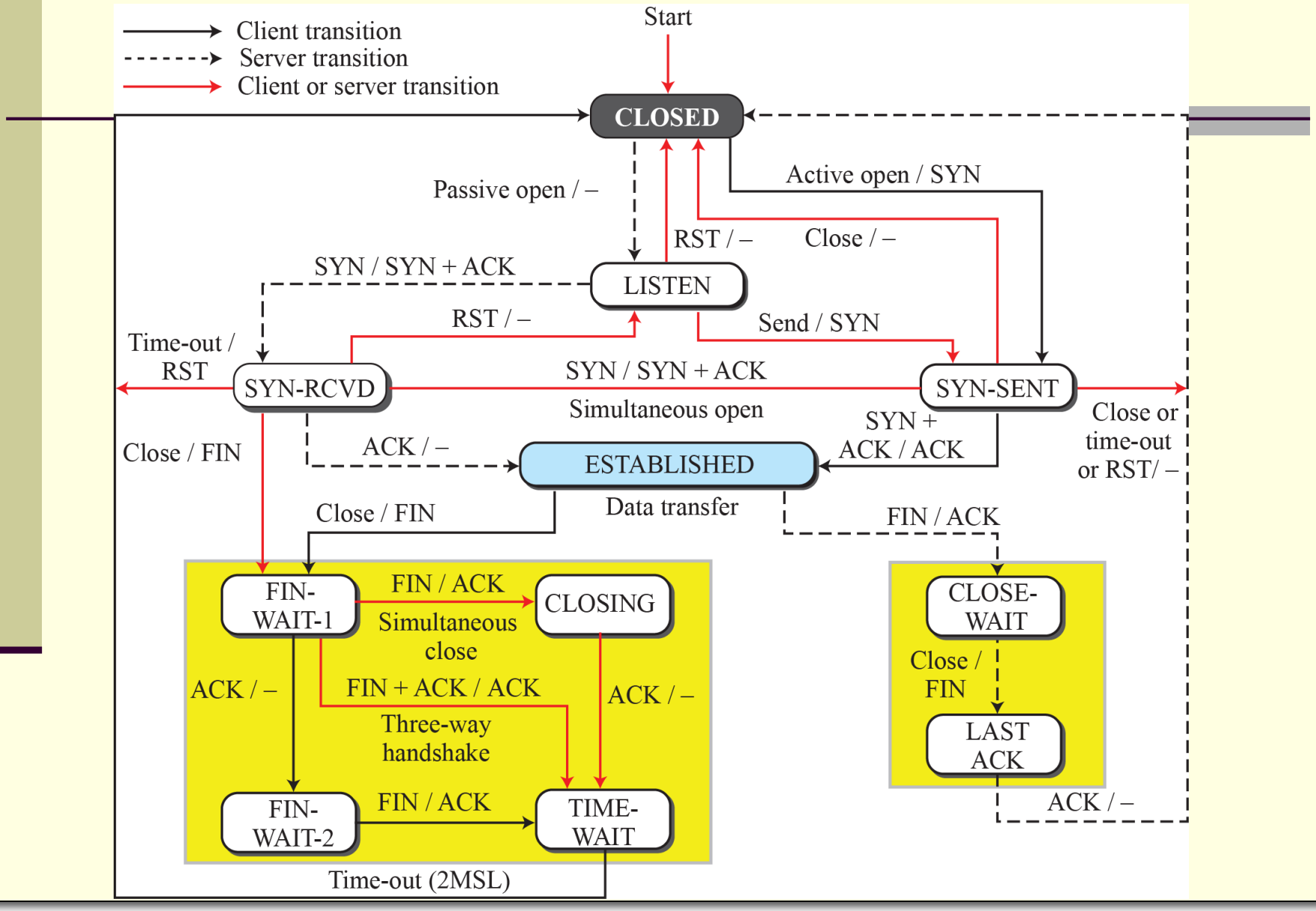


# 3.4.5 State Transmission Diagram

*To keep track of all the different events happening during connection establishment, connection termination, and data transfer, TCP is specified as the finite state machine (FSM) as shown in Figure .*

- **Scenarios**
  - A Half-Close Scenario

# Figure 3.51: State transition diagram



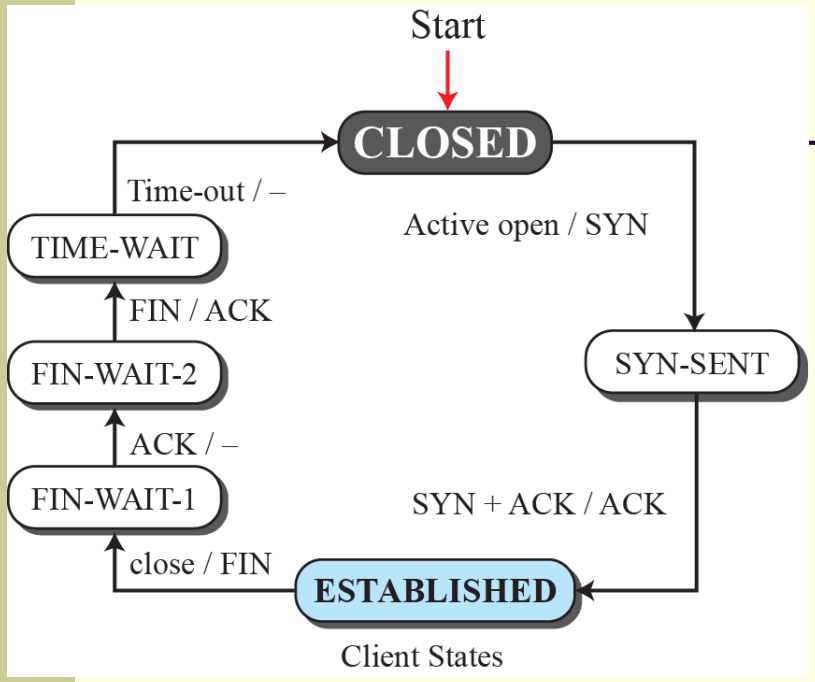
Note

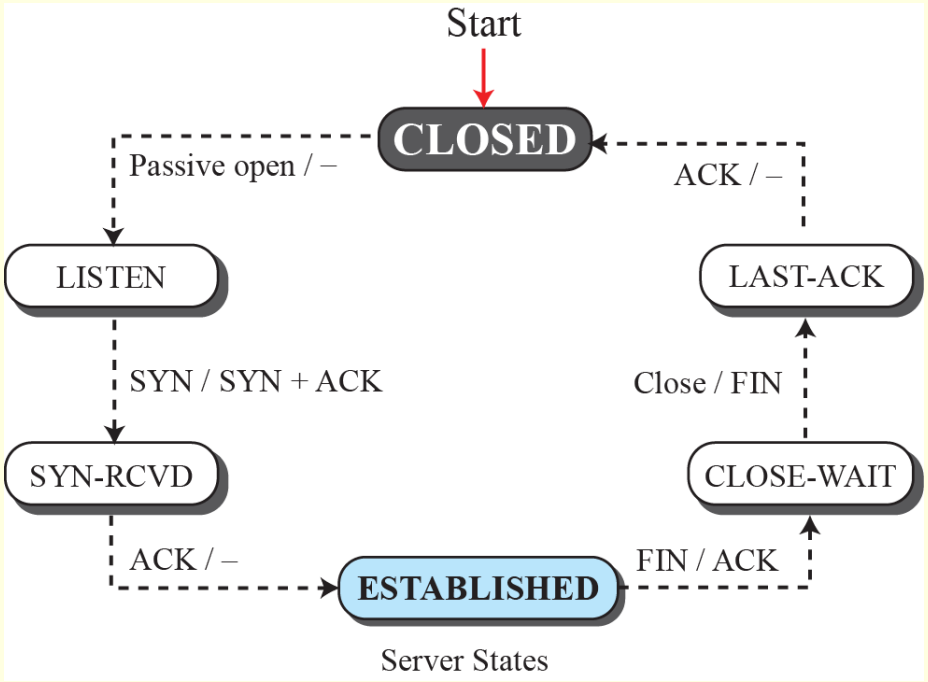
*The state marked as ESTABLISHED in the FSM is in fact two different sets of states that the client and server undergo to transfer data.*

# Table 3.2: States for TCP

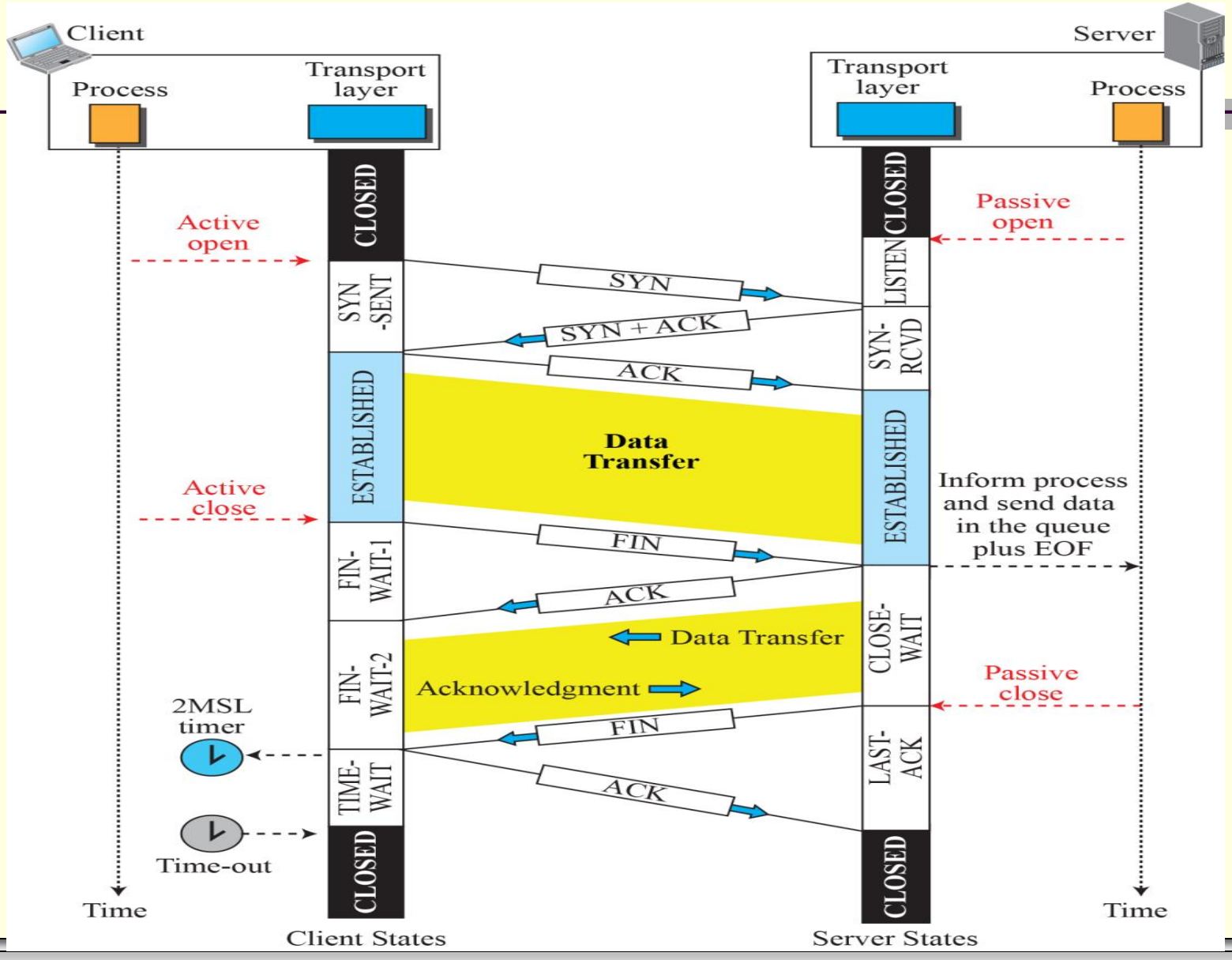
| *State* | *Description* |
|---|---|
| **CLOSED** | No connection exists |
| **LISTEN** | Passive open received; waiting for SYN |
| **SYN-SENT** | SYN sent; waiting for ACK |
| **SYN-RCVD** | SYN+ACK sent; waiting for ACK |
| **ESTABLISHED** | Connection established; data transfer in progress |
| **FIN-WAIT-1** | First FIN sent; waiting for ACK |
| **FIN-WAIT-2** | ACK to first FIN received; waiting for second FIN |
| **CLOSE-WAIT** | First FIN received, ACK sent; waiting for application to close |
| **TIME-WAIT** | Second FIN received, ACK sent; waiting for 2MSL time-out |
| **LAST-ACK** | Second FIN sent; waiting for ACK |
| **CLOSING** | Both sides decided to close simultaneously |

# Figure 3.52: Transition diagram with half-close connection termination



# Figure 3.53: Time-line diagram for a common scenario



# *Time-line diagram*

Enough time for an ACK to be lost and a new FIN to arrive. If during the TIME-WAIT state, a new FIN arrives, the client sends a new ACK and restarts the 2MSL timer

To prevent a duplicate segment from one connection appearing in the next one, TCP requires that incarnation cannot take place unless 2MSL amount of time has elapsed.

Another solution: the ISN of the incarnation is greater than the last seq. # used in the previous connection.