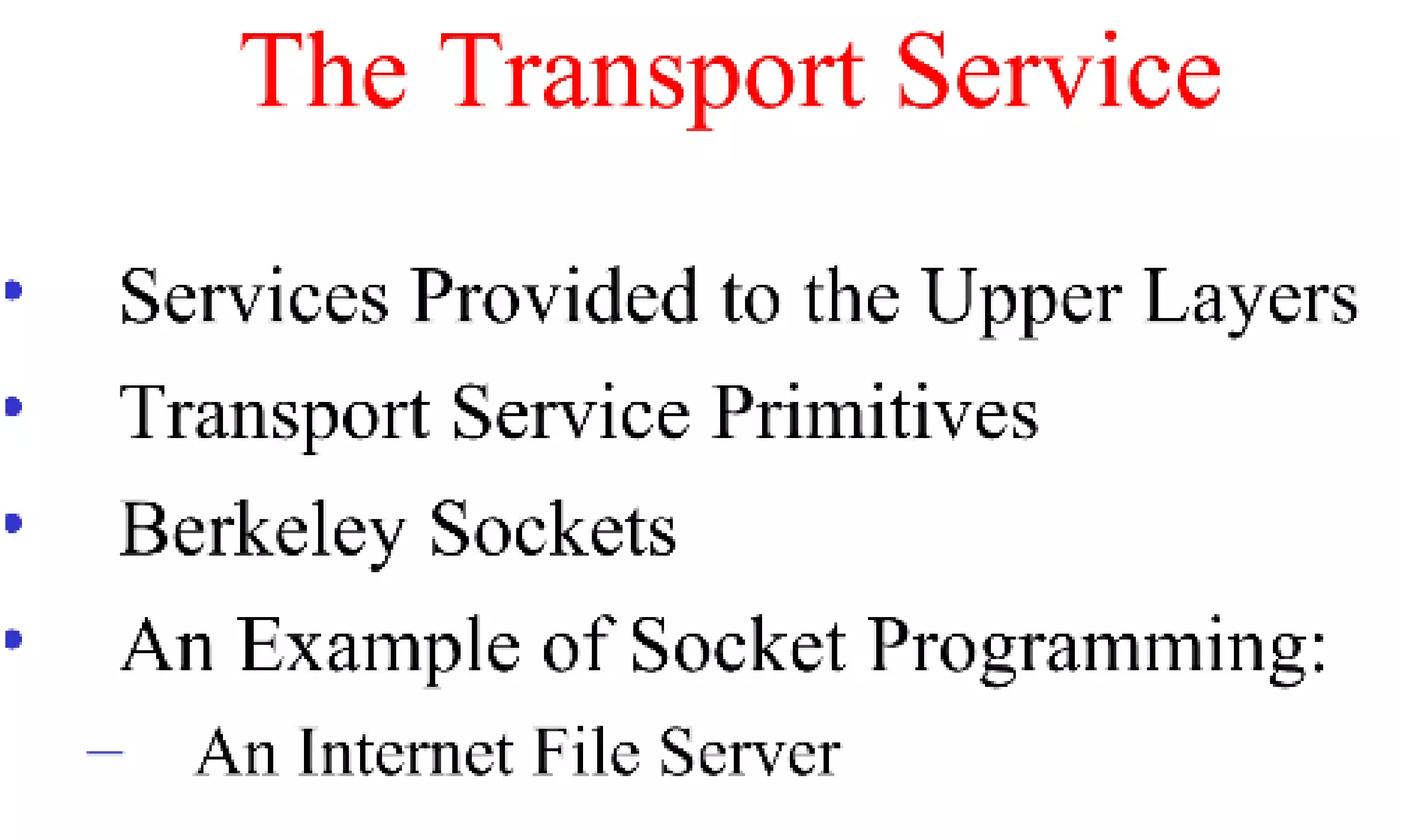
Data Communication & Networks

Chapter 6: Transport Layer (Textbook)

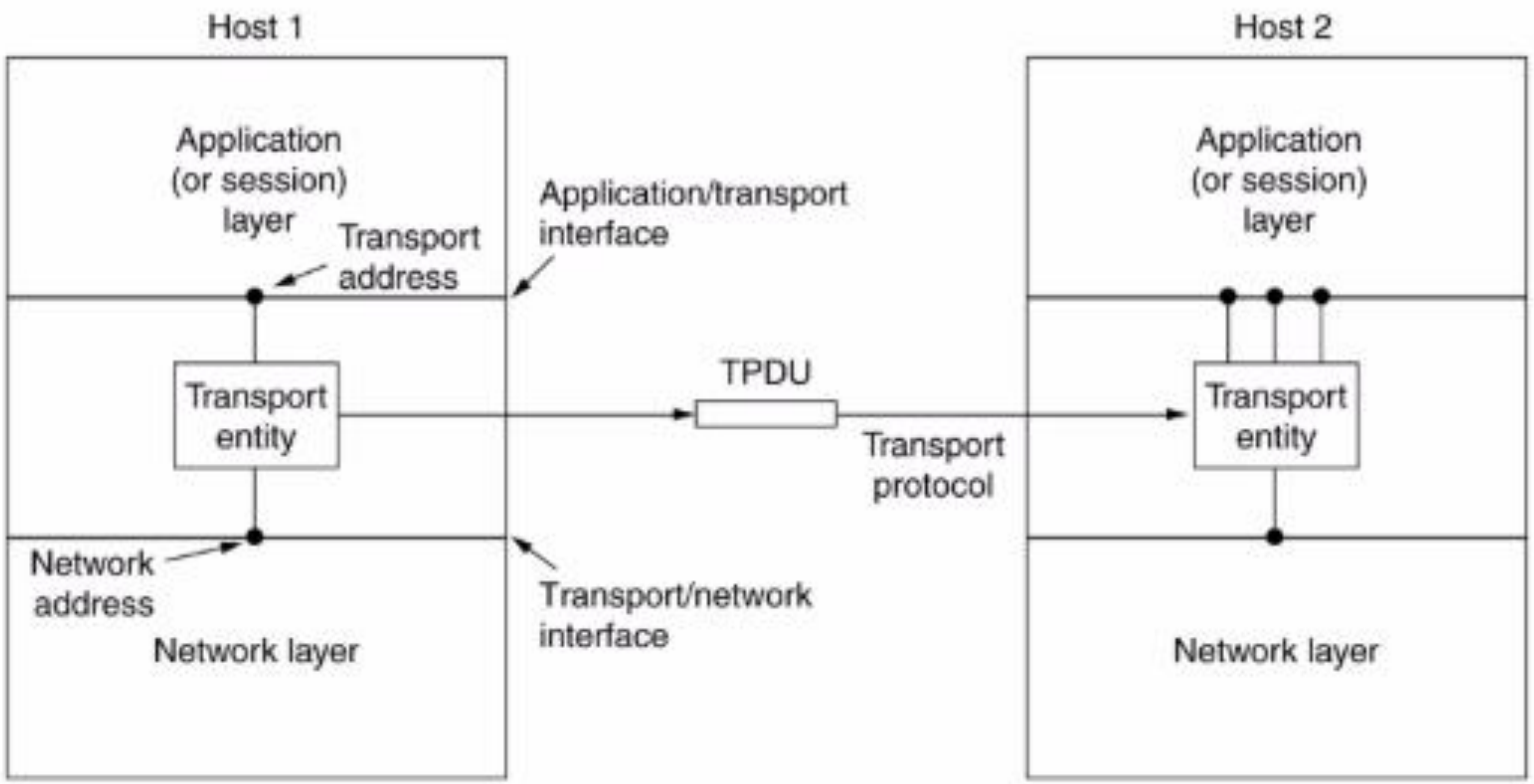
Transport Layer



The Transport Service

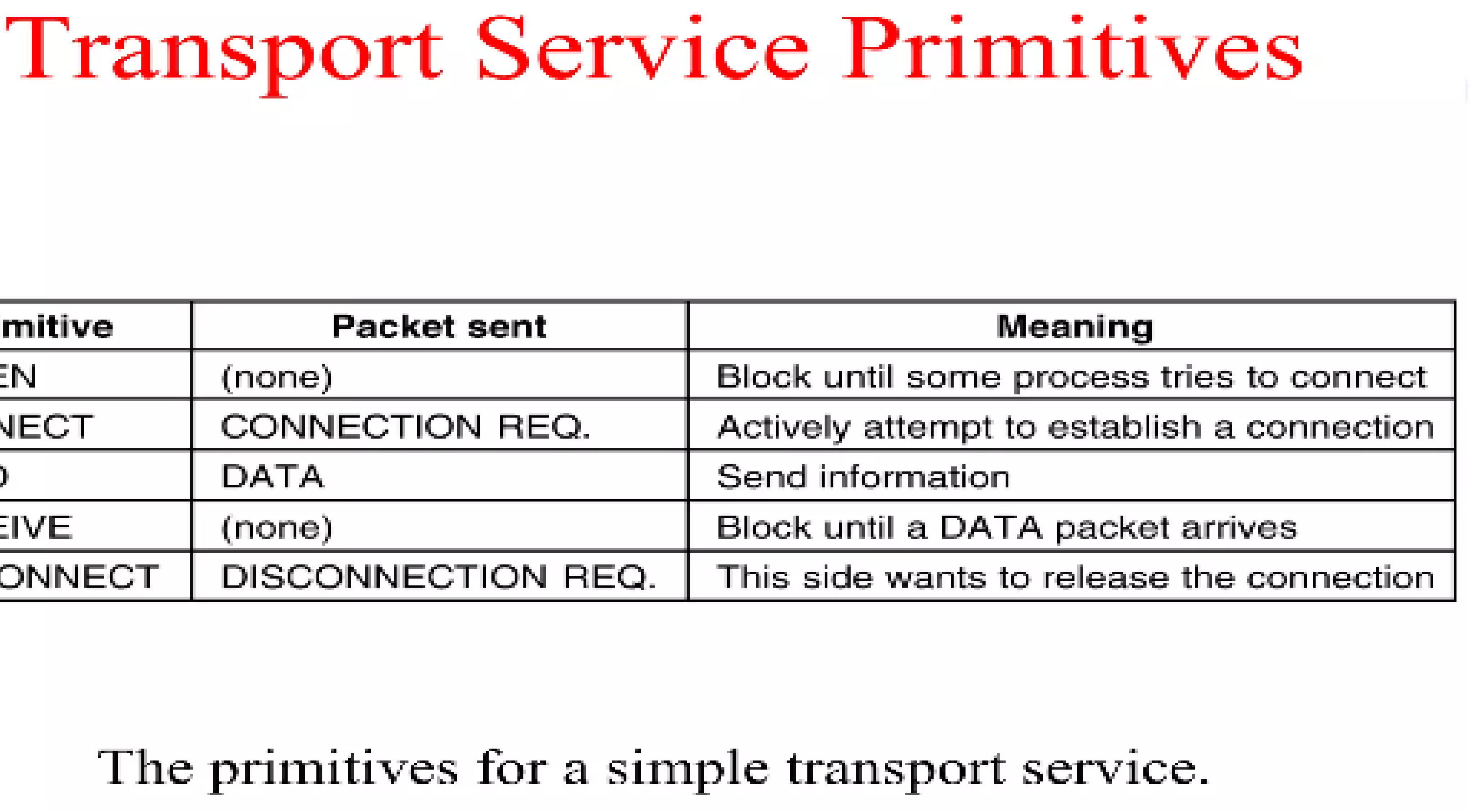
- Services Provided to the Upper Layers
- Transport Service Primitives
- Berkeley Sockets
- An Example of Socket Programming:
  - An Internet File Server

Services Provided to the Upper Layer



The network, transport, and application layers.

Transport Service Primitives

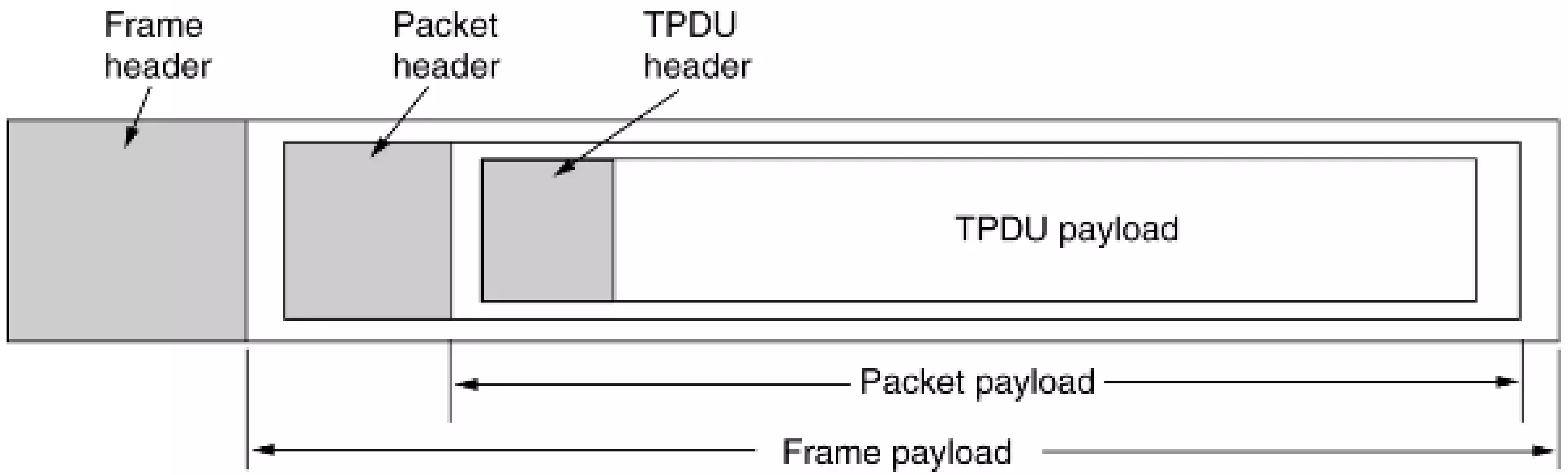

Transport Service Primitives

Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

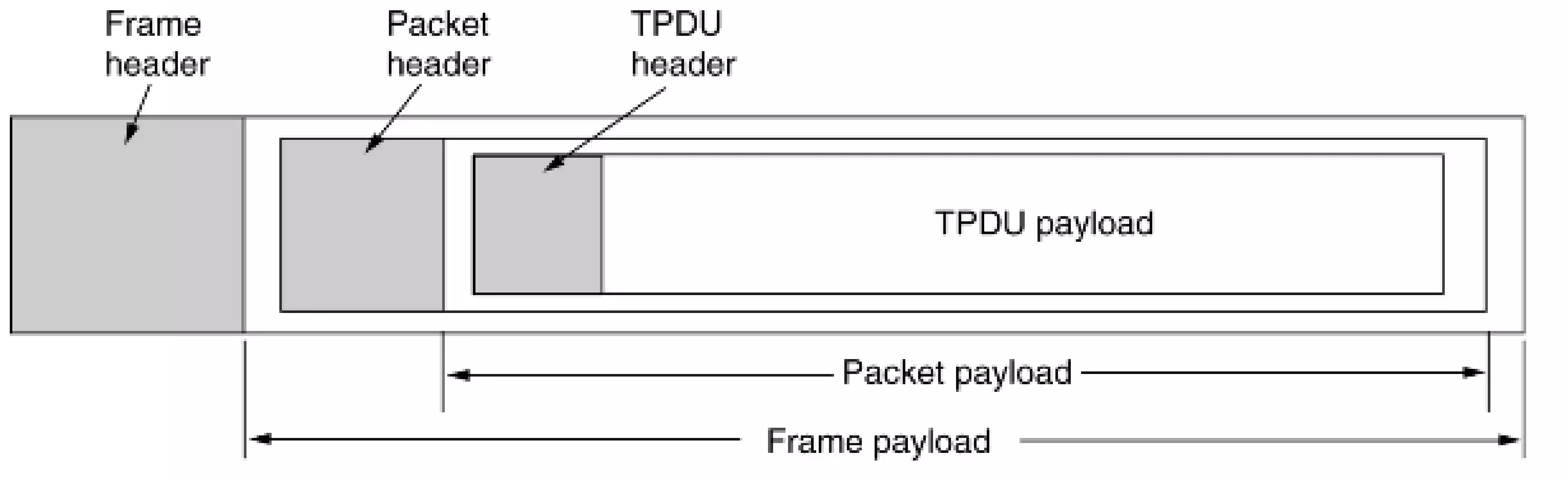
The primitives for a simple transport service.

- **A quick note on terminology is now in order. For lack of a better term, we will use the term segment for messages sent from transport entity to transport entity. TCP, UDP and other Internet protocols use this term. Some older protocols used the ungainly name TPDU (Transport Protocol Data Unit)**

Transport Service Primitives

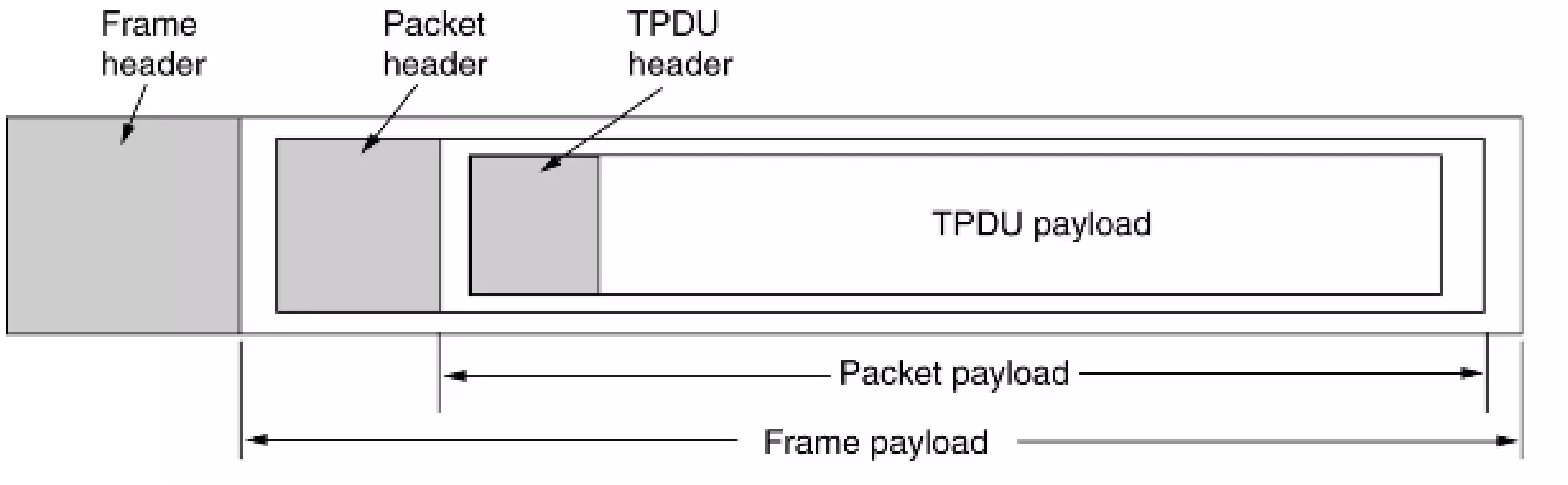


Transport Service Primitives



- Getting back to our client-server example, the client's CONNECT call causes a CONNECTION REQUEST segment to be sent to the server. When it arrives, the transport entity checks to see that the server is blocked on a LISTEN (i.e., is ready to handle requests). If so, it then unblocks the server and sends a CONNECTION ACCEPTED segment back to the client. When this segment arrives, the client is unblocked and the connection is established.

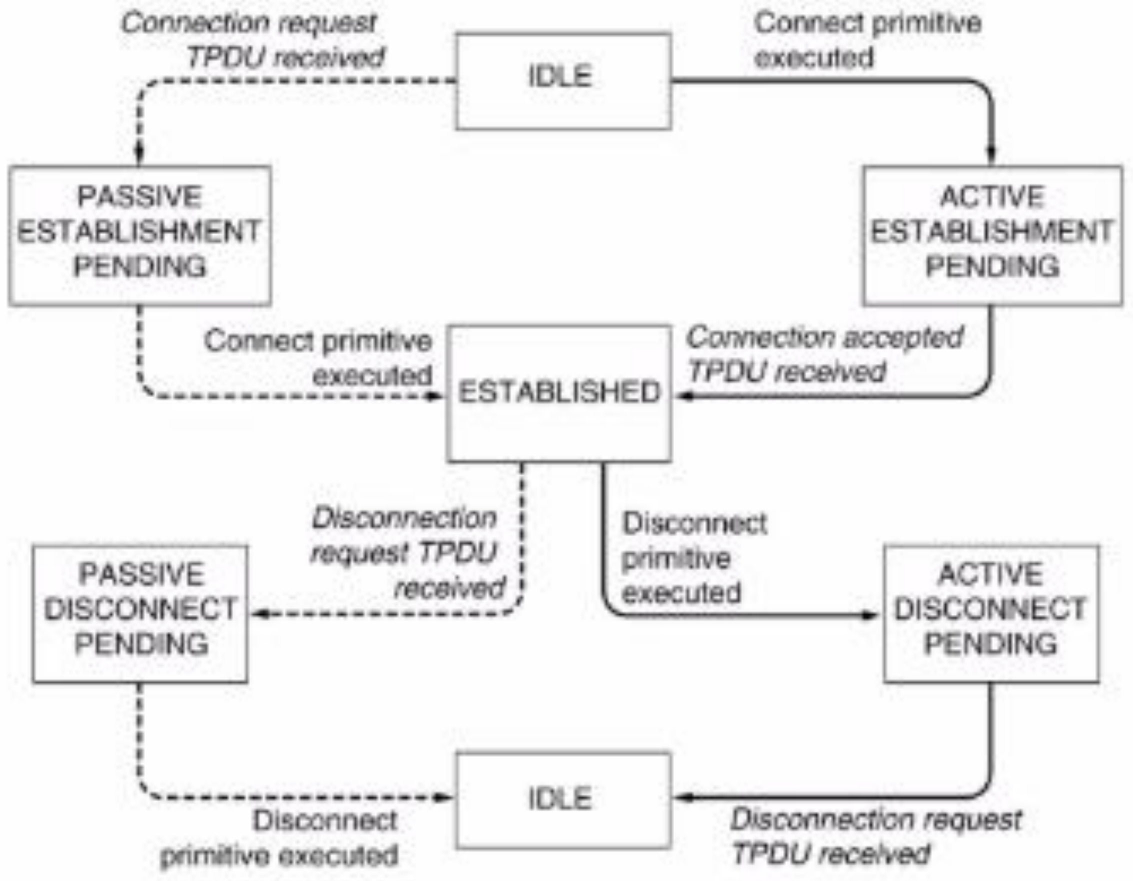
Transport Service Primitives



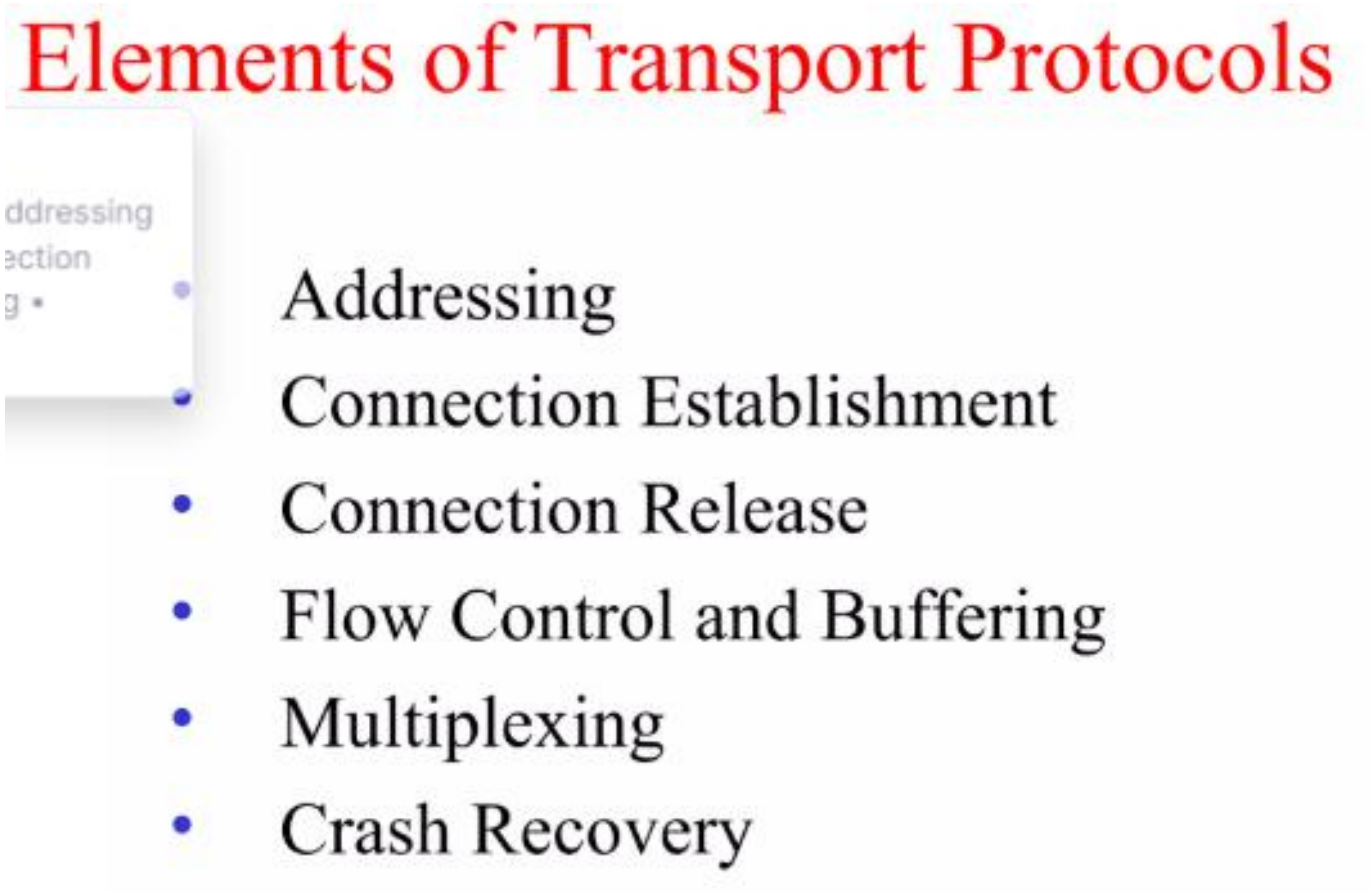
- Every data segment is acknowledged in Transport Layer.

Transport Service Primitives

Transport Service Primitives (3)


Connection request TPDU received
IDLE
Connect primitive executed
PASSIVE ESTABLISHMENT PENDING
ACTIVE ESTABLISHMENT PENDING
Connect primitive executed
ESTABLISHED
Connection accepted TPDU received
Disconnection request TPDU received
Disconnect primitive executed
PASSIVE DISCONNECT PENDING
ACTIVE DISCONNECT PENDING
IDLE
Disconnect primitive executed
Disconnection request TPDU received

Transport Protocol Elements



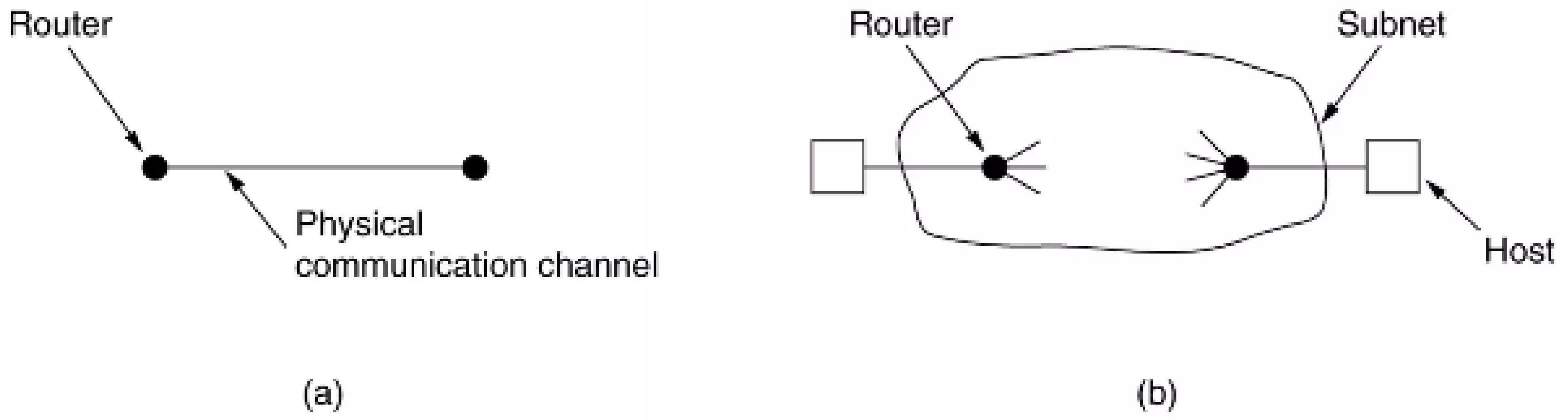
Elements of Transport Protocols

ddressing
ection
g •

- Addressing
- Connection Establishment
- Connection Release
- Flow Control and Buffering
- Multiplexing
- Crash Recovery

Transport Protocol Elements

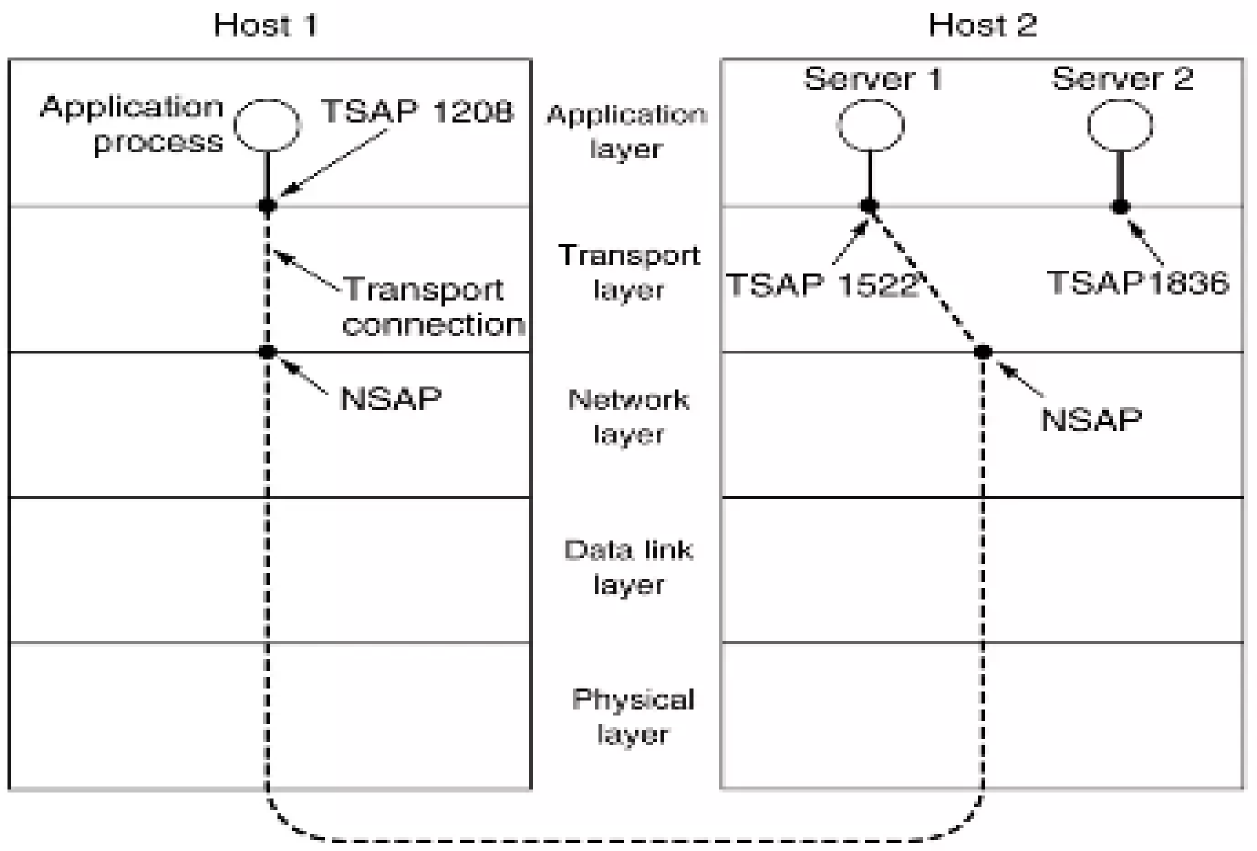
Transport Protocol



(a) Environment of the data link layer.
(b) Environment of the transport layer.

Transport Protocol Elements

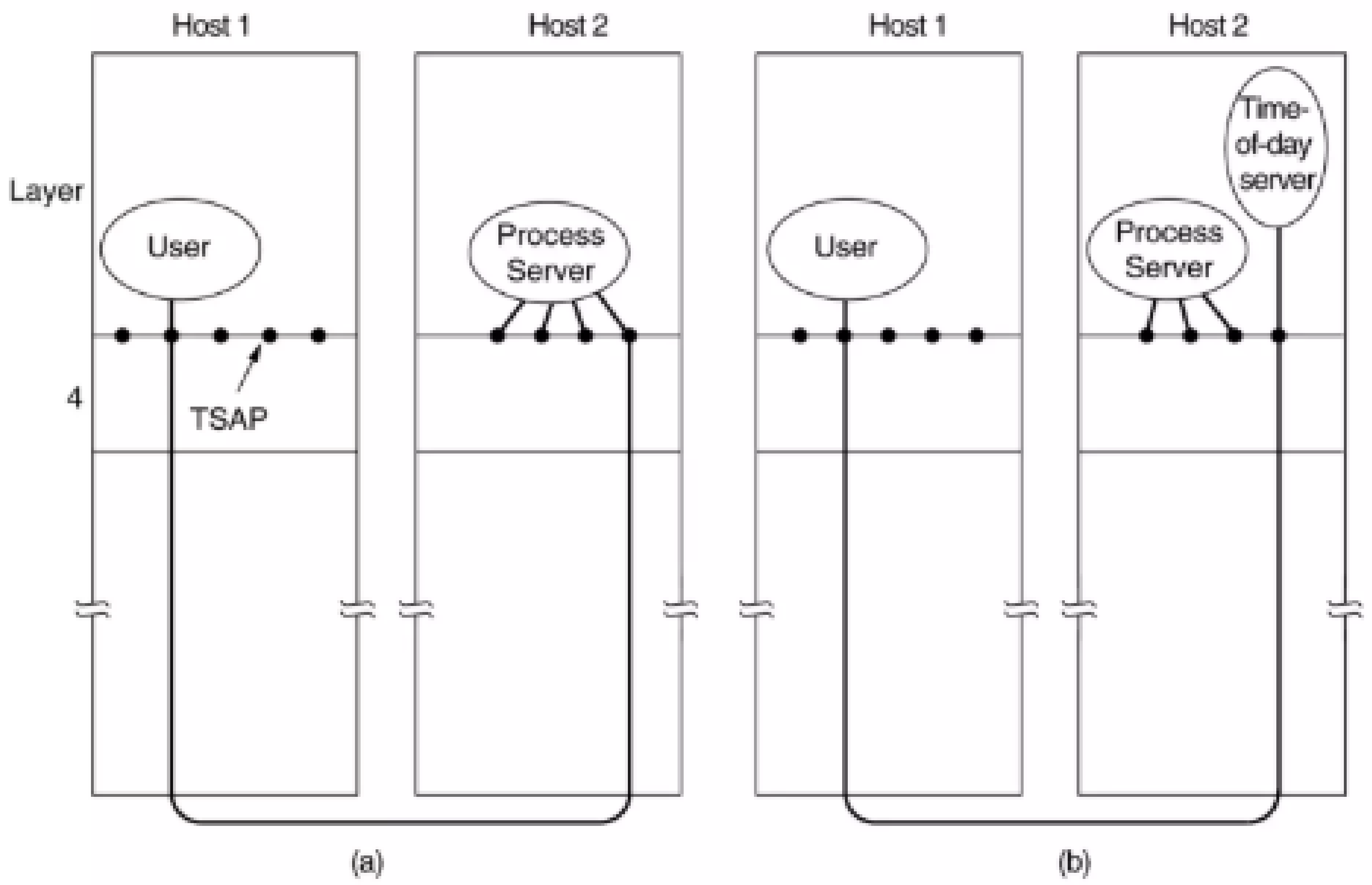
Addressing



TSAPs, NSAPs and transport connections.

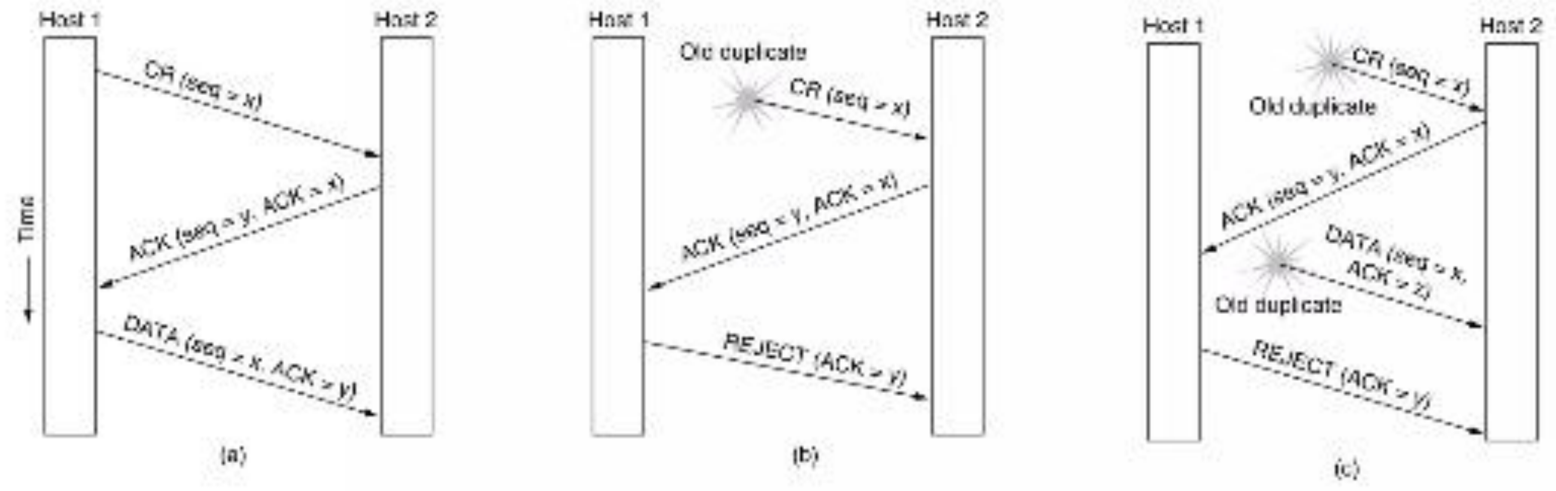
Transport Protocol Elements

Connection Establishment



Transport Protocol Elements

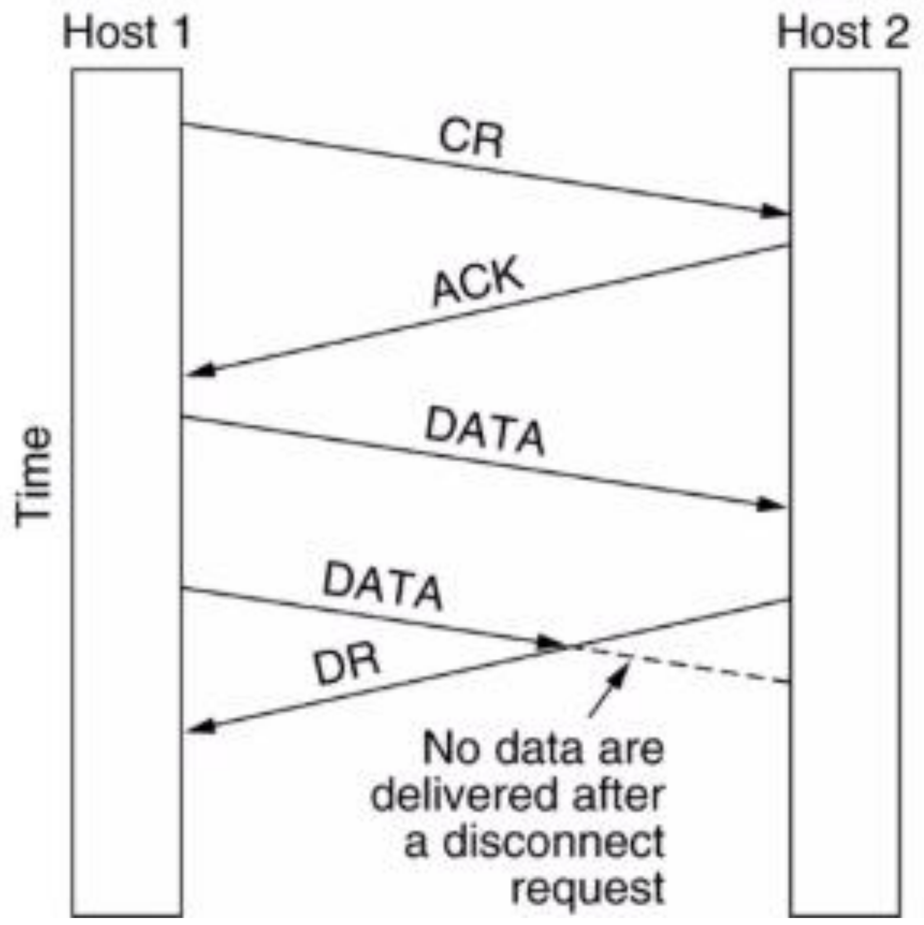
Connection Establishment (3)



Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNECTION REQUEST
(a) Normal operation,
(b) Old CONNECTION REQUEST appearing out of nowhere.
(c) Duplicate CONNECTION REQUEST and duplicate ACK.

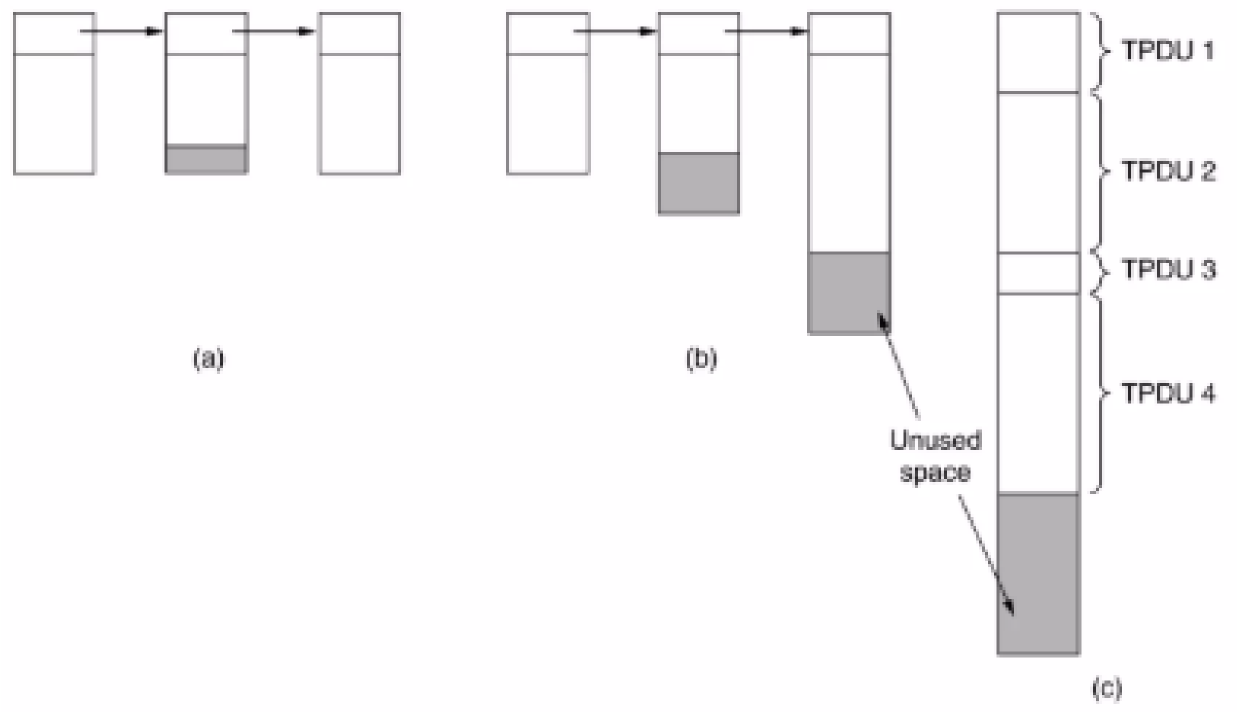
Transport Protocol Elements

Connection Release



Transport Protocol Elements

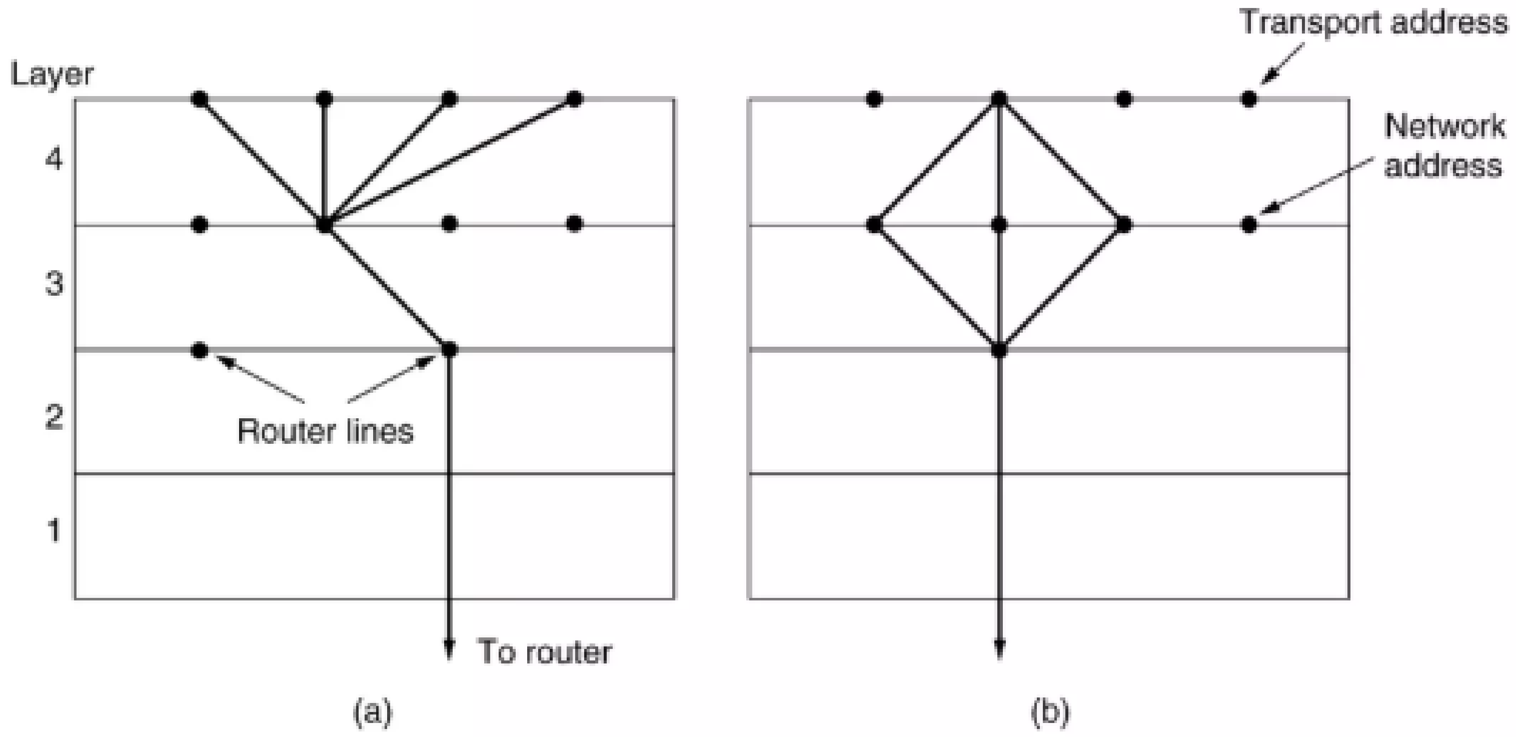
Flow Control and Buffering



(a) Chained fixed-size buffers. (b) Chained variable-sized buffers.
(c) One large circular buffer per connection.

Transport Protocol Elements

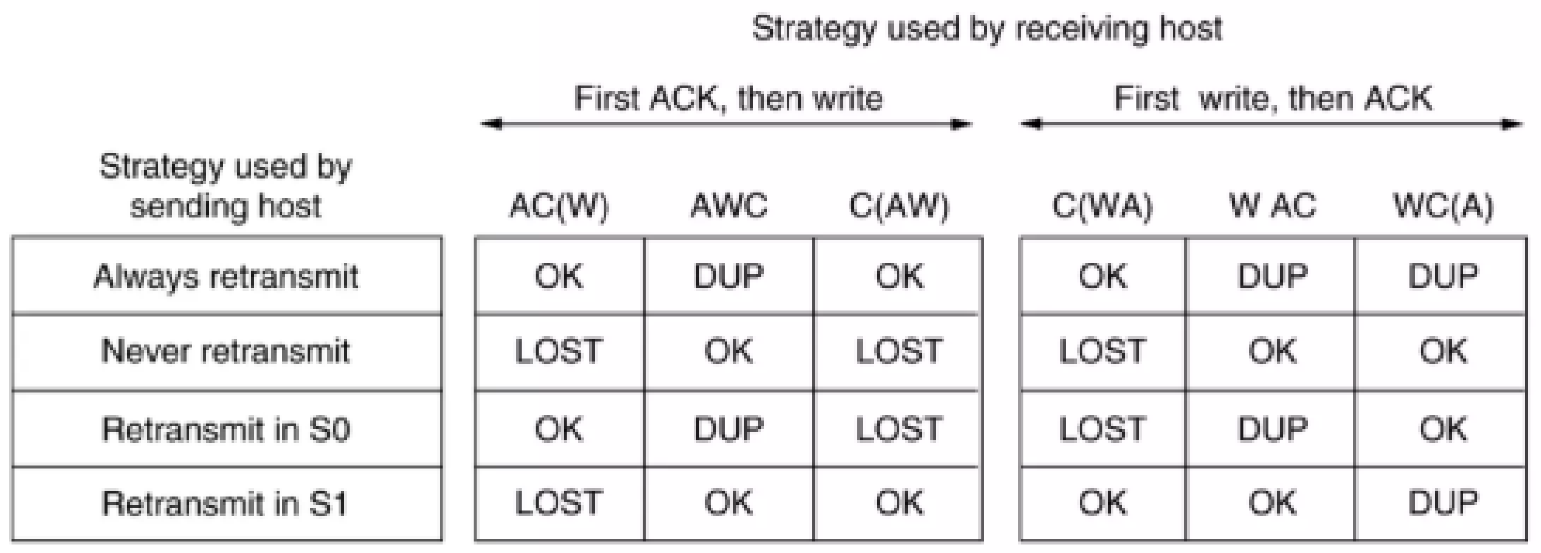
Multiplexing



(a) Upward multiplexing. (b) Downward multiplexing.

Transport Protocol Elements

Crash Recovery



	Strategy used by receiving host					
	First ACK, then write			First write, then ACK		
Strategy used by sending host	AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Always retransmit	OK	DUP	OK	OK	DUP	DUP
Never retransmit	LOST	OK	LOST	LOST	OK	OK
Retransmit in S0	OK	DUP	LOST	LOST	DUP	OK
Retransmit in S1	LOST	OK	OK	OK	OK	DUP

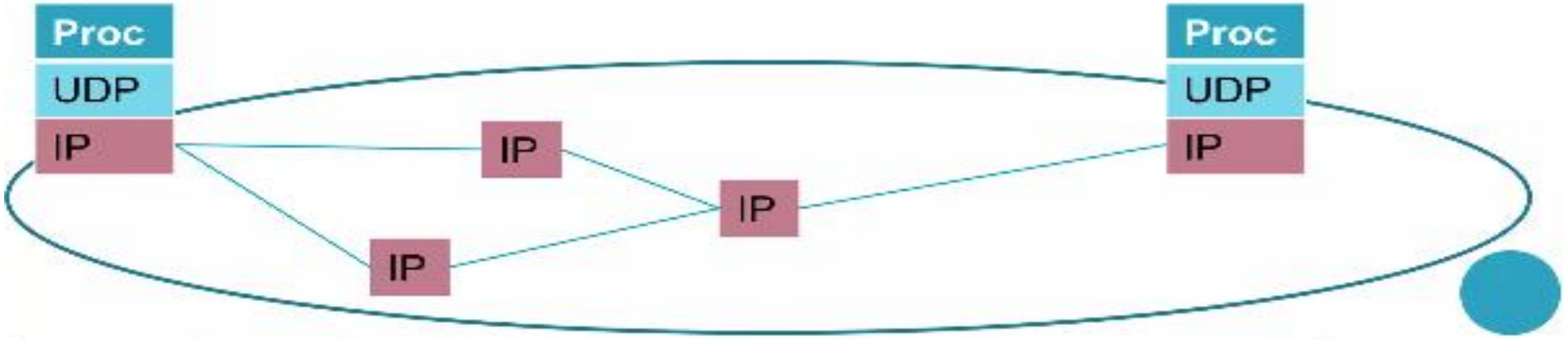
OK	= Protocol functions correctly
DUP	= Protocol generates a duplicate message
LOST	= Protocol loses a message

Different combinations of client and server strategy.

UDP Protocol

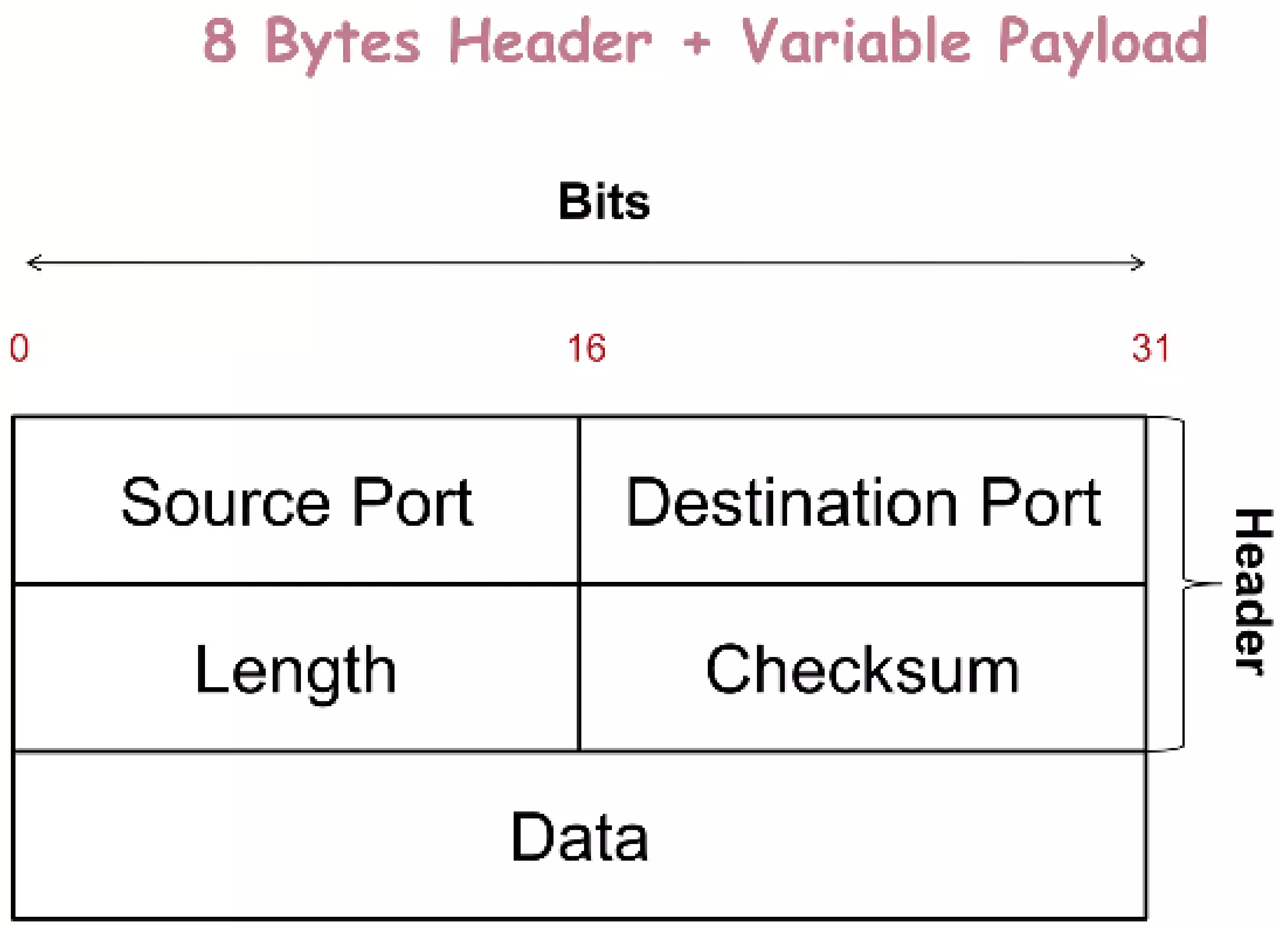
Layer 4 (Transport Layer) protocol

- Runs over IP
- Unreliable i.e. Best Effort service
  - UDP segments may be:
    - lost
    - delivered out of order
- Connectionless
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others



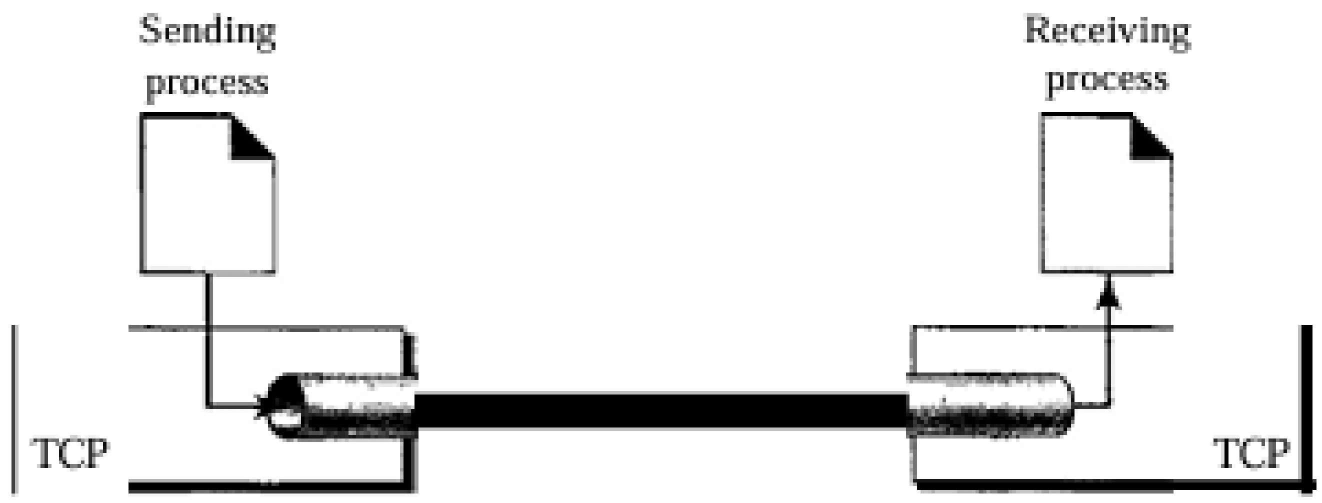
Provides a *lossy* connection (data may vanish).

UDP Protocol



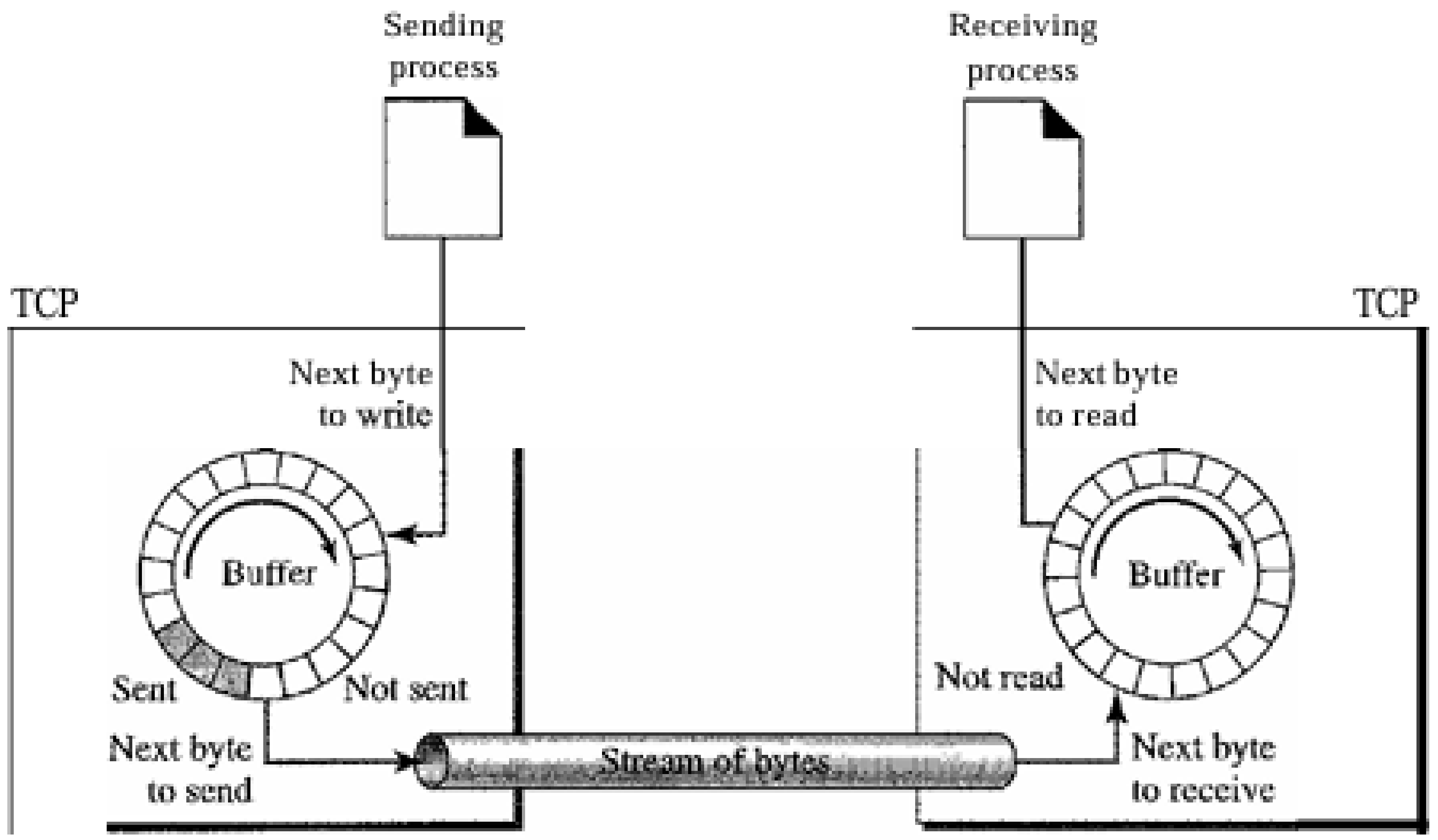
TCP Protocol (Behrouz Ferozan)

Figure 23.13 *Stream delivery*



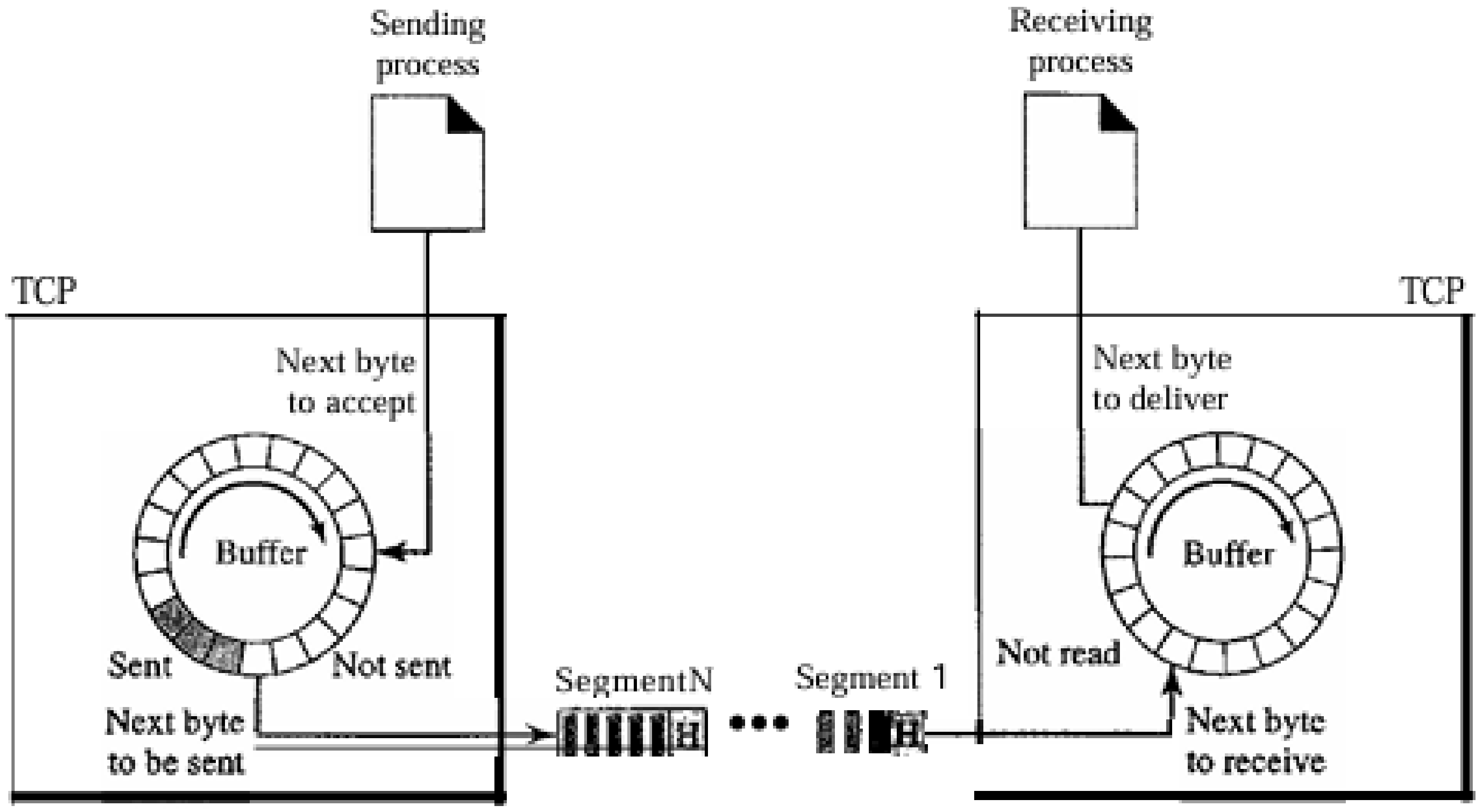
TCP Protocol (Behrouz Ferozan)

Figure 23.14 *Sending and receiving buffers*



TCP Protocol (Behrouz Ferozan)

Figure 23.15 *TCP segments*



TCP Protocol: Flow Control

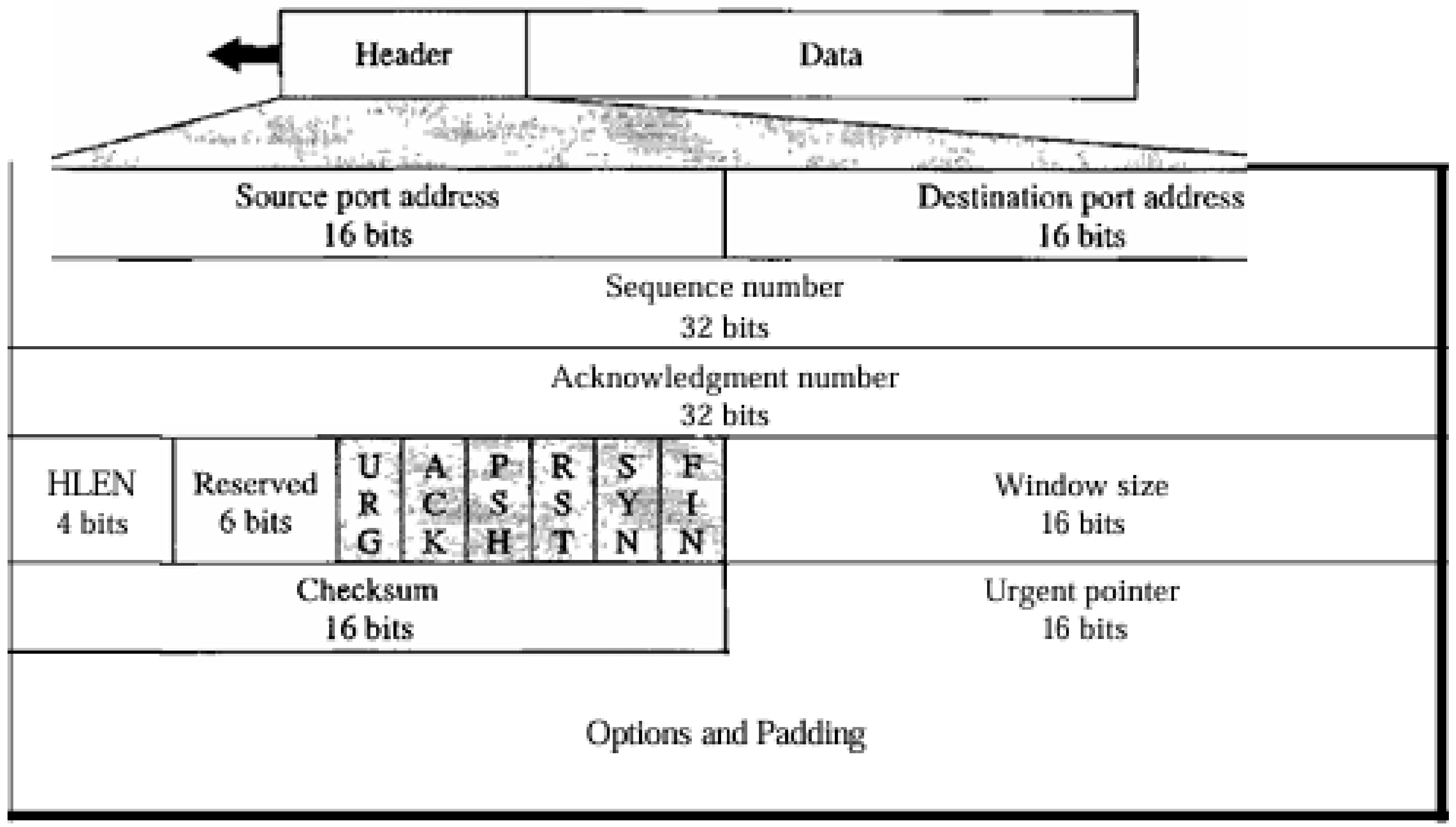
- **Flow Control:** TCP, Unlike UDP, provides flow control. The receiver of the data controls the amount of data that are to be sent by the sender. This is done to prevent the receiver from being over whelmed with data. The numbering system allows TCP to use a byte-oriented flow control.

- **Error Control:** To provide reliable service, TCP implements an error control mechanism. Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented.

- **Congestion Control:** TCP, unlike UDP, takes into account congestion in the network. The amount of data sent by a sender is not only controlled by the receiver (flow control), but is also detennined by the level of congestion in the network.

TCP Protocol Structure

Figure 23.16 *TCP segment format*



TCP Protocol Structure

- **Sequence number:** This 32-bit field defines the number assigned to the first byte of data contained in this segment. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered.

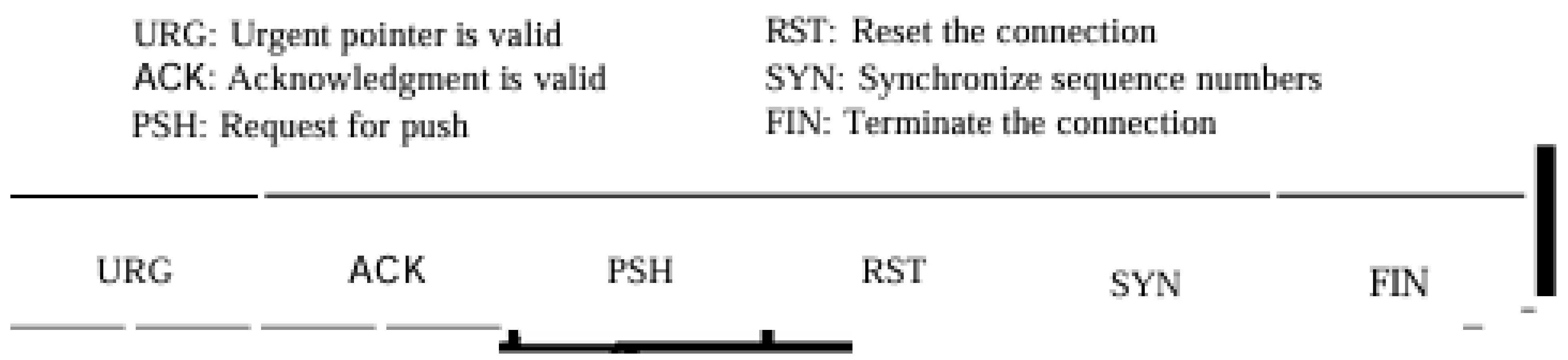
- **Acknowledgment number:** This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number x from the other party, it defines x + I as the acknowledgment number. Acknowledgment and data can be piggybacked together.

- **Header length:** This 4-bit field indicates the number of4-byte words in the TCP header.

TCP Protocol Structure

- **Control:** This field defines 6 different control bits or flags as shown in Figure 23.17. One or more of these bits can be set at a time.

Figure 23.17 *Control field*

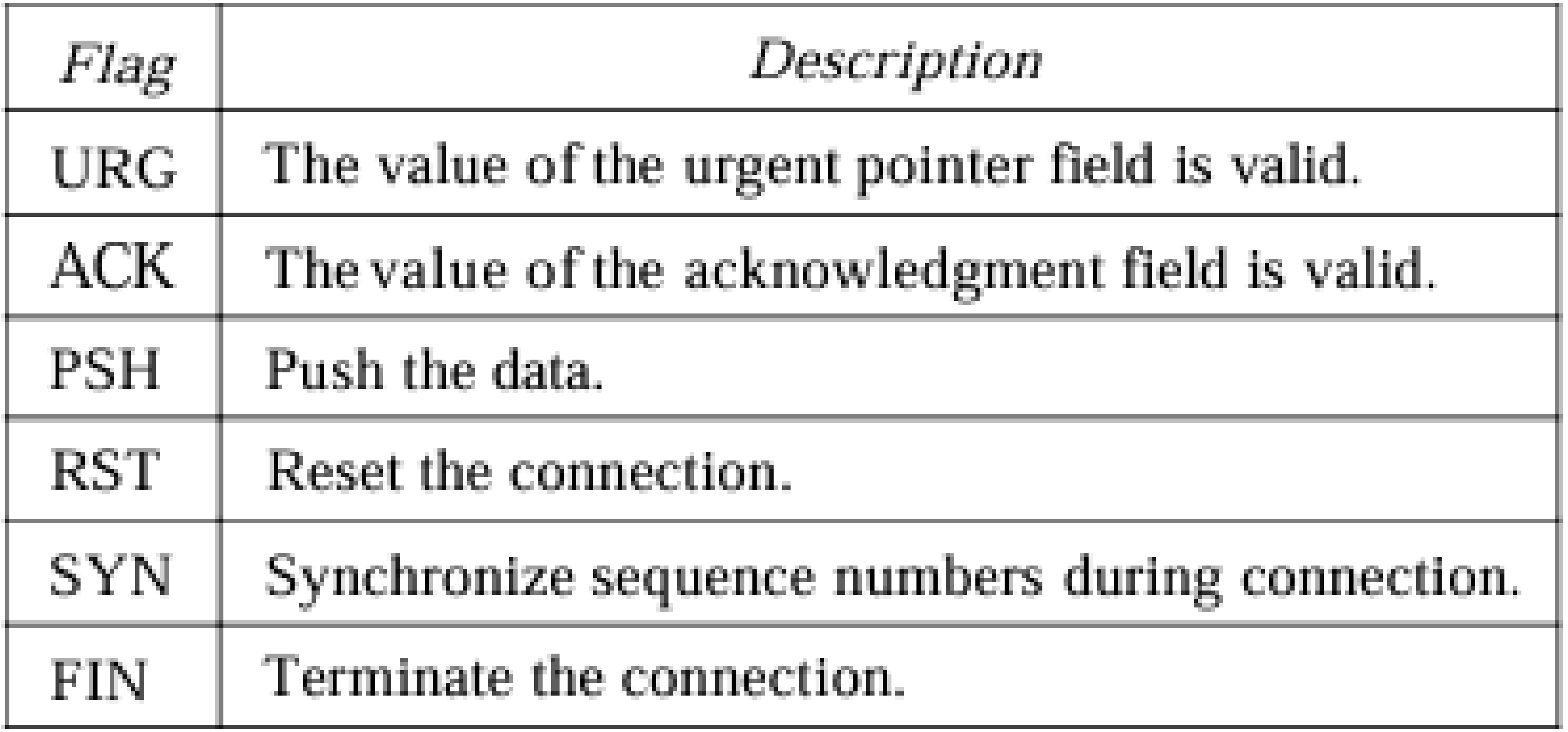


URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection
URG
ACK
PSH
RST
SYN
FIN

TCP Protocol Structure

- **Control**

Table 23.3 *Description offlags in the control field*



Flag	*Description*
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.

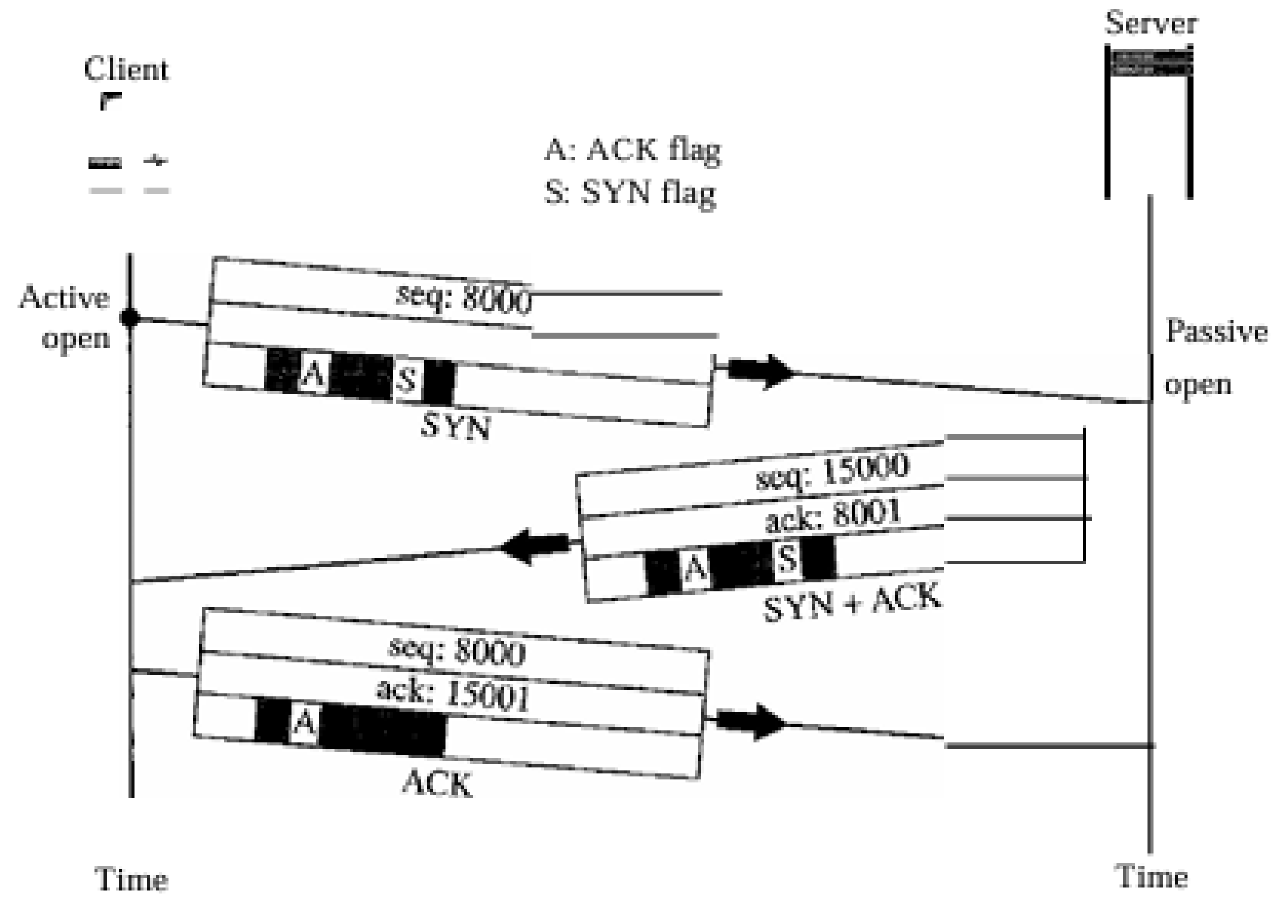
TCP Protocol Structure

- **Window size:** This field defines the size of the window, in bytes, that the other party must maintain. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window (rwnd) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.

- **Urgent pointer:** This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines the number that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment. Urgent data can be interrupt, abort etc.

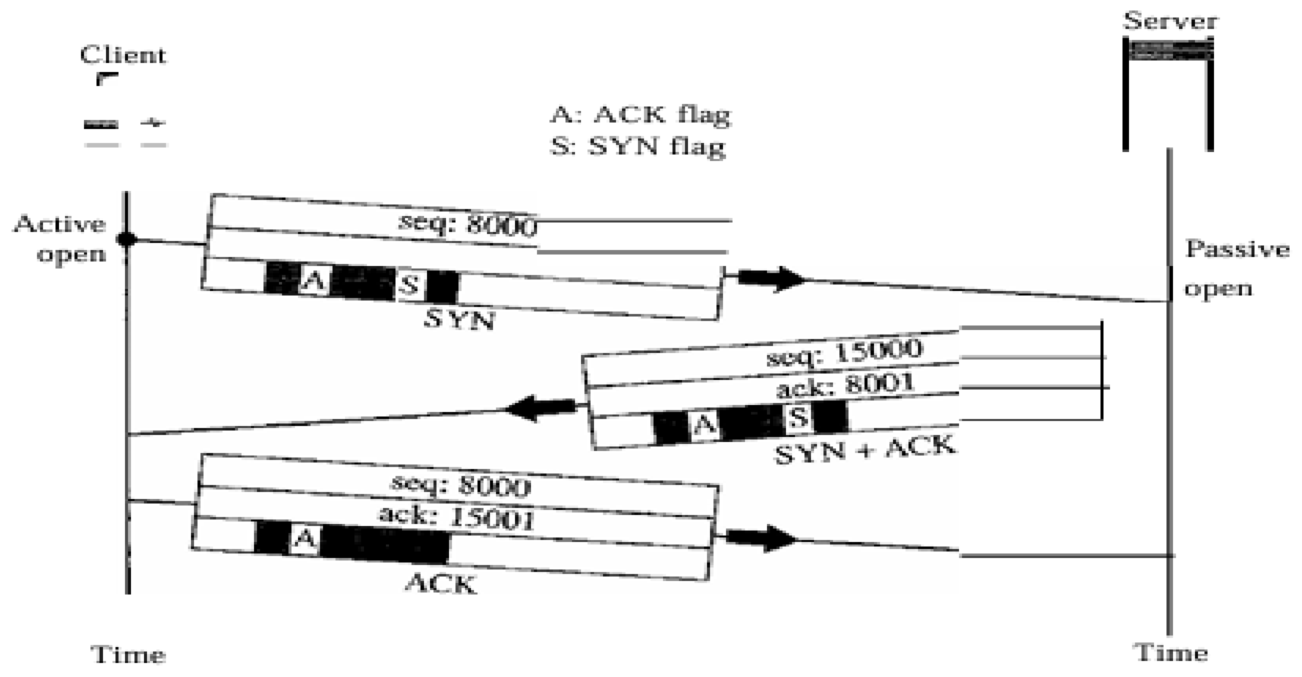
Connection Establishment in TCP Protocol

Figure 23.18 *Connection establishment using three-way handshaking*



Connection Establishment in TCP Protocol

Figure 23.18 *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.

Connection Establishment in TCP Protocol

- **Solve Example 23.4 and 23.5**