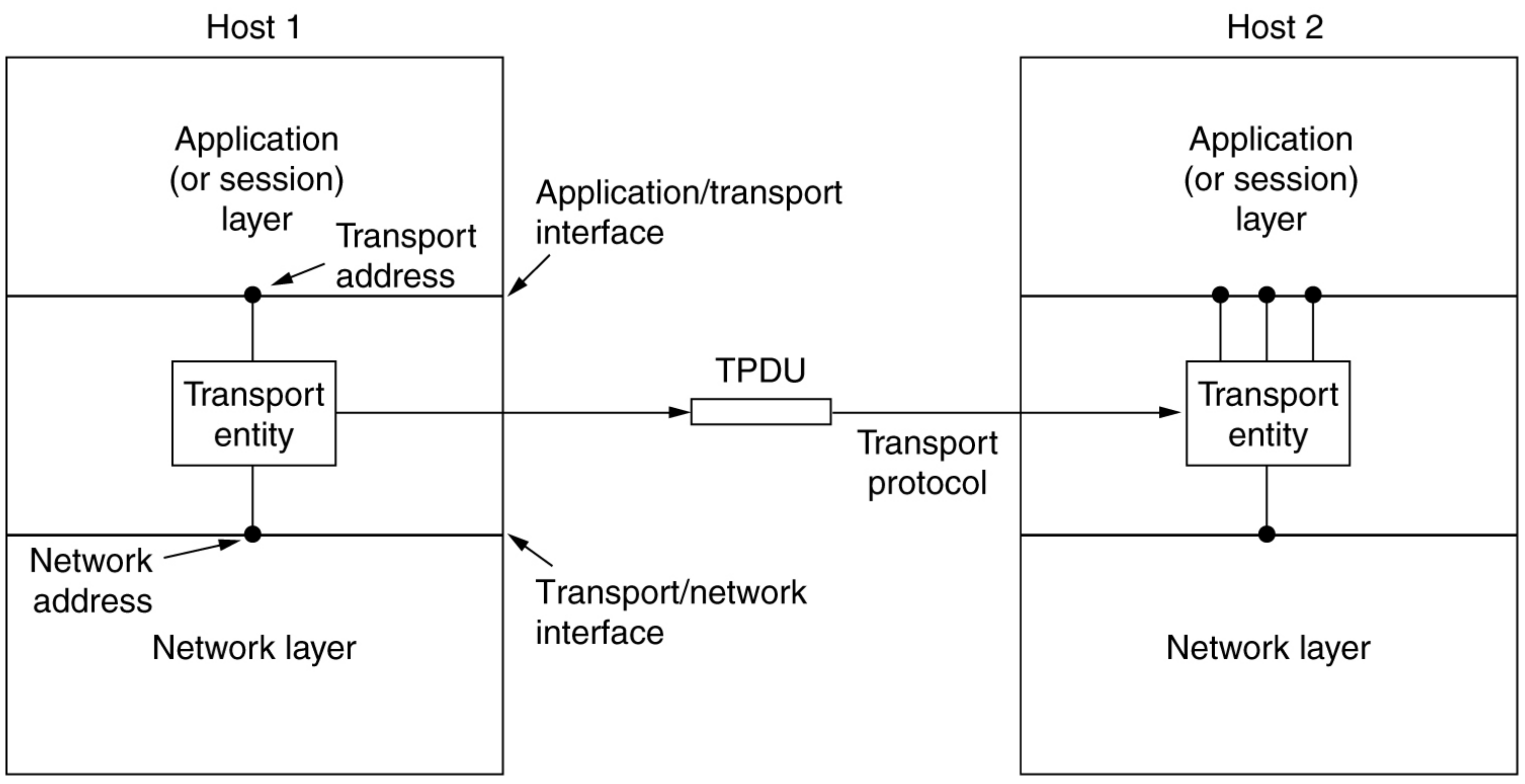
The Transport Layer

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 Layers



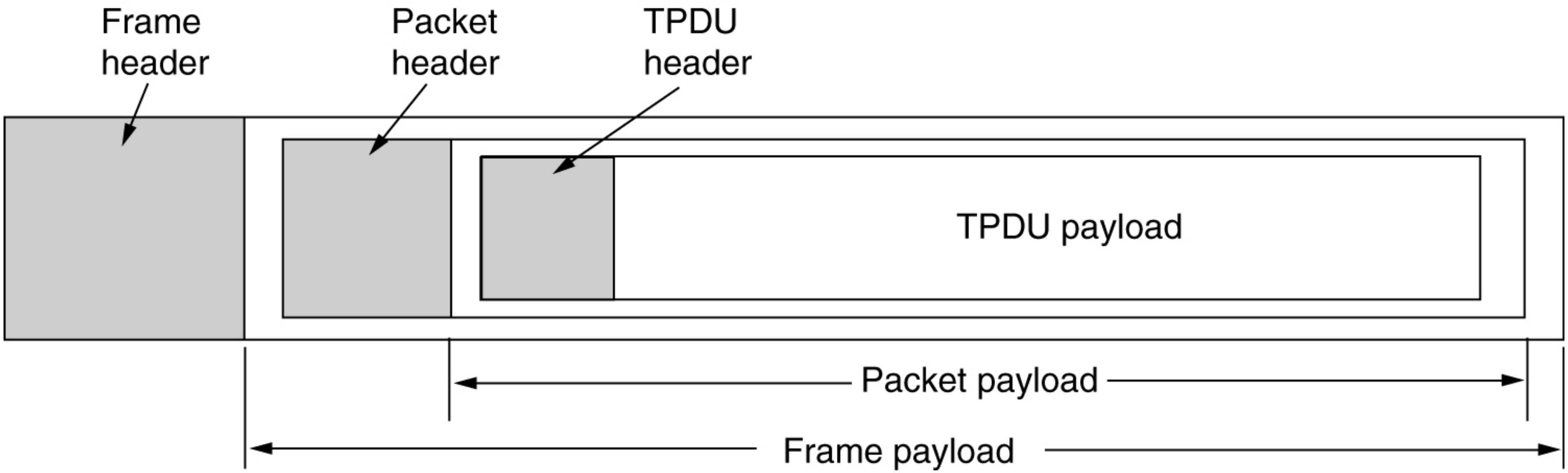
The network, transport, and application layers.

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

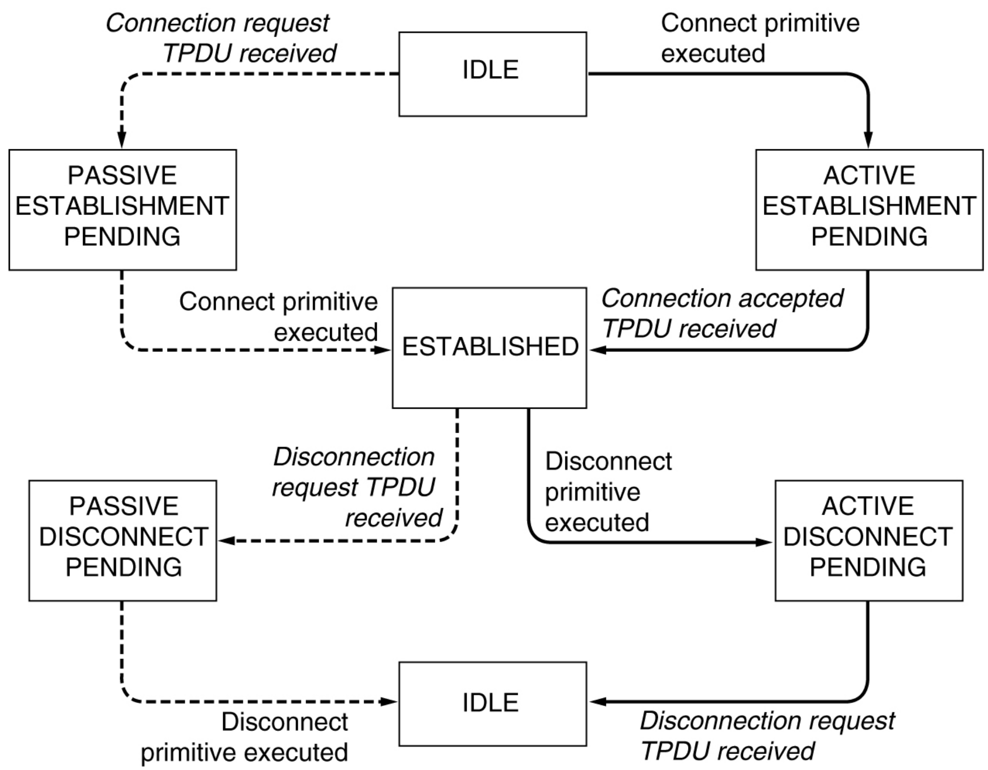
The primitives for a simple transport service.

Transport Service Primitives (2)



The nesting of TPDUs, packets, and frames.

Transport Service Primitives (3)



A state diagram for a simple connection management scheme. Transitions labeled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence.

Berkeley Sockets

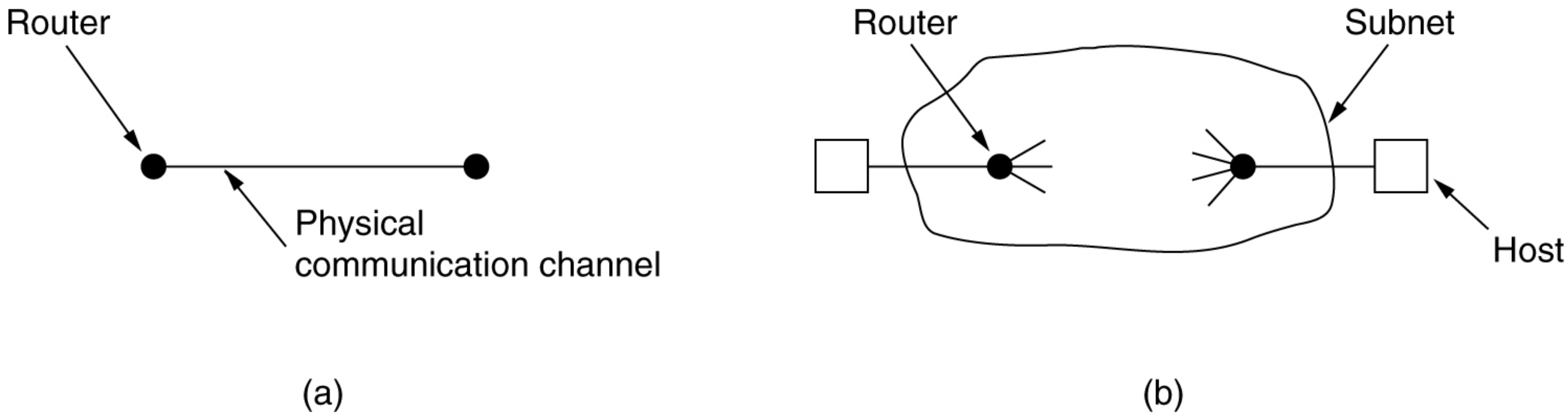
Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

The socket primitives for TCP.

Elements of Transport Protocols

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

Transport Protocol

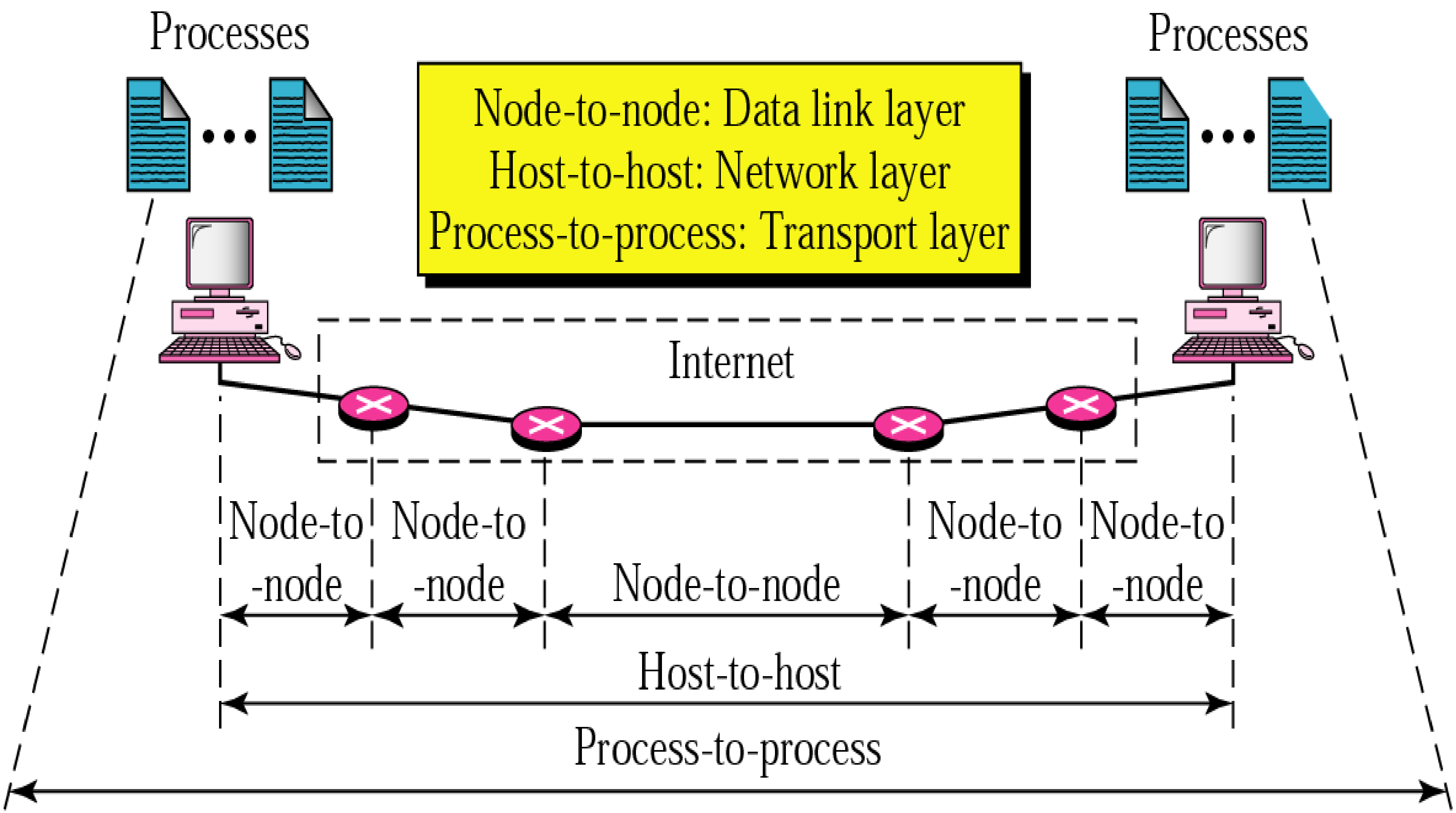


(a) Environment of the data link layer.
(b) Environment of the transport layer.

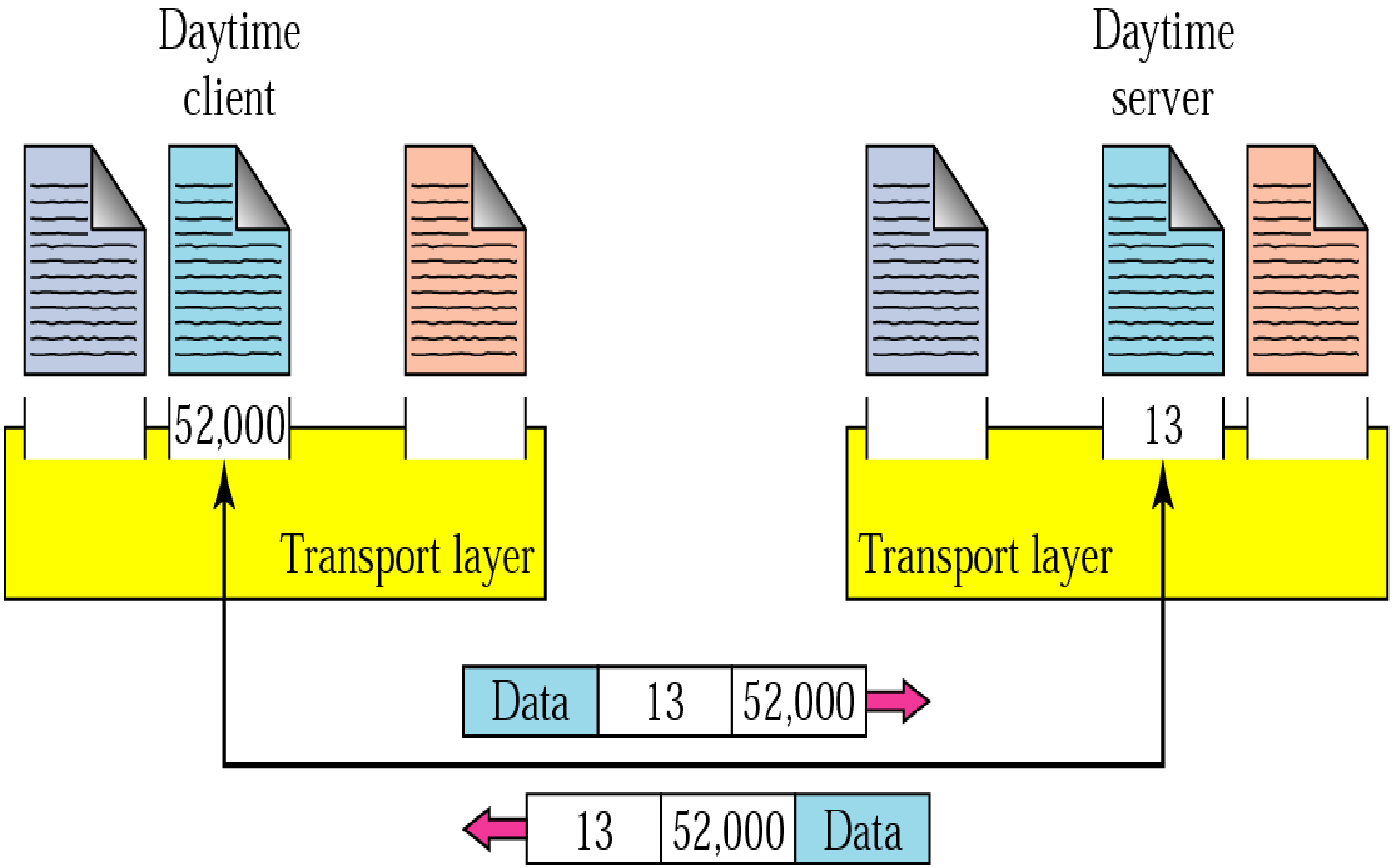
Note:

The transport layer is responsible for process-to-process delivery.

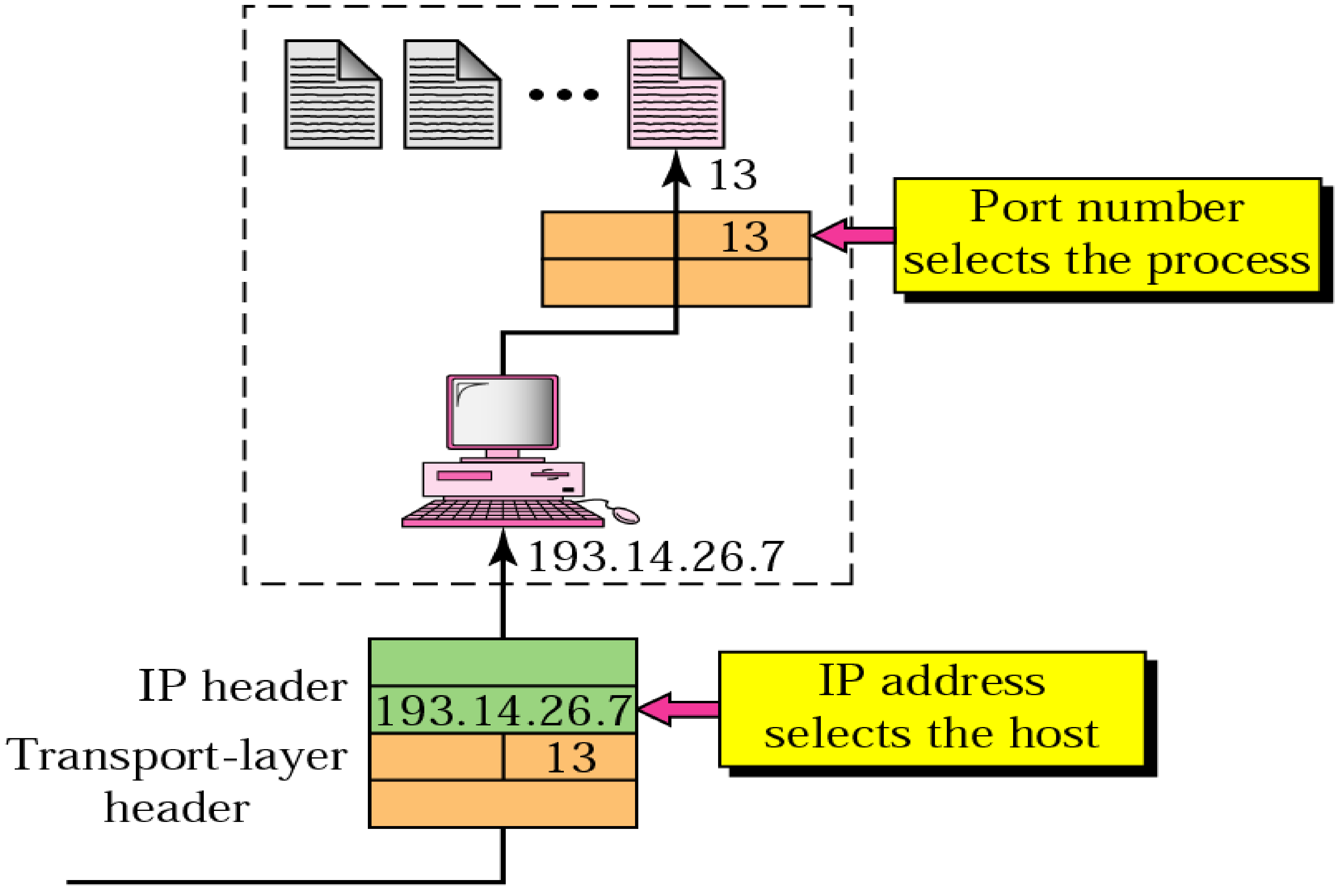
Types of data deliveries



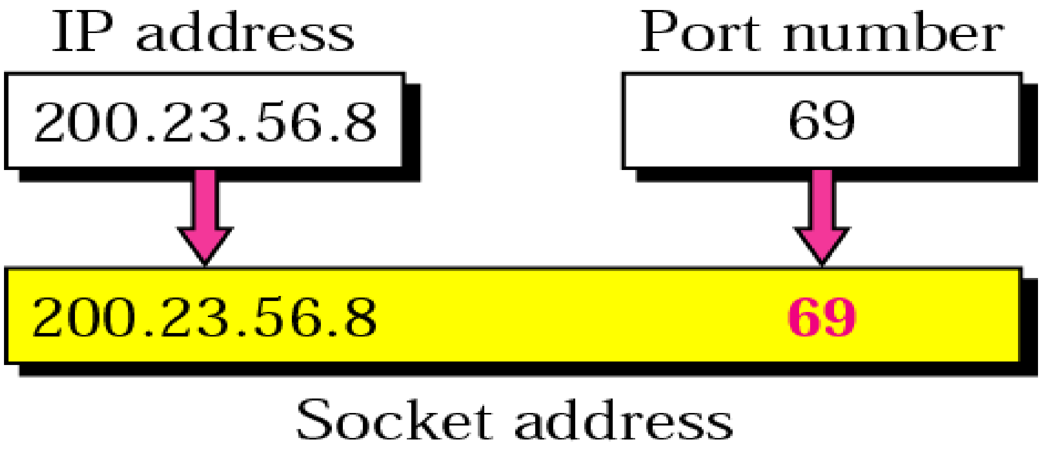
Port Numbers



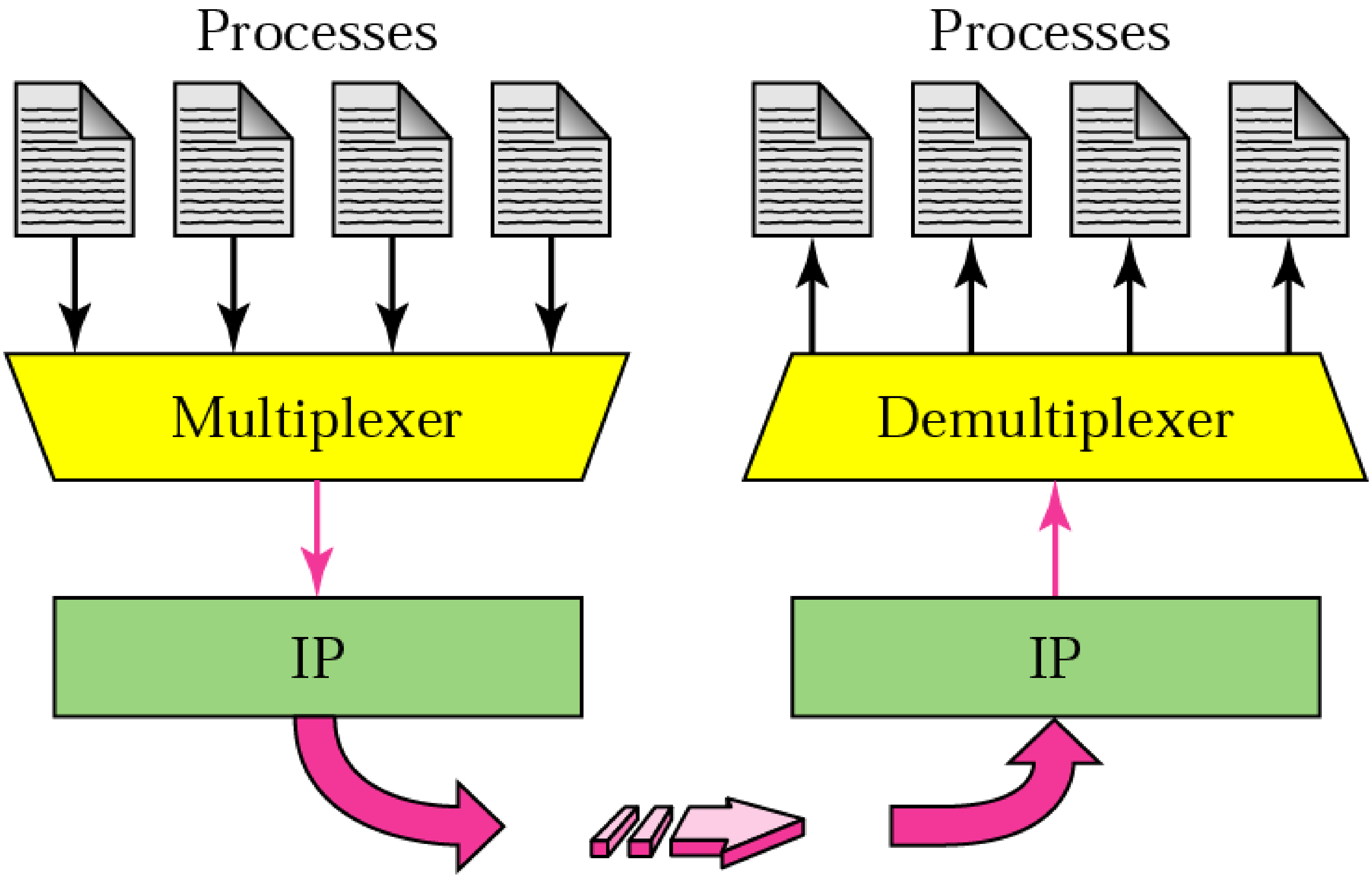
IP addresses versus port numbers



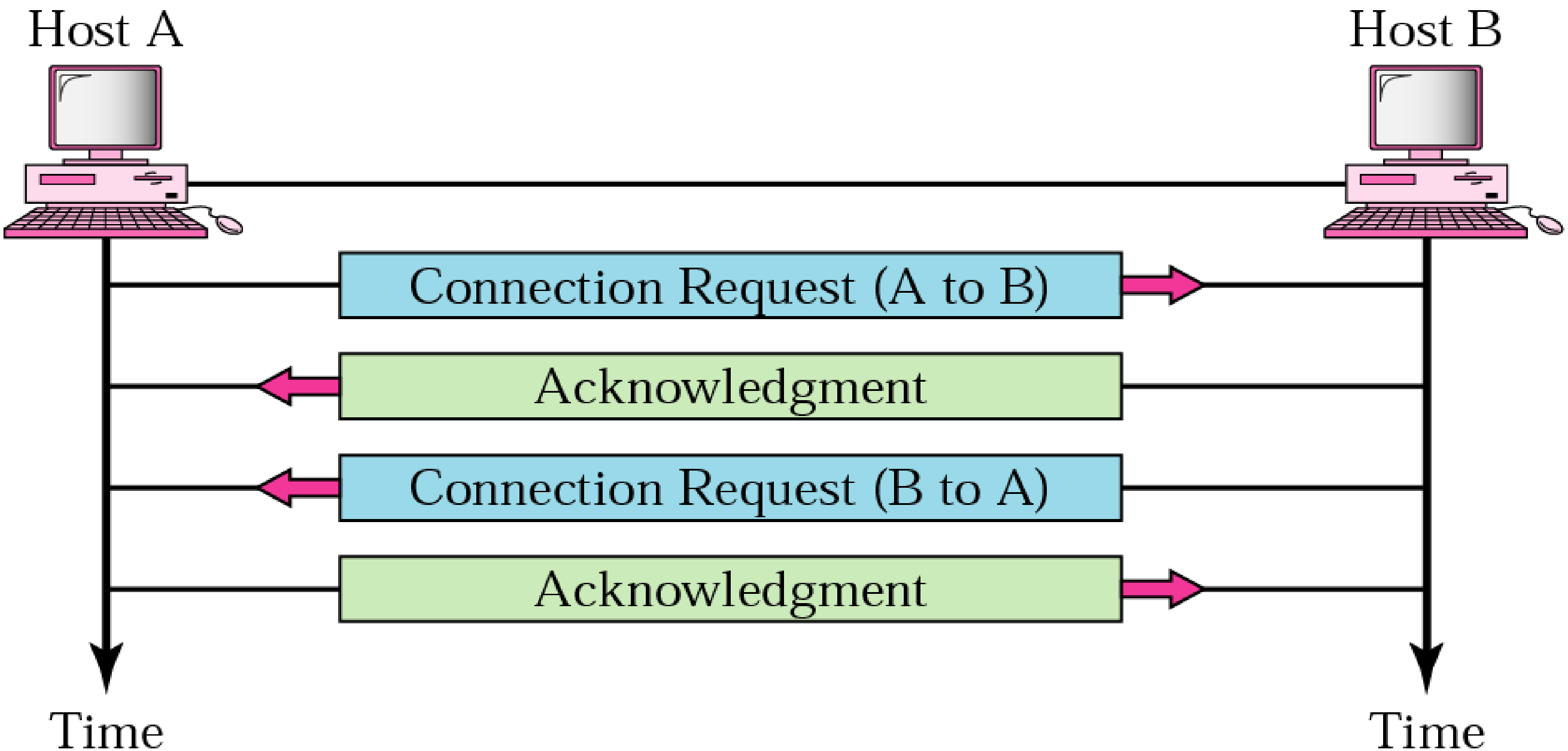
Socket Sample



Multiplexing & De-multiplexing



Connection Establishment (1)



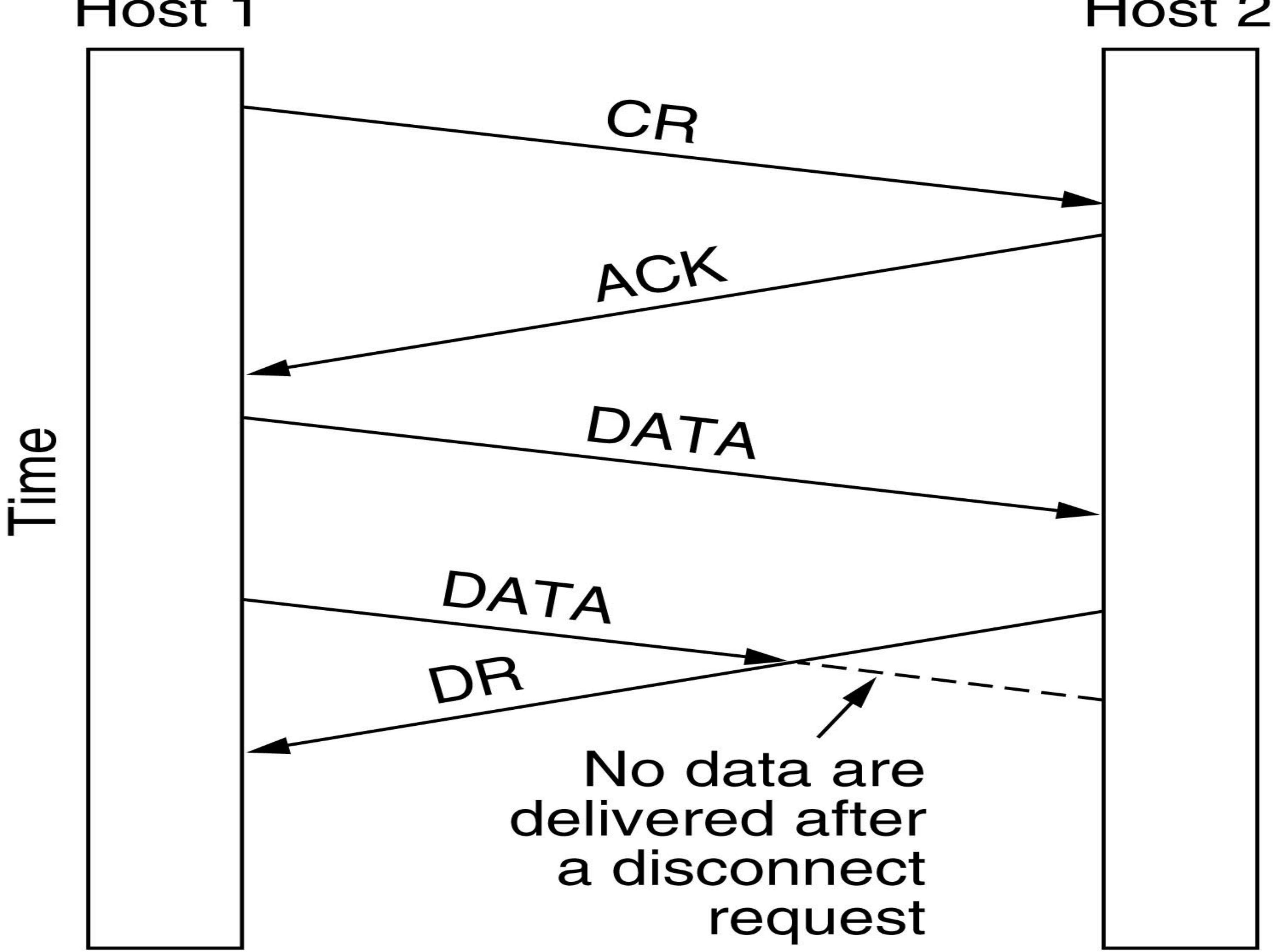
Connection Release

a) There are two styles of terminating a connection: asymmetric release and symmetric release.

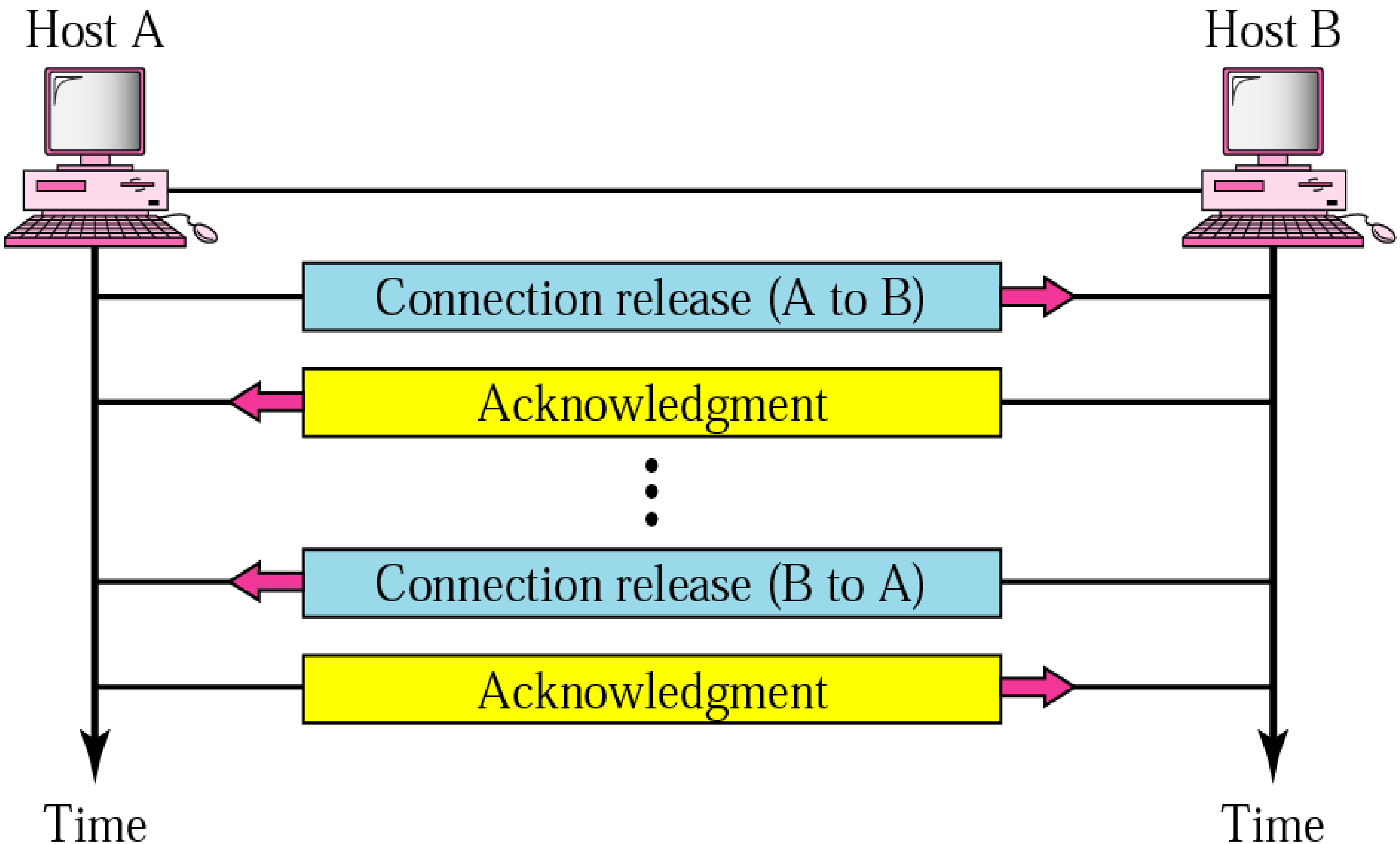
b) asymmetric release is the way the telephone system works: when one party hangs up, the connection is broken

c) symmetric release treats the connection as two separate unidirectional connections and requires each one to be released separately.

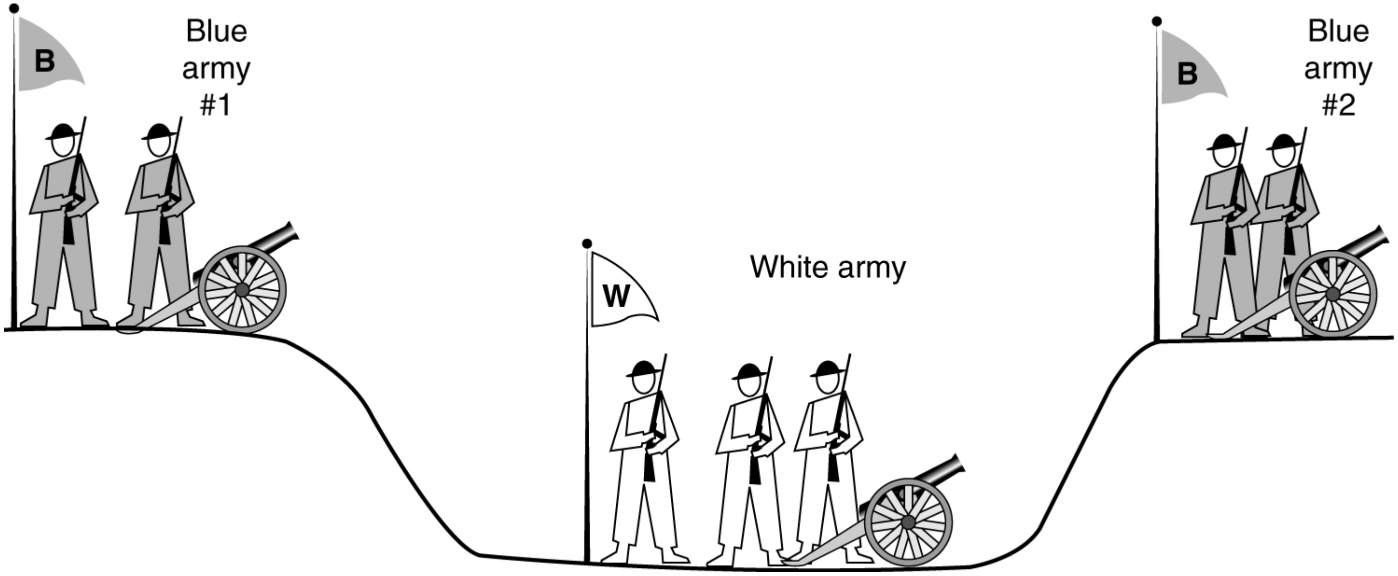
d) asymmetric release may result in loss of data:


Host 1
Host 2
CR
ACK
DATA
DATA
DR
Time
No data are delivered after a disconnect request

Connection Release



Connection Release (2)

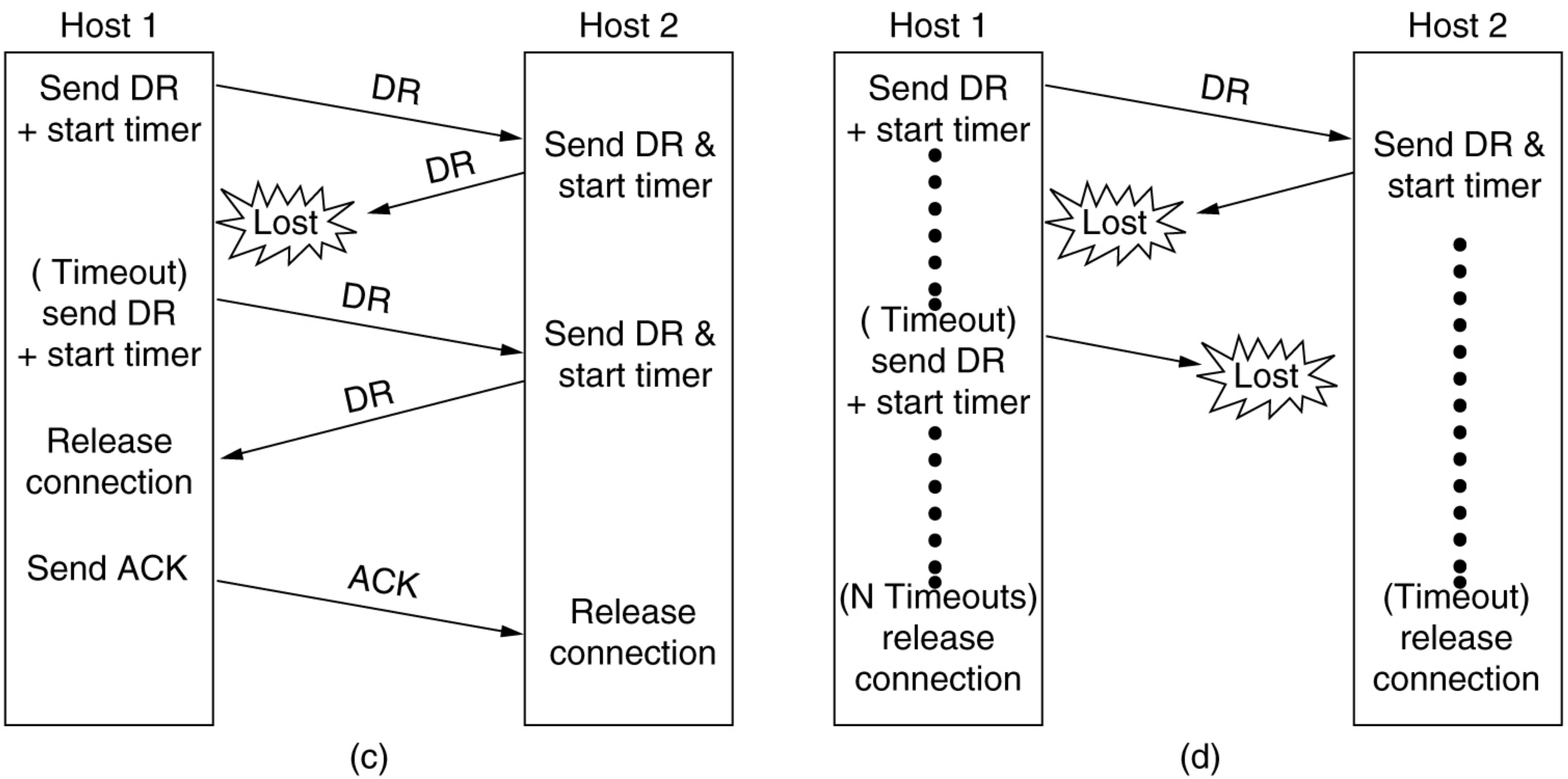


The two-army problem.

Symmetric Release (3)

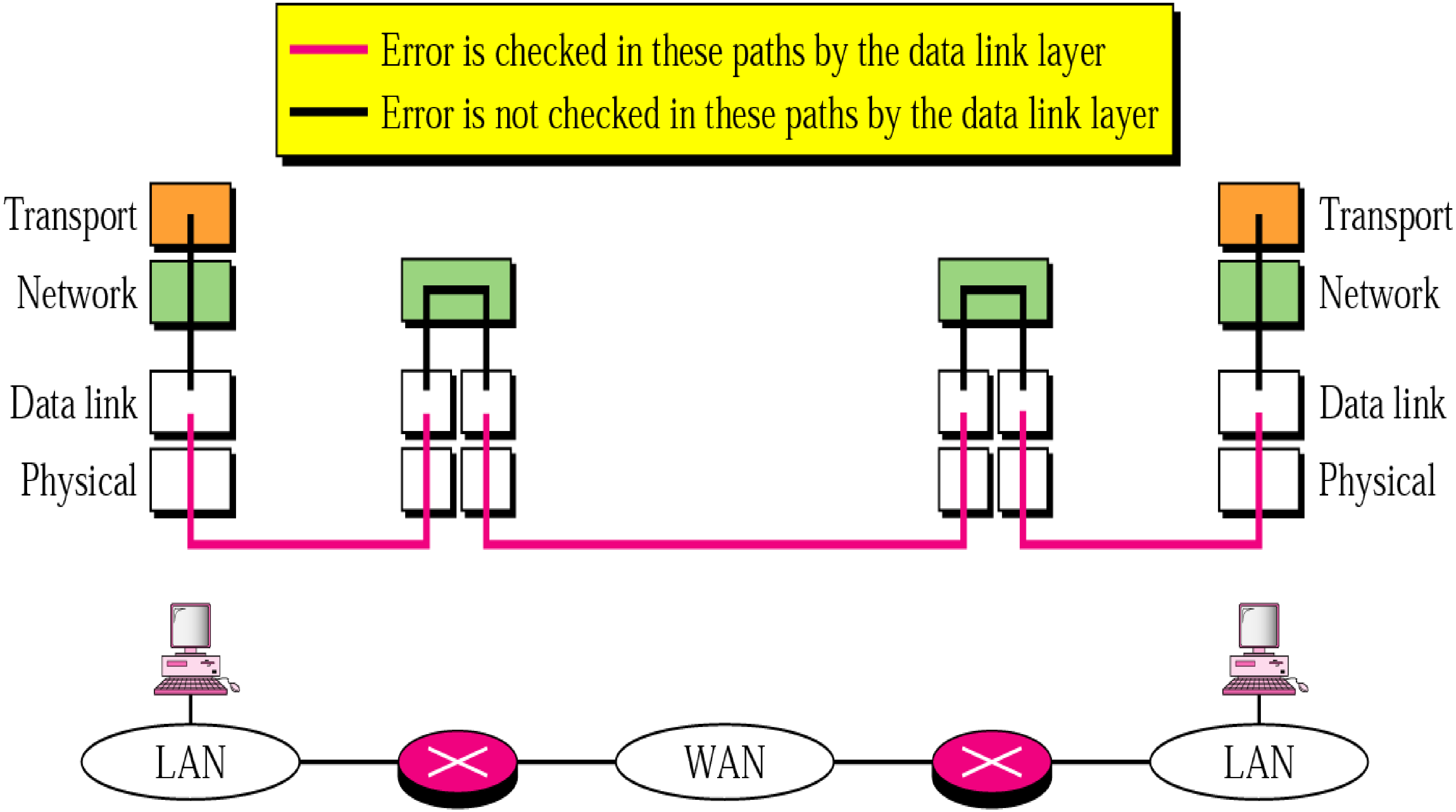
a) To see the relevance of the two-army problem to releasing connections, just substitute ”disconnect” for ”attack.”
b) if neither side is prepared to disconnect until it is convinced that the other side is prepared to disconnect too, the disconnection will never happen.
c) Examples:
d) Normal case: Host 1 sends disconnect request (DR). Host 2 responds with a DR. Host 1 acknowledges, and ACK arrives at host 2.
e) ACK is lost: What should host 2 do? It doesn’t know for sure that its DR came through.
f) Host 2’s DR is lost: What should host 1 do? Of course, send another DR, but this brings us back to the normal case. This still means that the ACK sent by host 1 may still get lost.

Connection Release (4)

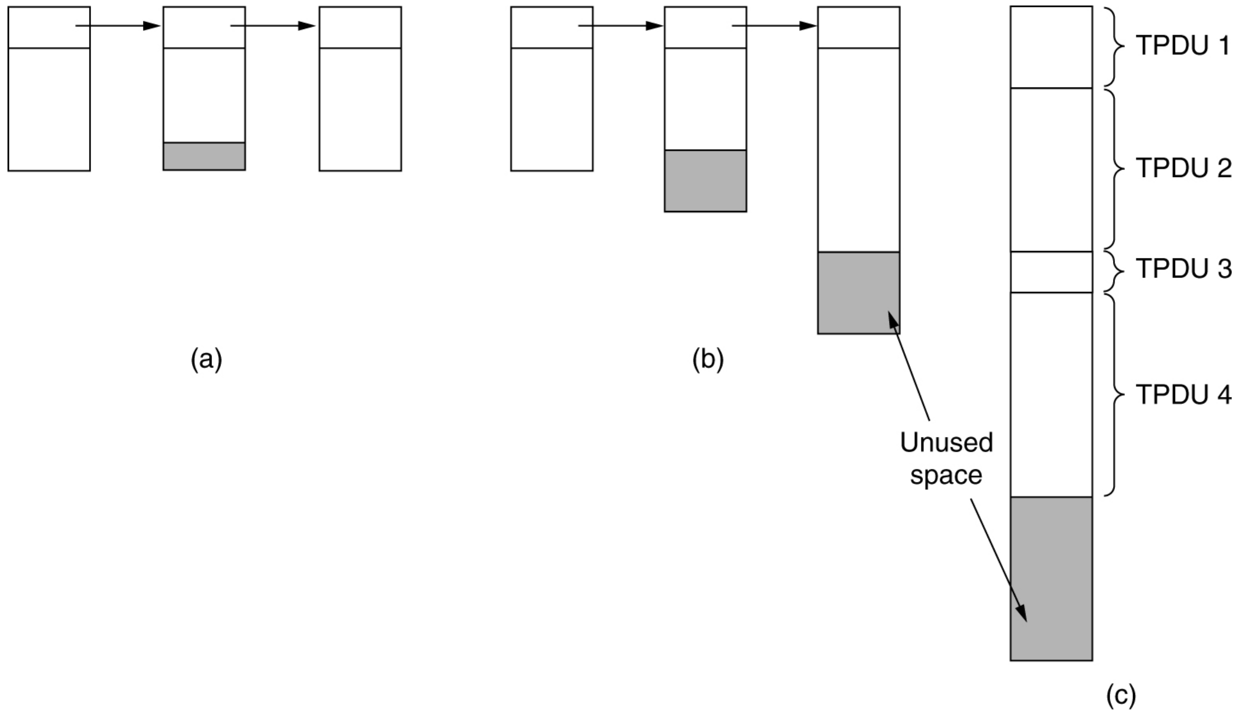


(c) Response lost. (d) Response lost and subsequent DRs lost.

Error Control



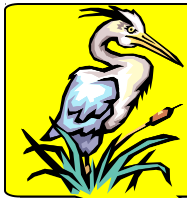
Flow Control and Buffering



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

The Internet Transport Protocols: UDP

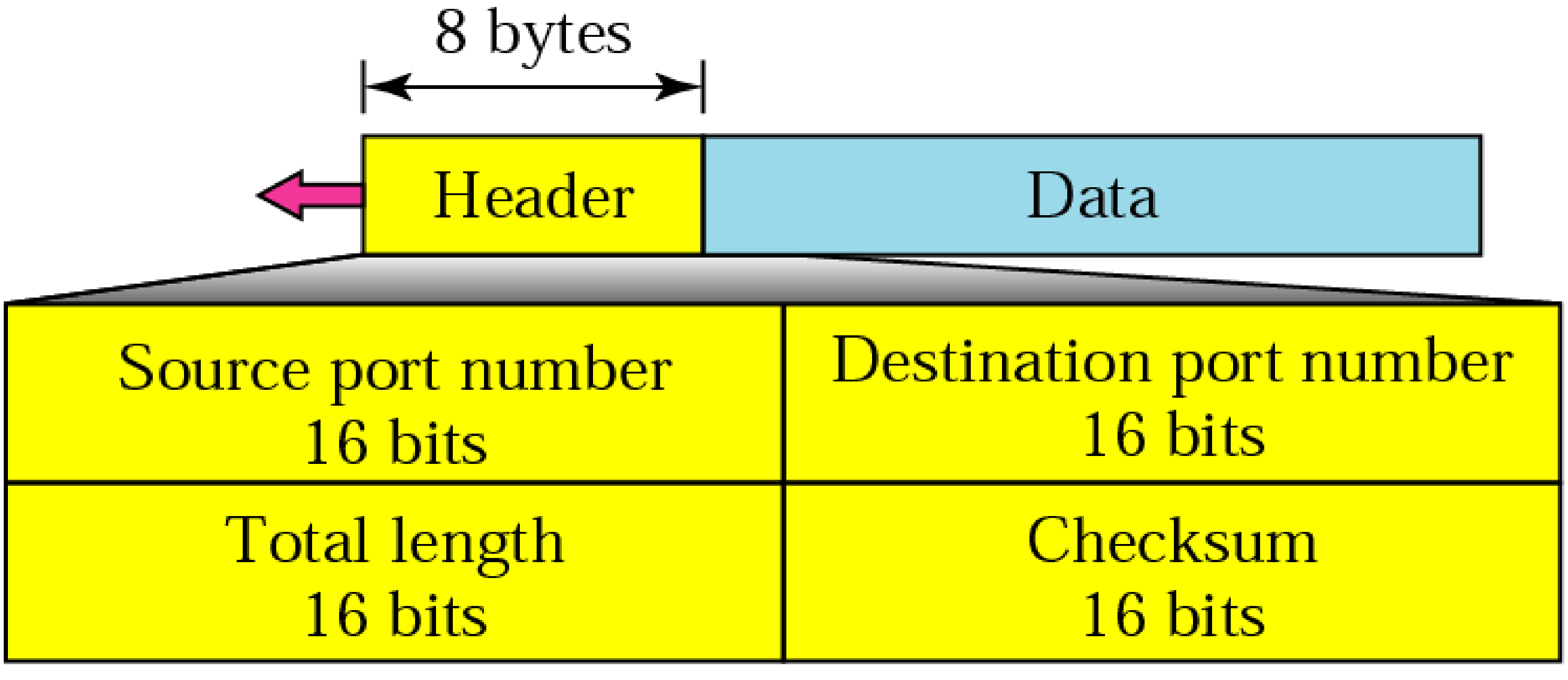
- Introduction to UDP
- Remote Procedure Call
- The Real-Time Transport Protocol



Note:

UDP is a connectionless, unreliable protocol that has no flow and error control. It uses port numbers to multiplex data from the application layer.

Introduction to UDP

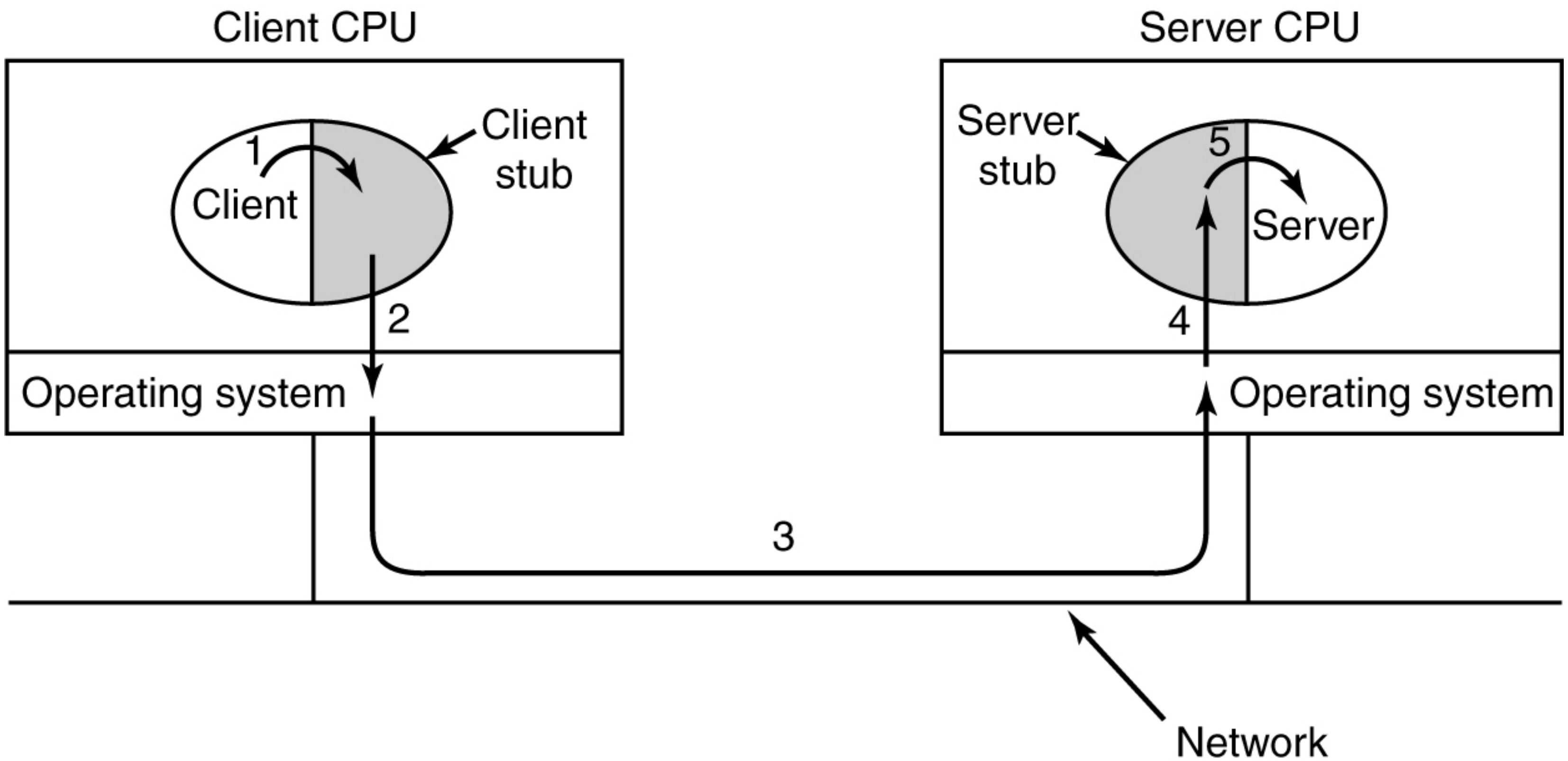


The UDP header.

Popular Application

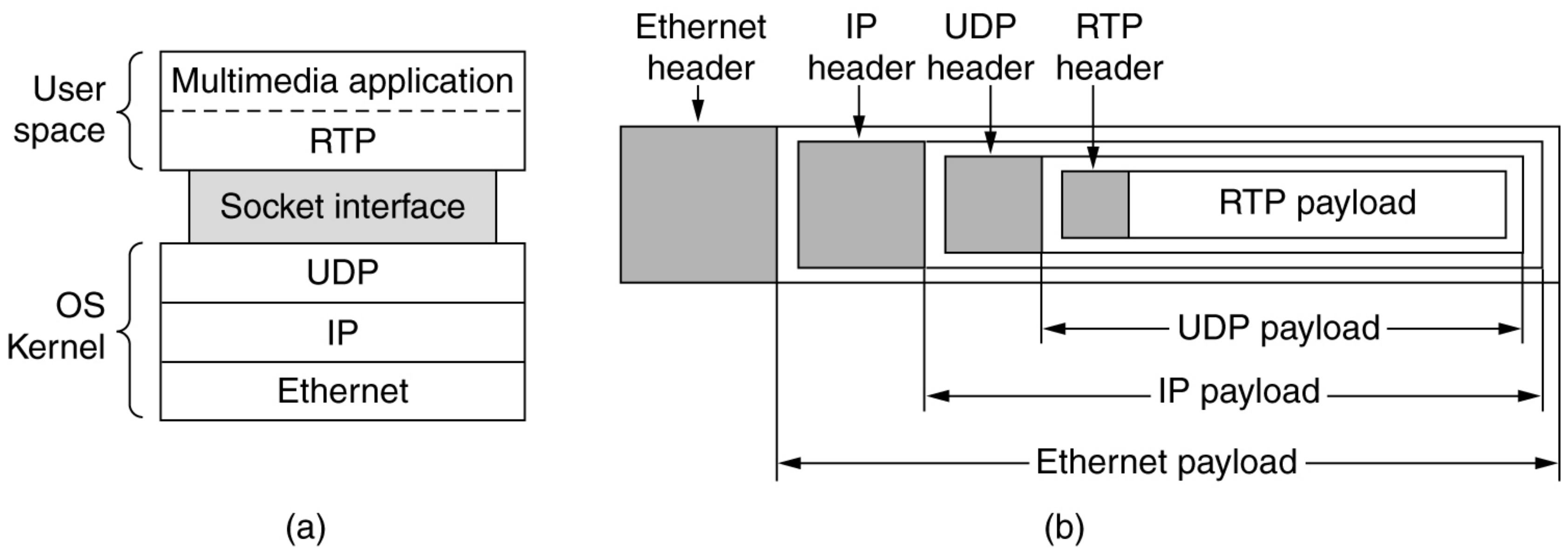
Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	Bootps	Server port to download bootstrap information
68	Bootpc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

Remote Procedure Call



Steps in making a remote procedure call. The stubs are shaded.

The Real-Time Transport Protocol



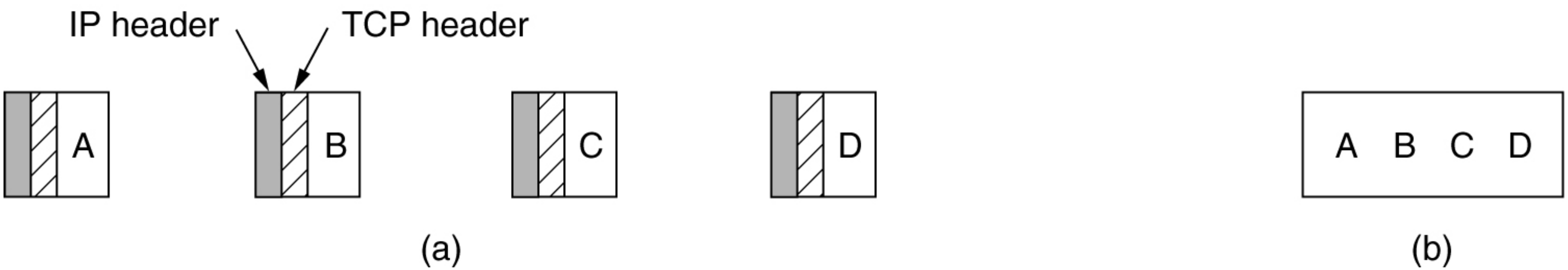
(a) The position of RTP in the protocol stack. (b) Packet nesting.

The Internet Transport Protocols: TCP

- Introduction to TCP
- The TCP Service Model
- The TCP Protocol
- The TCP Segment Header
- TCP Connection Establishment
- TCP Connection Release

Port	Protocol	Description
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (data connection)
21	FTP, Control	File Transfer Protocol (control connection)
23	TELNET	Terminal Network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name Server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

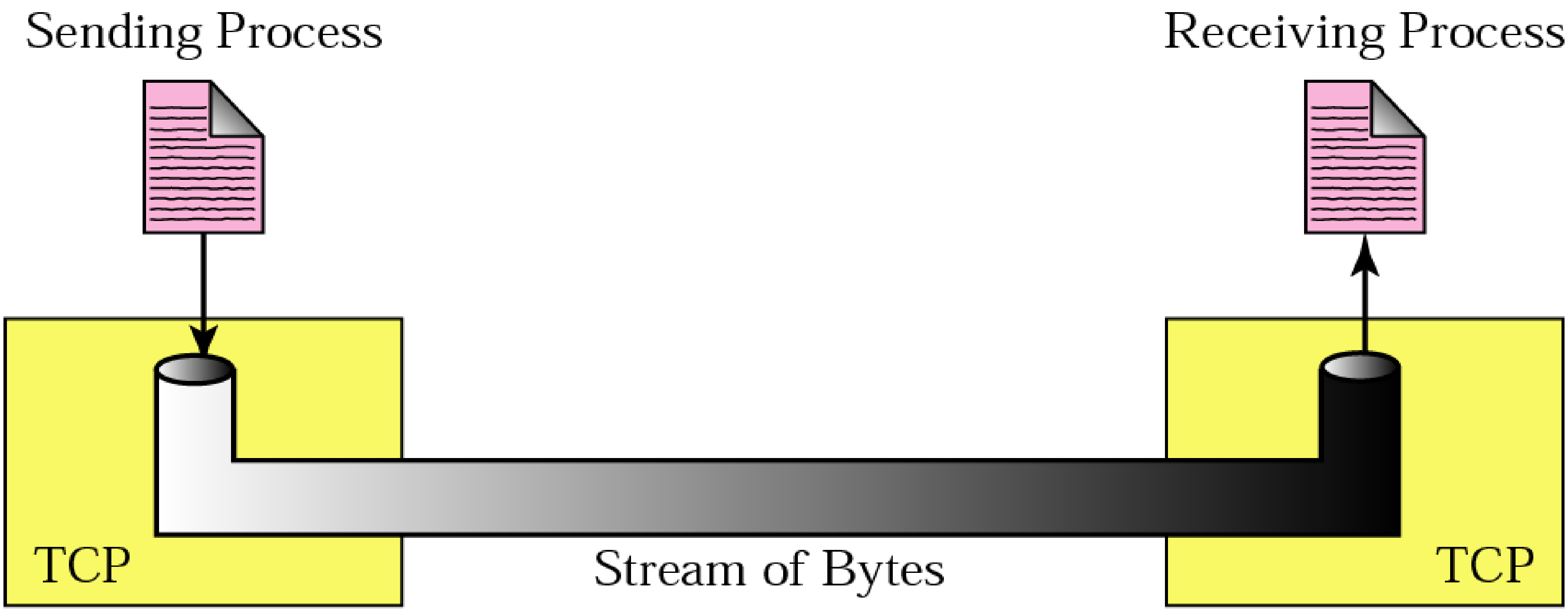
The TCP Service Model (2)



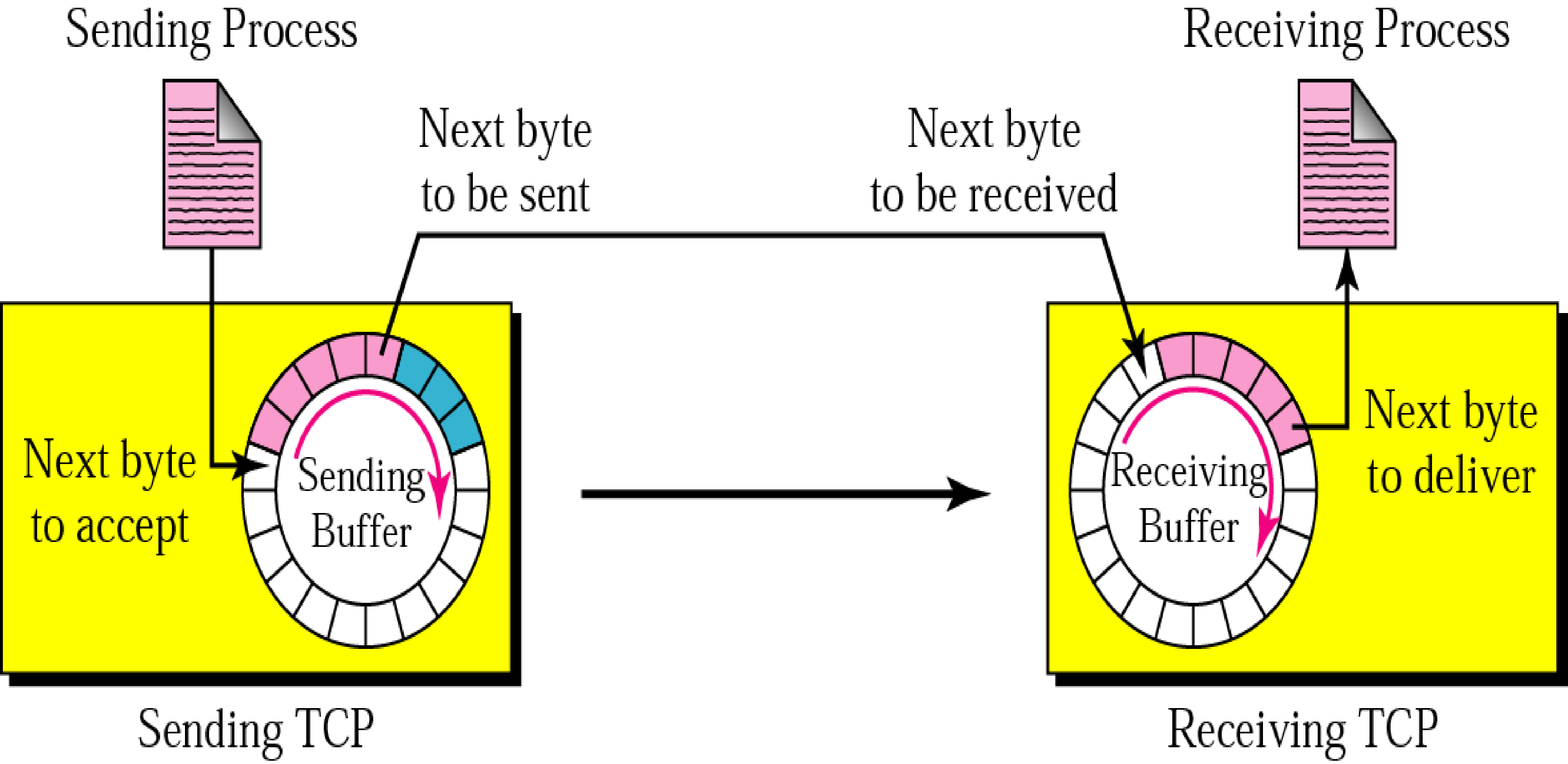
(a) Four 512-byte segments sent as separate IP datagrams.

(b) The 2048 bytes of data delivered to the application in a single READ CALL.

Stream Delivery



Sending & Receiving Window



Example 1

Imagine a TCP connection is transferring a file of 6000 bytes. The first byte is numbered 10010. What are the sequence numbers for each segment if data are sent in five segments with the first four segments carrying 1000 bytes and the last segment carrying 2000 bytes?

Solution

The following shows the sequence number for each segment:

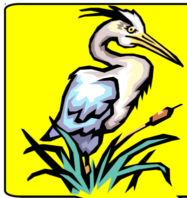
Segment 1 ==> sequence number: 10,010 (range: 10,010 to 11,009)

Segment 2 ==> sequence number: 11,010 (range: 11,010 to 12,009)

Segment 3 ==> sequence number: 12,010 (range: 12,010 to 13,009)

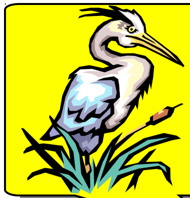
Segment 4 ==> sequence number: 13,010 (range: 13,010 to 14,009)

Segment 5 ==> sequence number: 14,010 (range: 14,010 to 16,009)



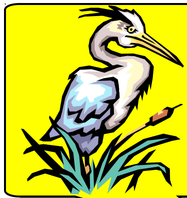
Note:

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



Note:

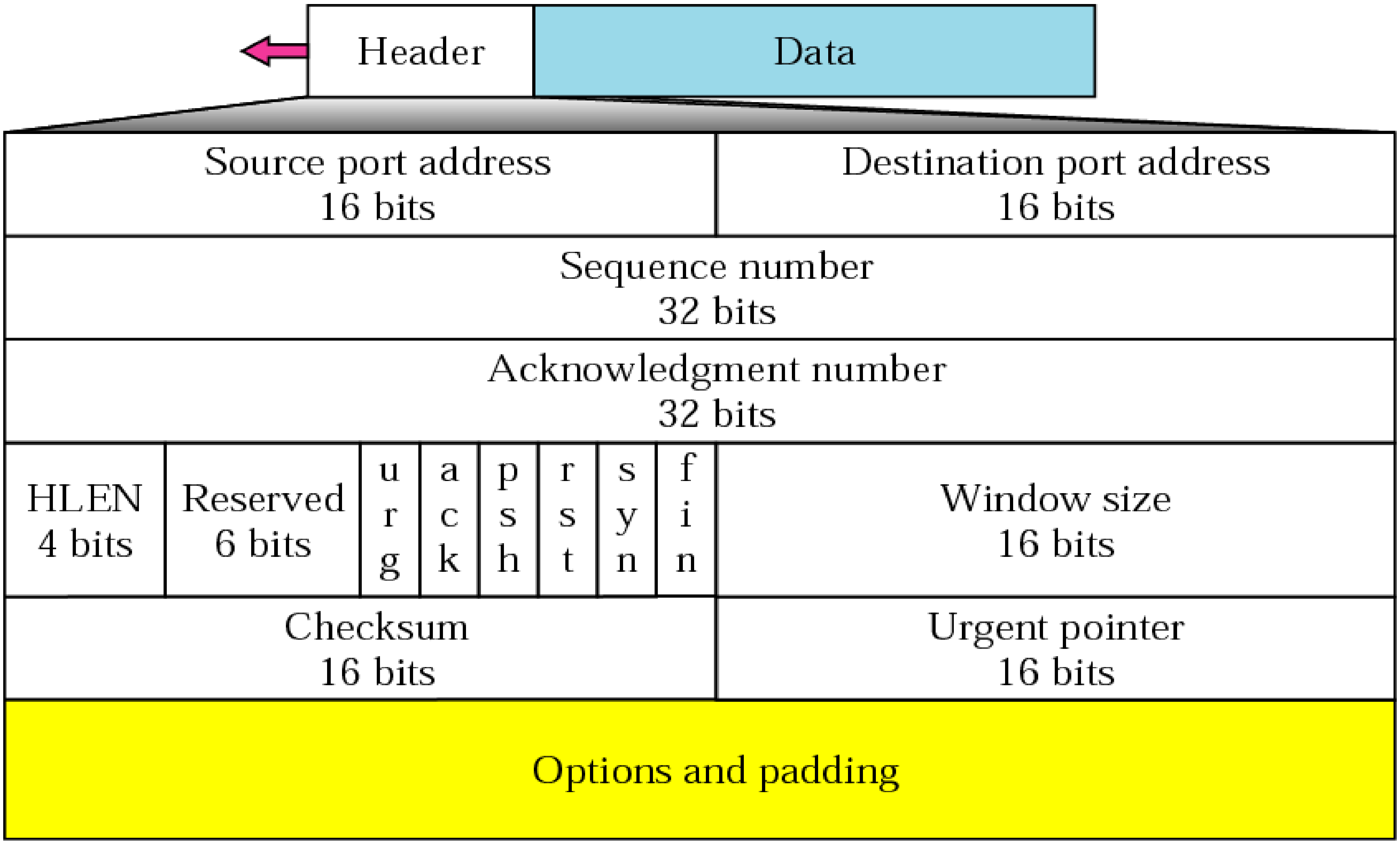
The value of the sequence number field in a segment defines the number of the first data byte contained in that segment.



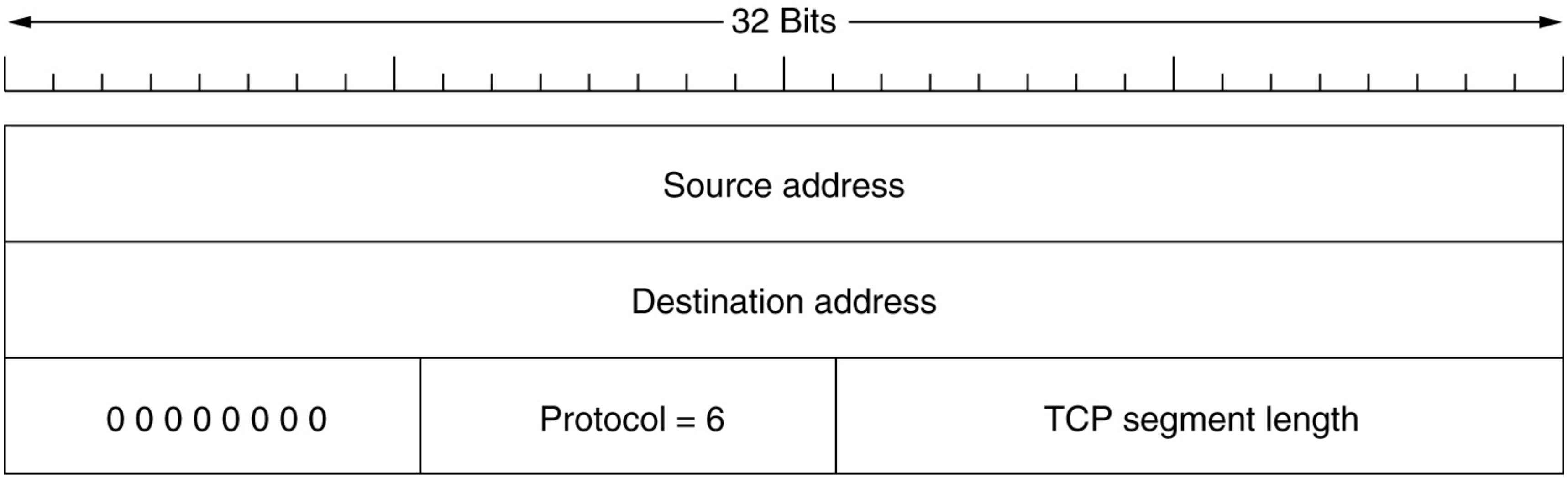
Note:

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.

The TCP Segment Header



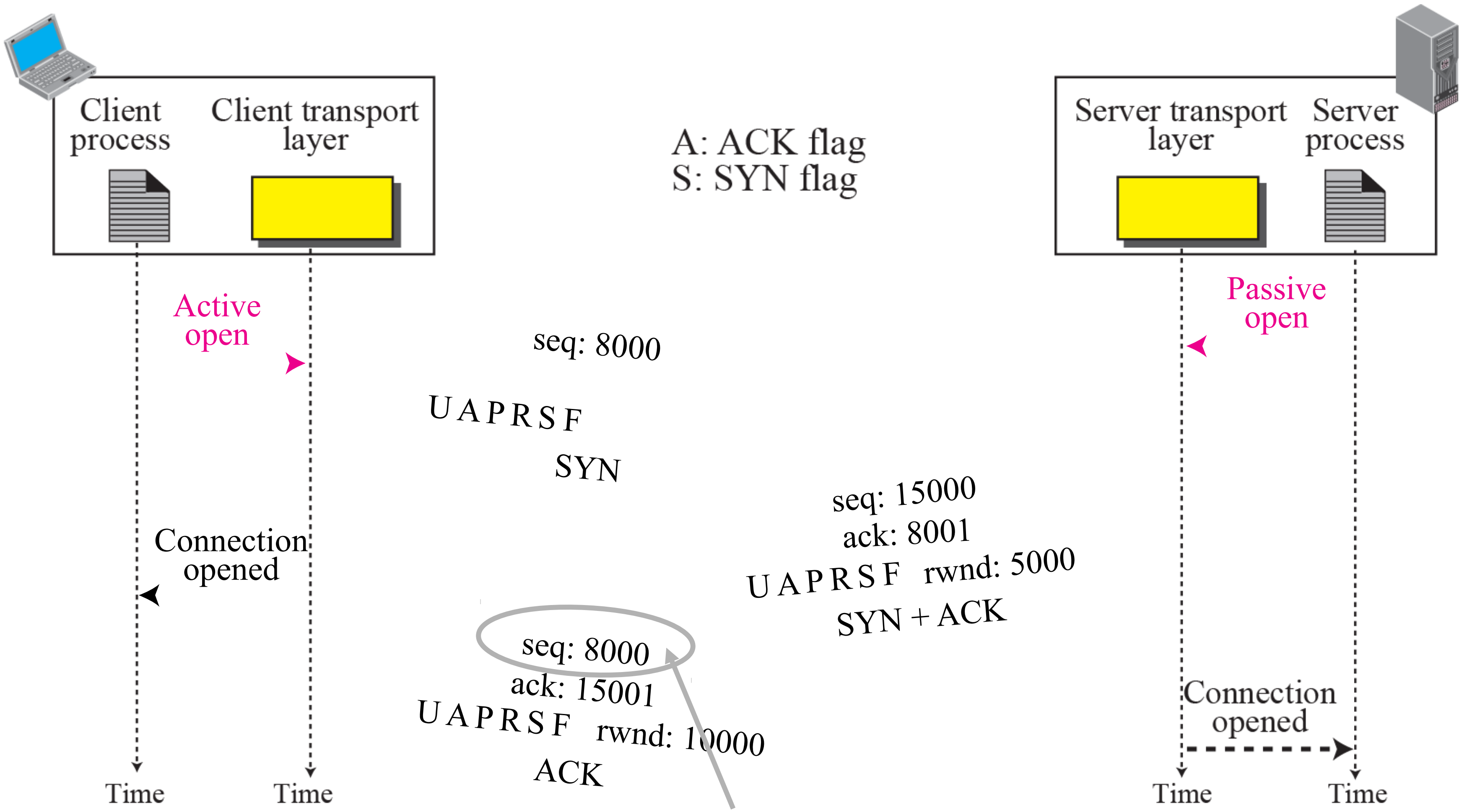
The TCP Segment Header (2)



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

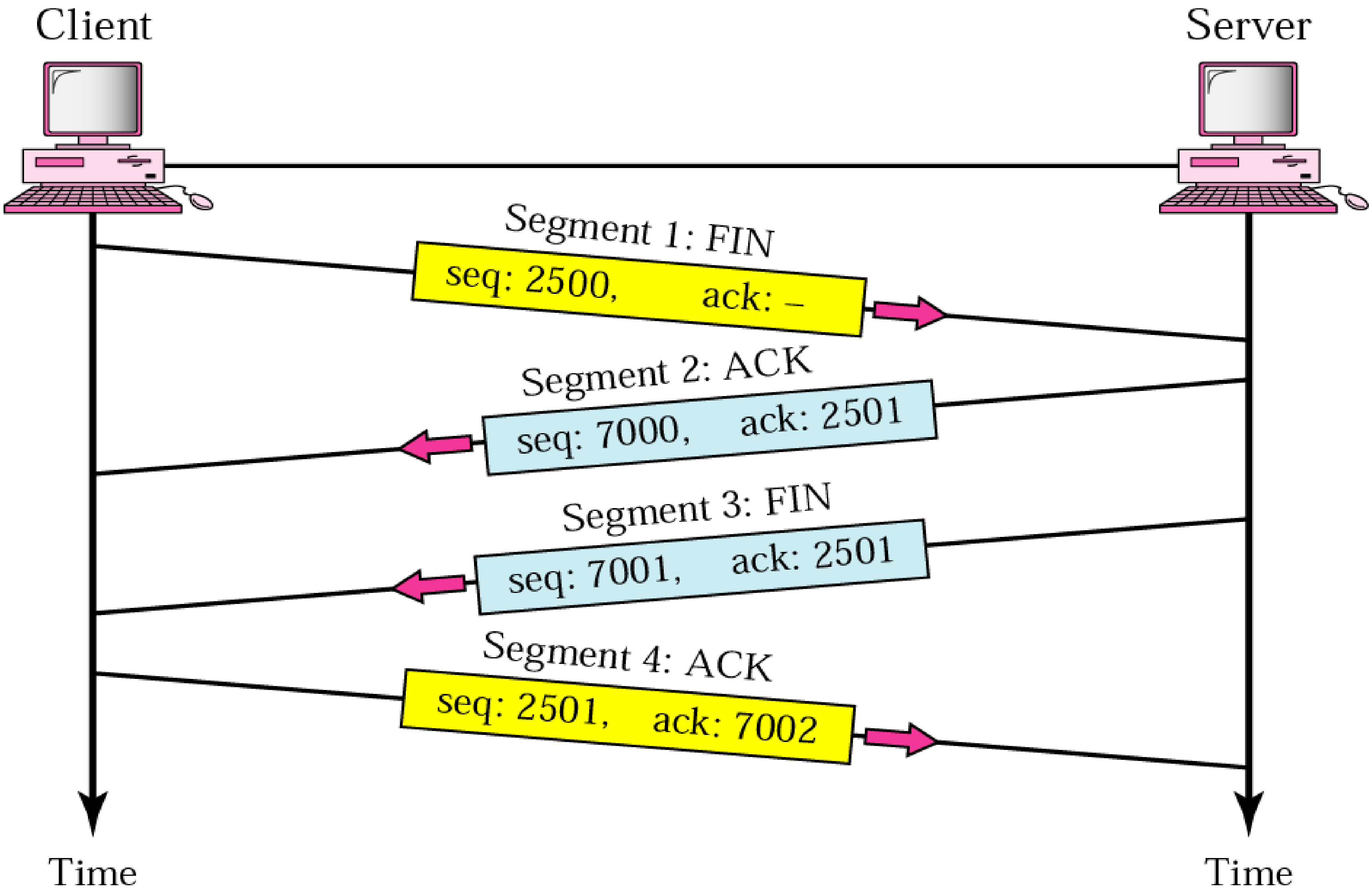
Connection Establishment



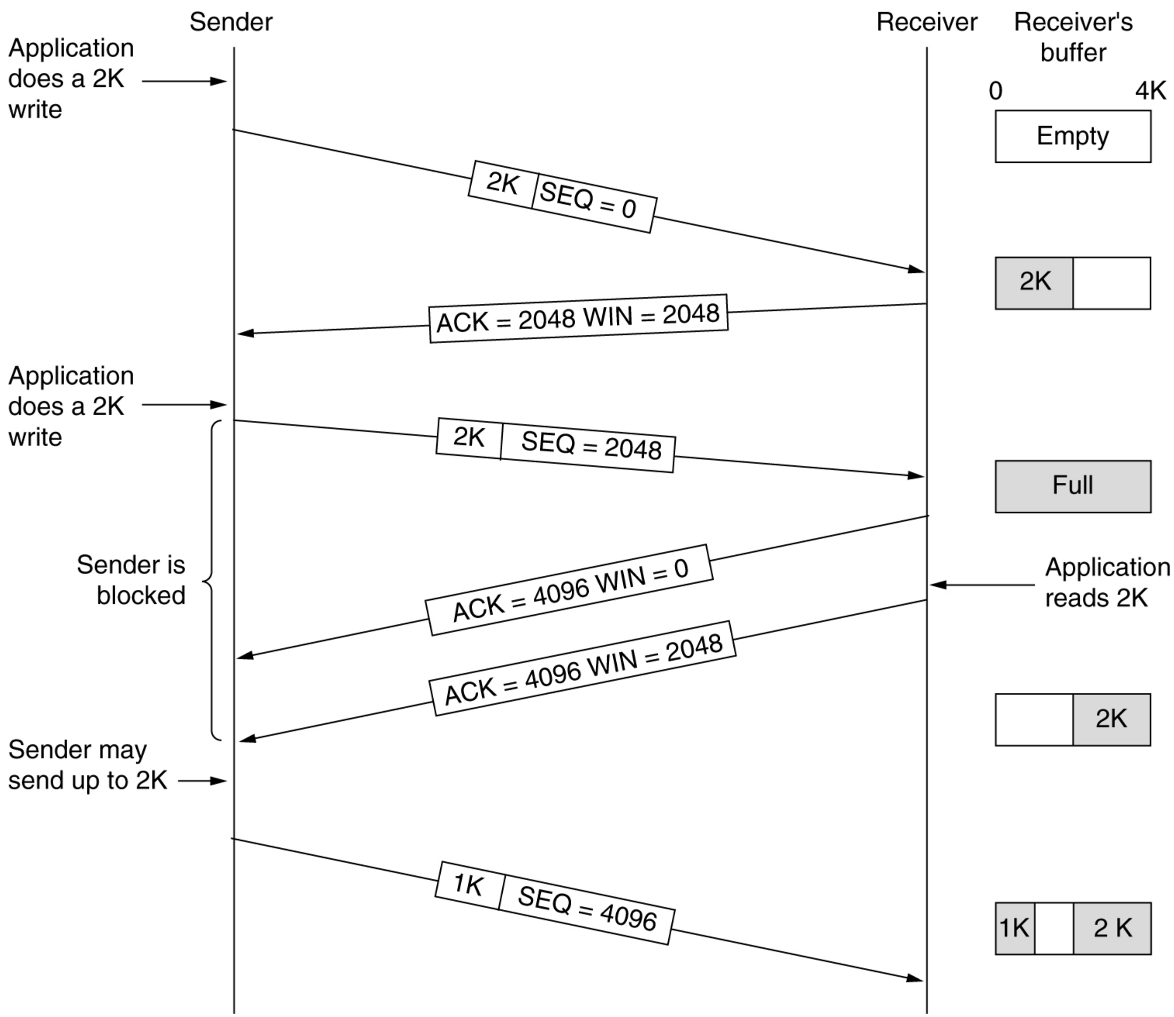
Means "no data" !

seq: 8001 if piggybacking

Connection Release

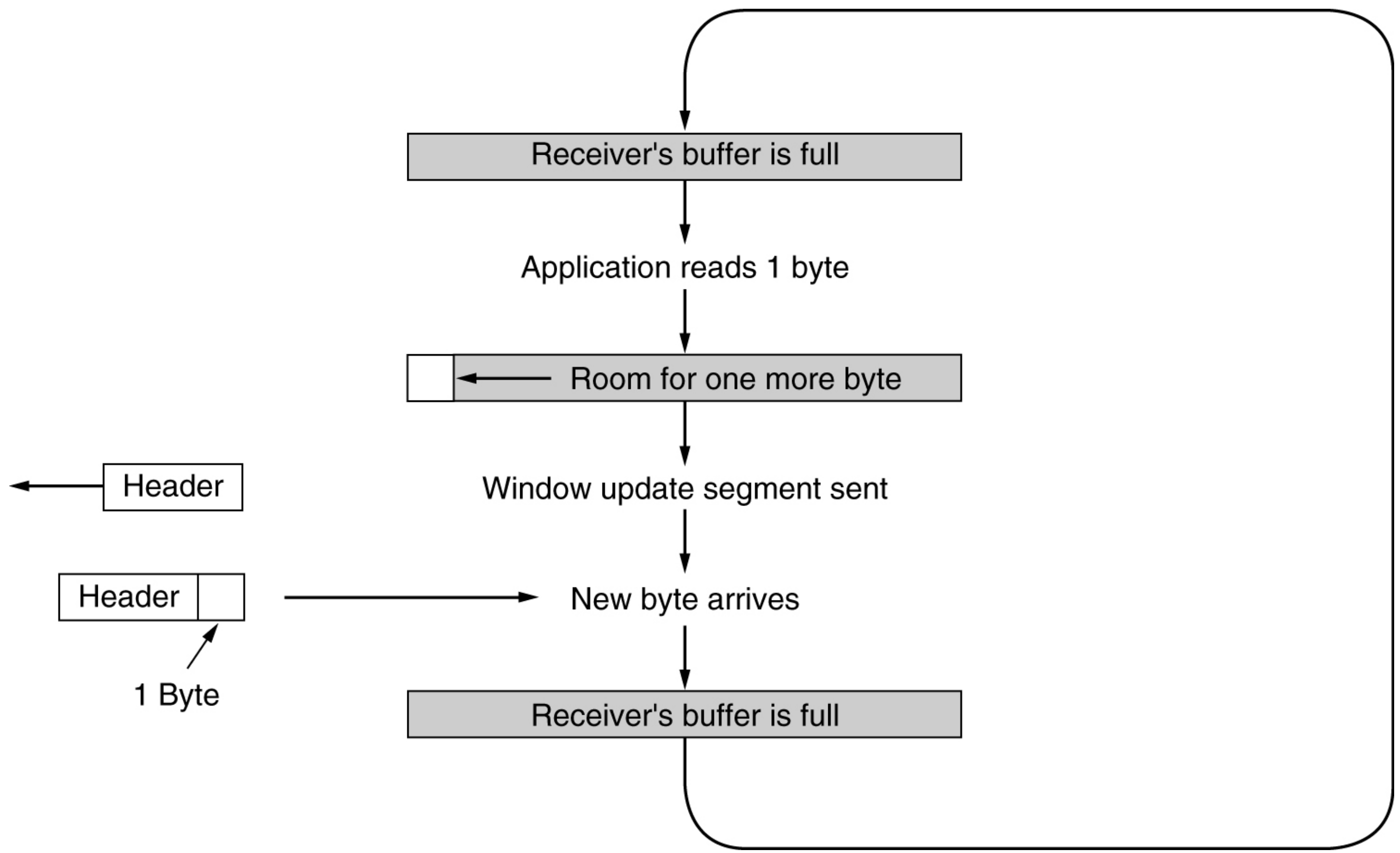


TCP Transmission Policy



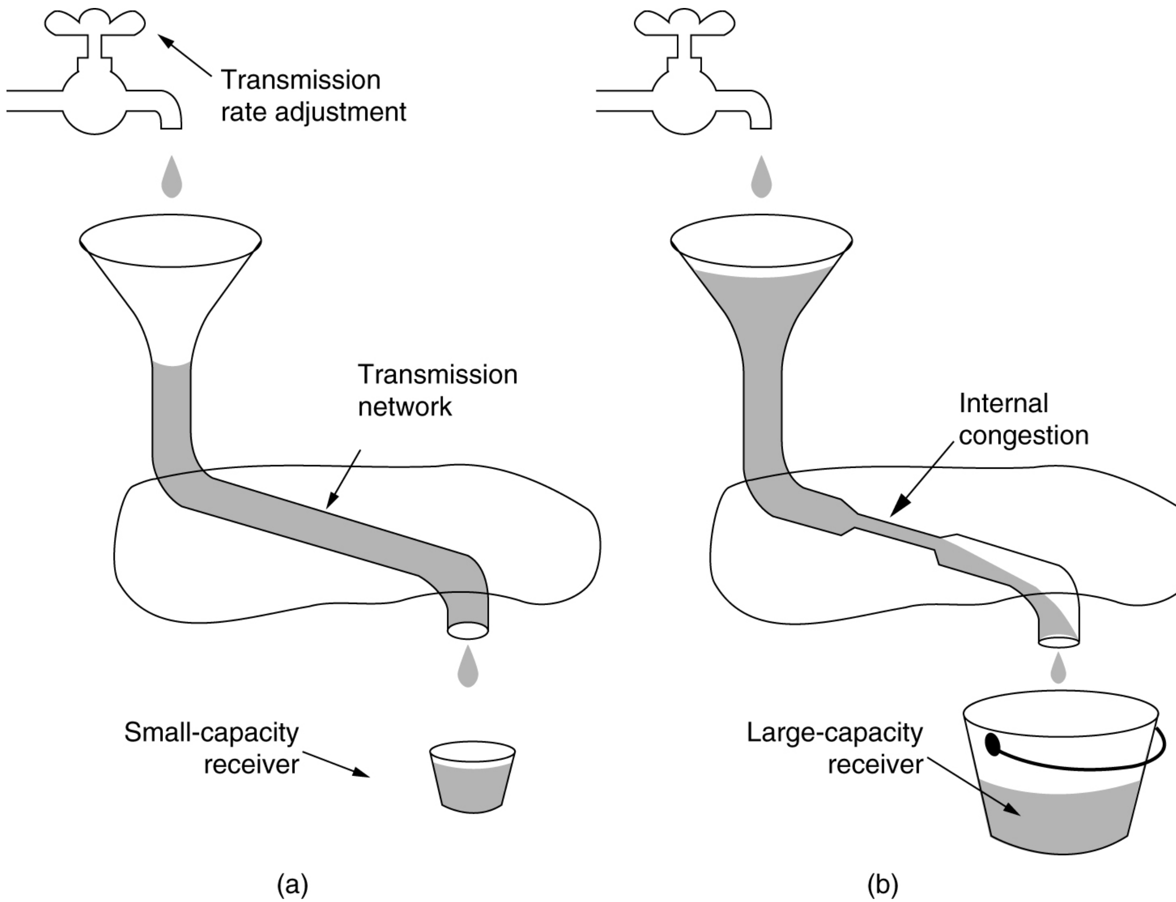
Window management in TCP.

TCP Transmission Policy (2)



Silly window syndrome.

TCP Congestion Control



(a) A fast network feeding a low capacity receiver.

(b) A slow network feeding a high-capacity receiver.

Congestion Control

a) TCP has a mechanism for congestion control. The mechanism is implemented at the sender

b) The window size at the sender is set as follows:

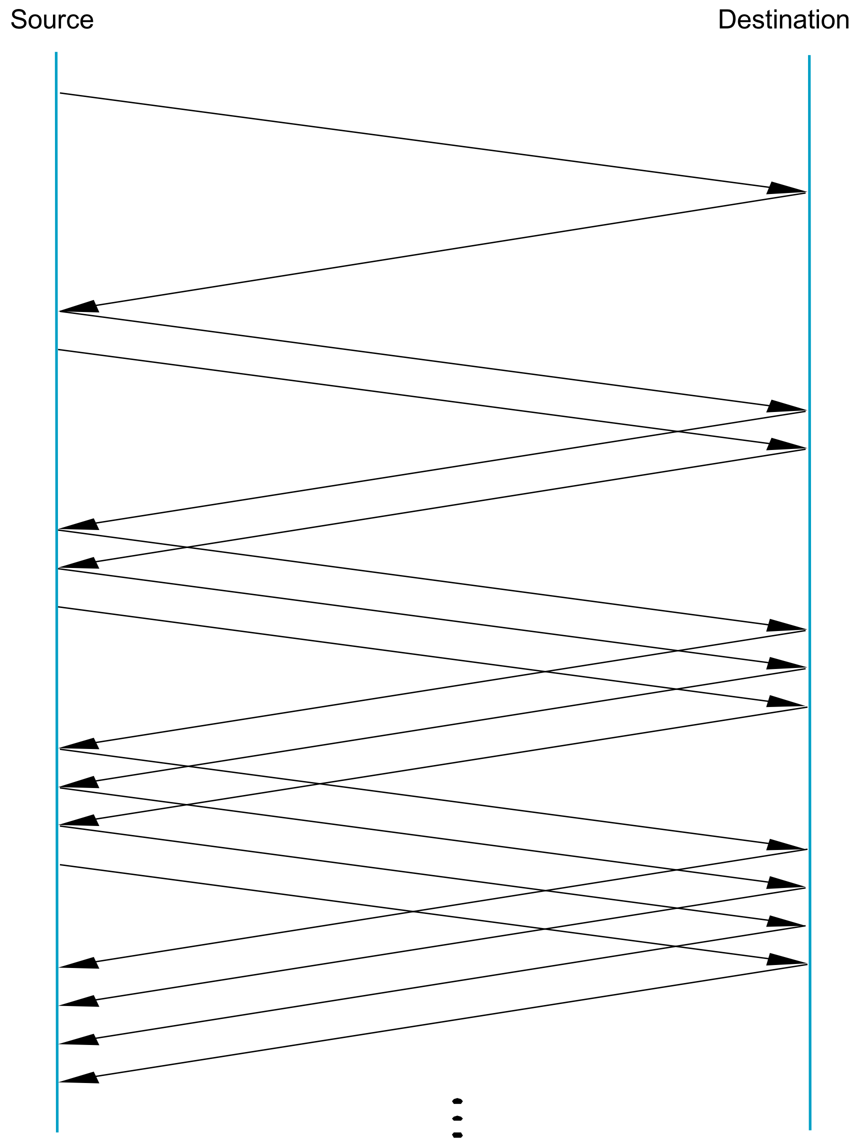
Send Window = MIN (flow control window, congestion window)

where

a) flow control window is advertised by the receiver

b) congestion window is adjusted based on feedback from the network

AIMD



Add one packet
each RTT

Disadvantage

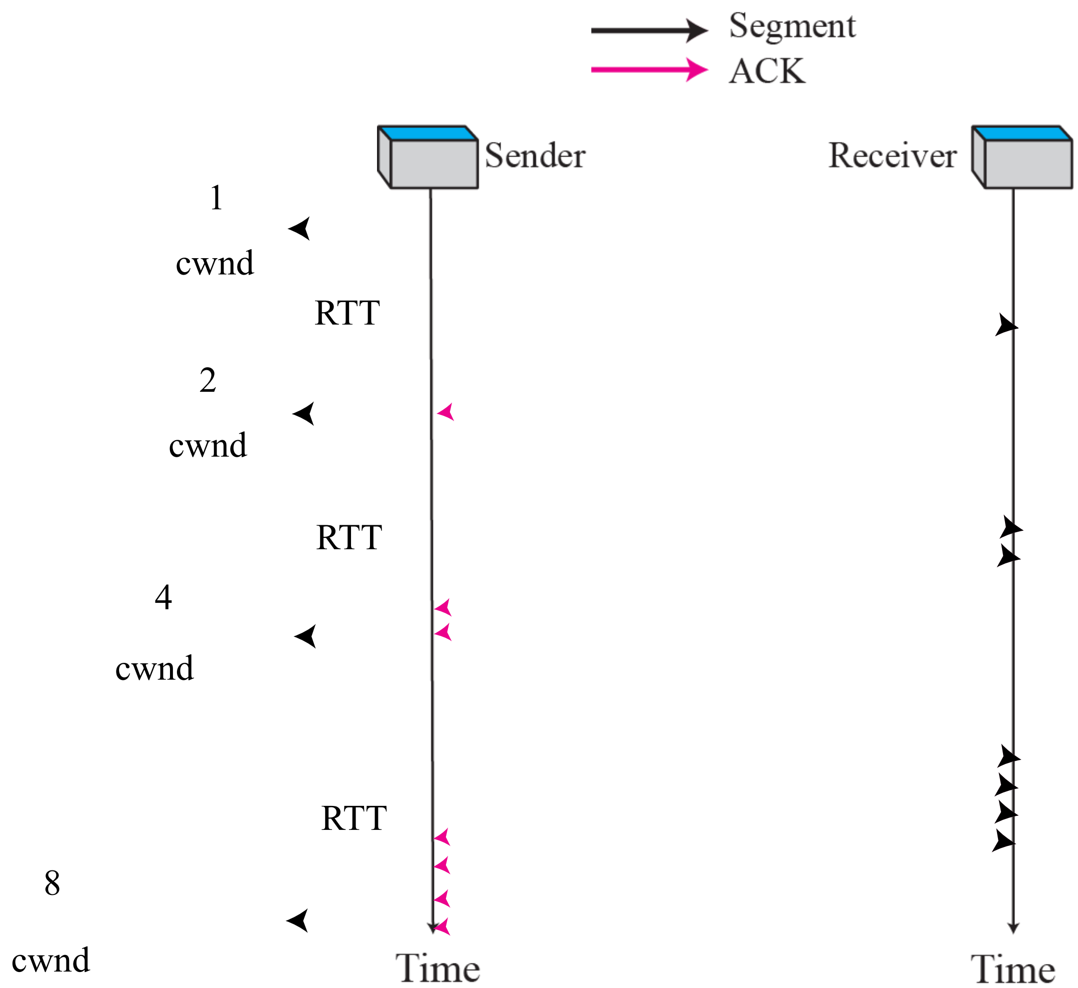
a) Too slow.
b) Reacts aggressively.
c) Wastage of bandwidth at initial stage.
d) Congestion is detected when time out occurs.

Congestion Control

a) The sender has two additional parameters:
   - **Congestion Window** (**cwnd**) Initial value is 1 MSS (=maximum segment size) counted as bytes
   - **Slow-start threshold Value** (**ssthresh)** Initial value is the advertised window size)

a) Congestion control works in two modes:
   - **Slow start** (cwnd < ssthresh)
   - **Congestion avoidance** (cwnd >= ssthresh)

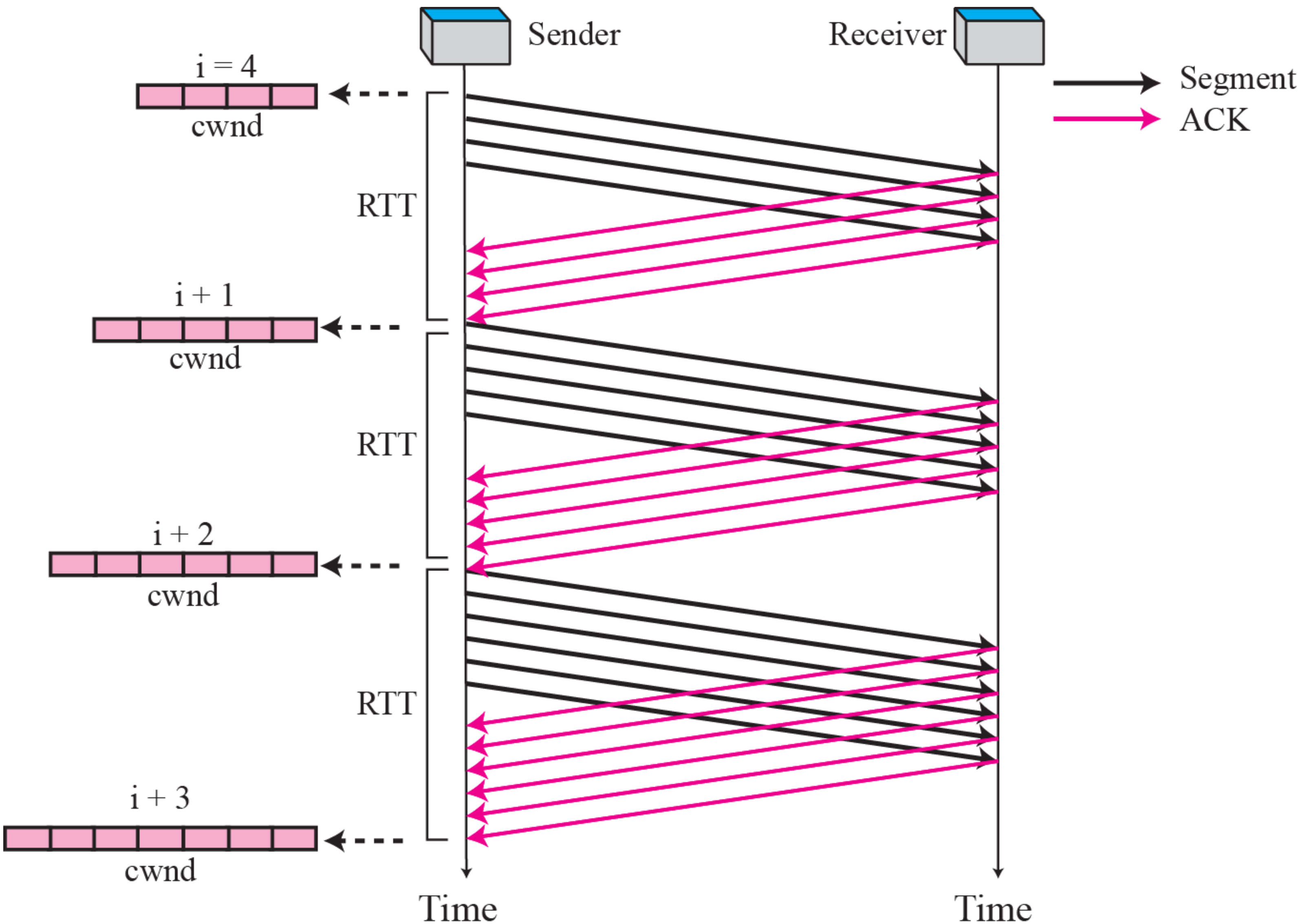
Slow start, exponential increase



Note

In the slow start algorithm, the size of the congestion window increases exponentially until it reaches a threshold.

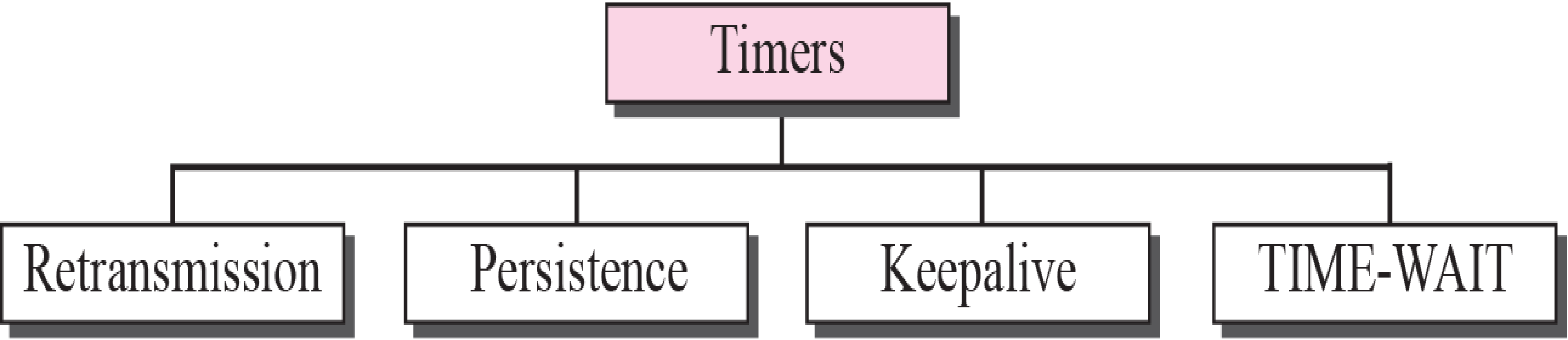
Congestion avoidance, additive increase



Note

In the congestion avoidance algorithm the size of the congestion window increases additively until congestion is detected.

TCP Timers



Retransmission Timer

a) When a segment is sent, a retransmission timer is started.

b) If the segment is acknowledged before the timer expires, the timer is stopped.

c) If, on the other hand, the timer goes off before the acknowledgement comes in, the segment is retransmitted (and the timer os started again).

Persistence Timer

a) It is designed to prevent the following deadlock.

b) The receiver sends an acknowledgement with a window size of 0, telling the sender to wait. Later, the receiver updates the window, but the packet with the update is lost. Now the sender and the receiver are each waiting for the other to do something.

c) When the persistence timer goes off, the sender transmits a probe to the receiver. The response to the probe gives the window size. If it is still 0, the persistence timer is set again and the cycle repeats. If it is nonzero, data can now be sent.

Keep-alive Timer

a) When a connection has been idle for a long time, the keep-alive timer may go off to cause one side to check whether the other side is still there.

b) If it fails to respond, the connection is terminated.

Timed Wait Timer

a) It runs for twice the maximum packet lifetime to make sure that when a connection is closed, all packets created by it have died off.

Thanks

It's beginning of end