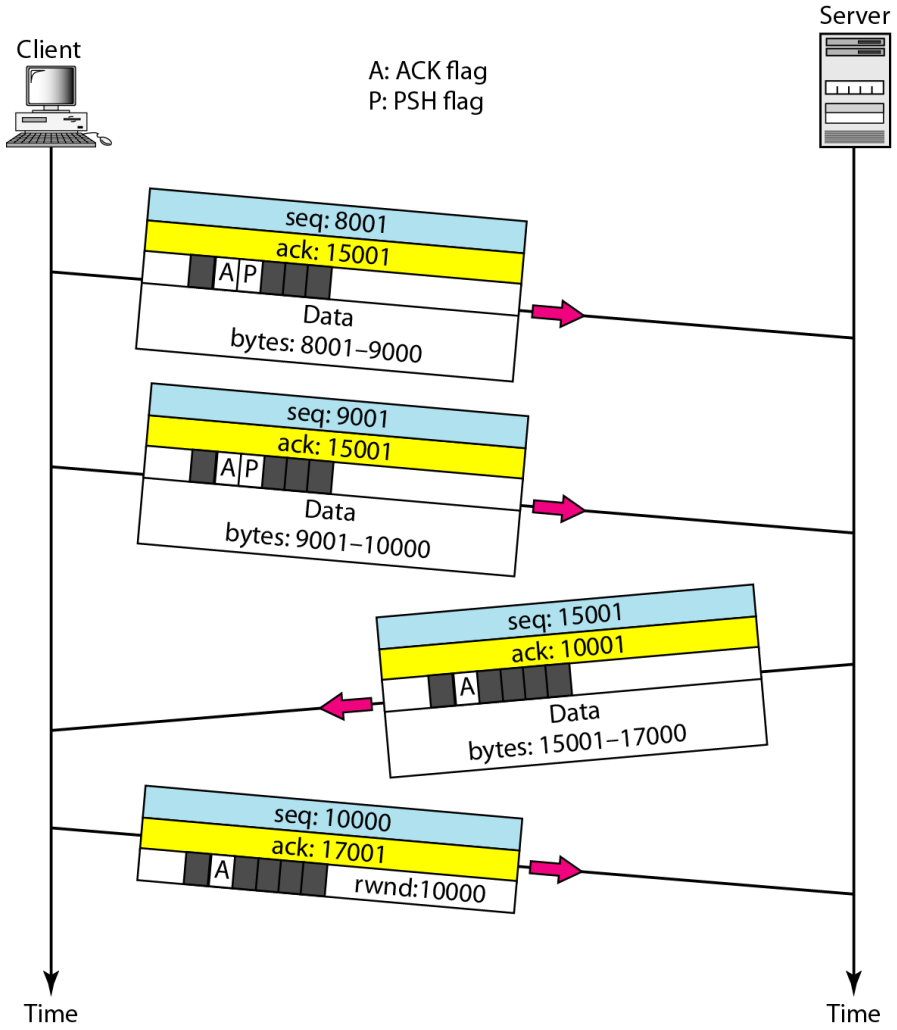
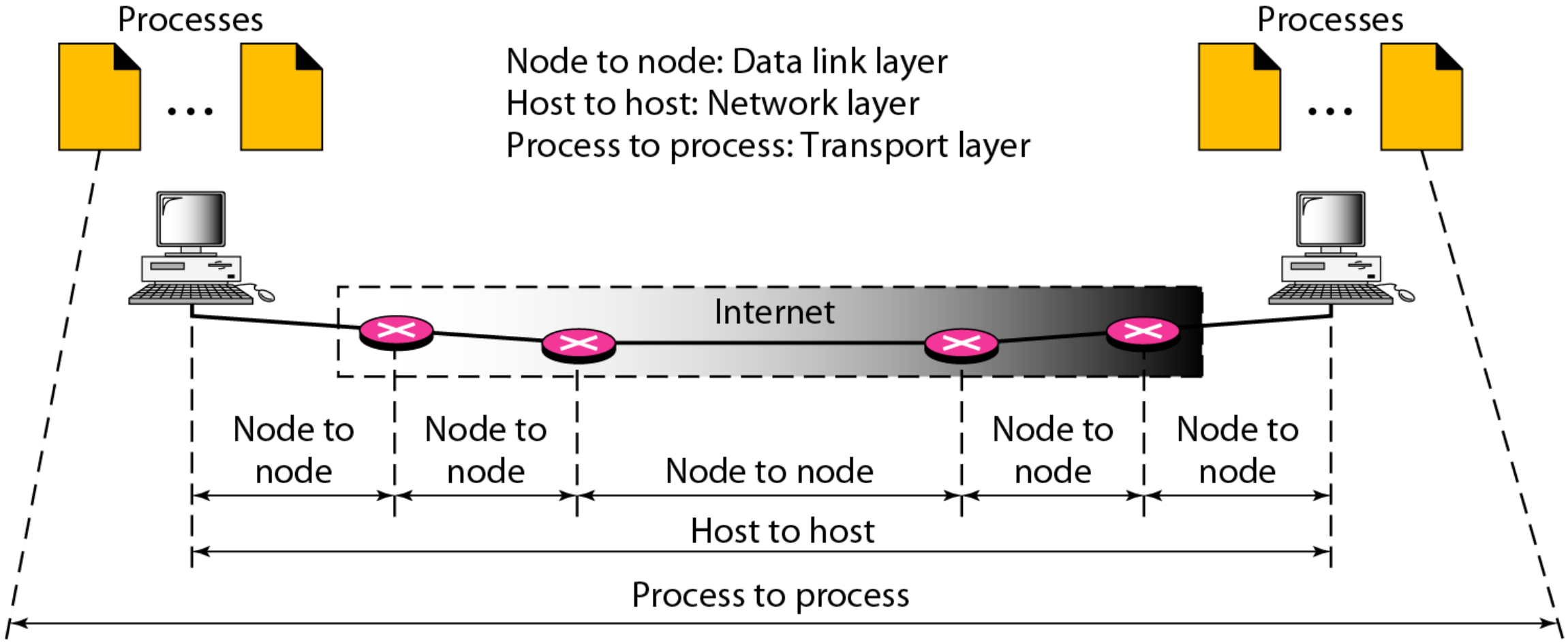


Transport Layer

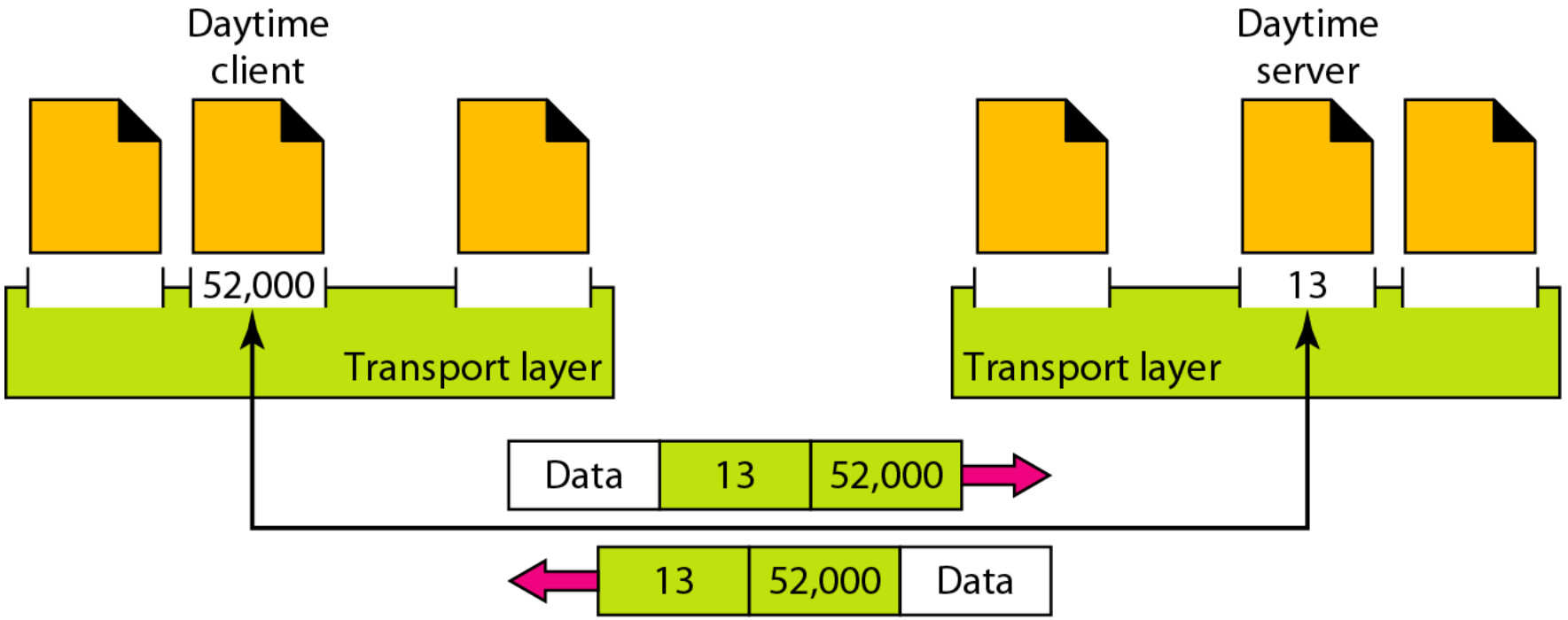
Outline

- Need of Transport Layer
- Functionalities (responsibilities) of Transport layer
  - True end to end layer
  - Provides Port address
  - Error & Flow control (in between Source & Destination)
  - Data transmission between process-to-process
  - Segmentation
- UDP
- TCP

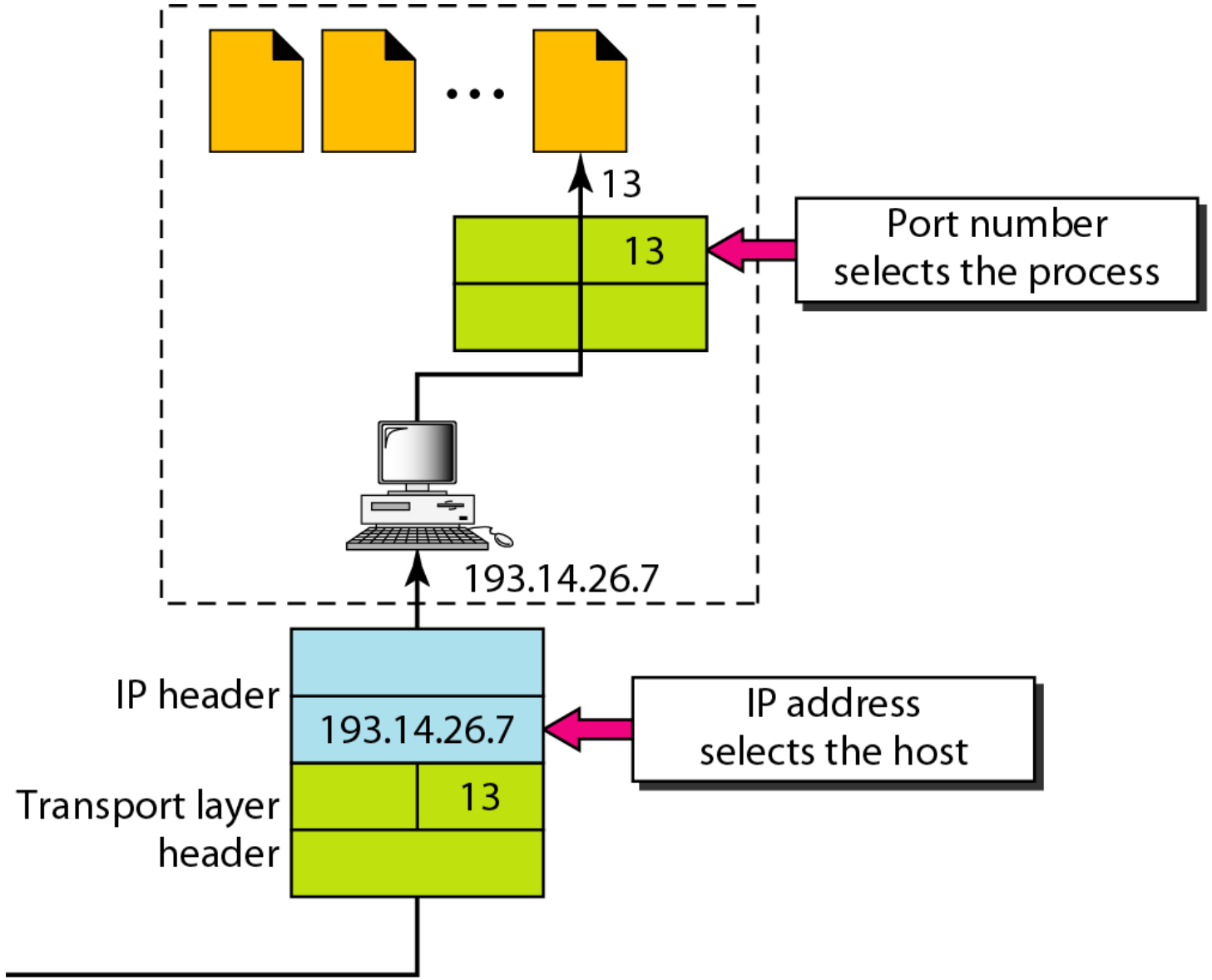
PROCESS-TO-PROCESS DELIVERY



Port numbers

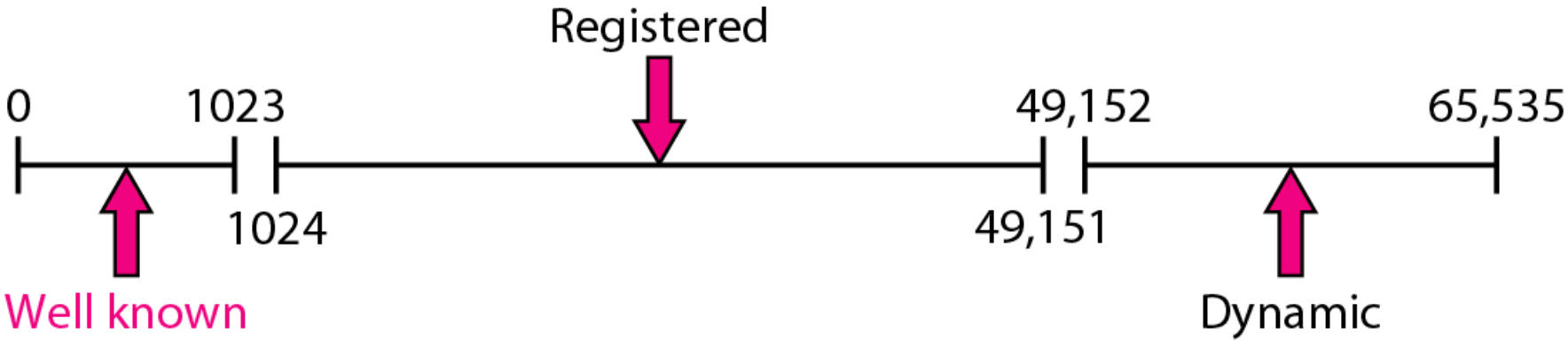


IP addresses versus port numbers

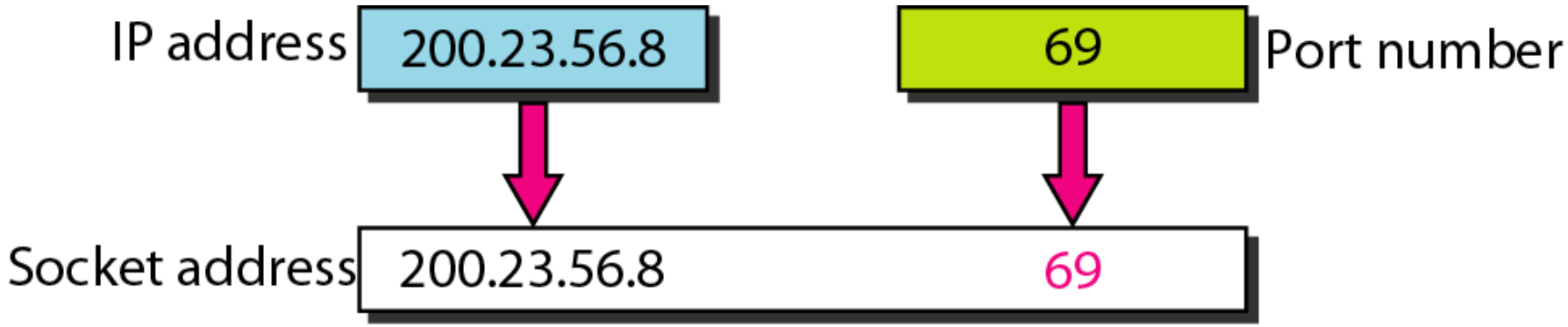


IANA ranges

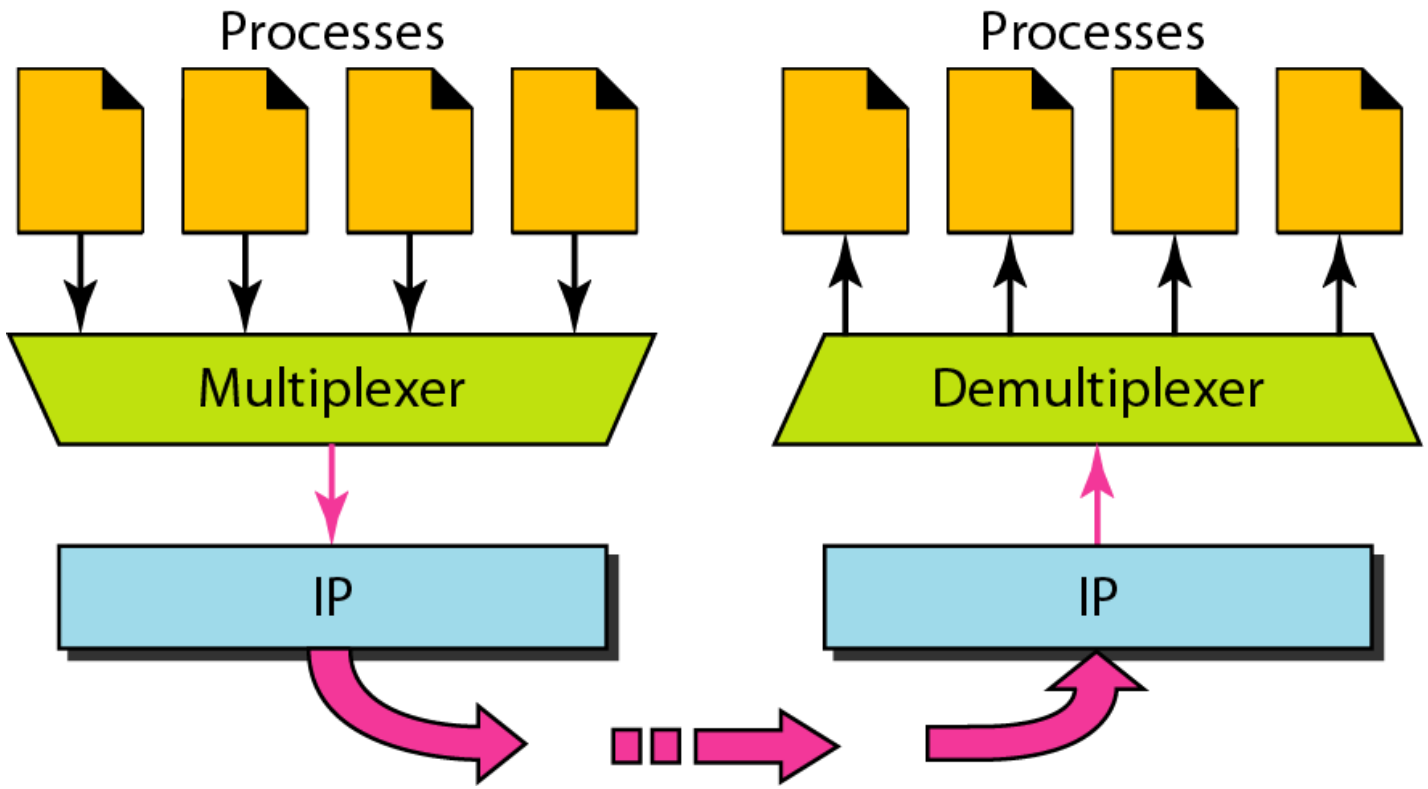
Port number = 16 bit



Socket address



Multiplexing and demultiplexing



Many-to-one relationship between processes and transport layer protocol at sender.

One-to-many relationship between processes and transport layer protocol at receiver.

Connection Oriented vs. Connection Less

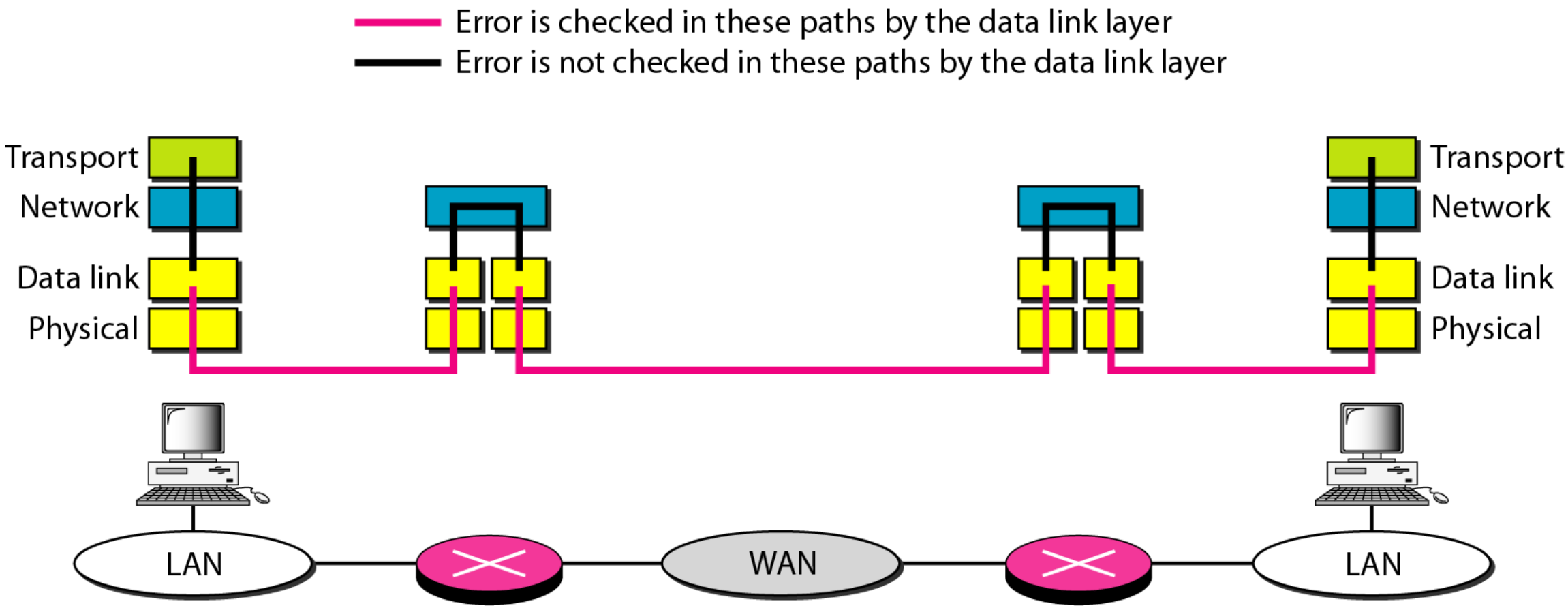
TCP: Will establish a connection, Data are numbered, has acknowledgement.

Byte oriented protocol

UDP: will not establish a connection, no acks, packets may arrive out of order, delayed or lost.

Message oriented protocol

Error control



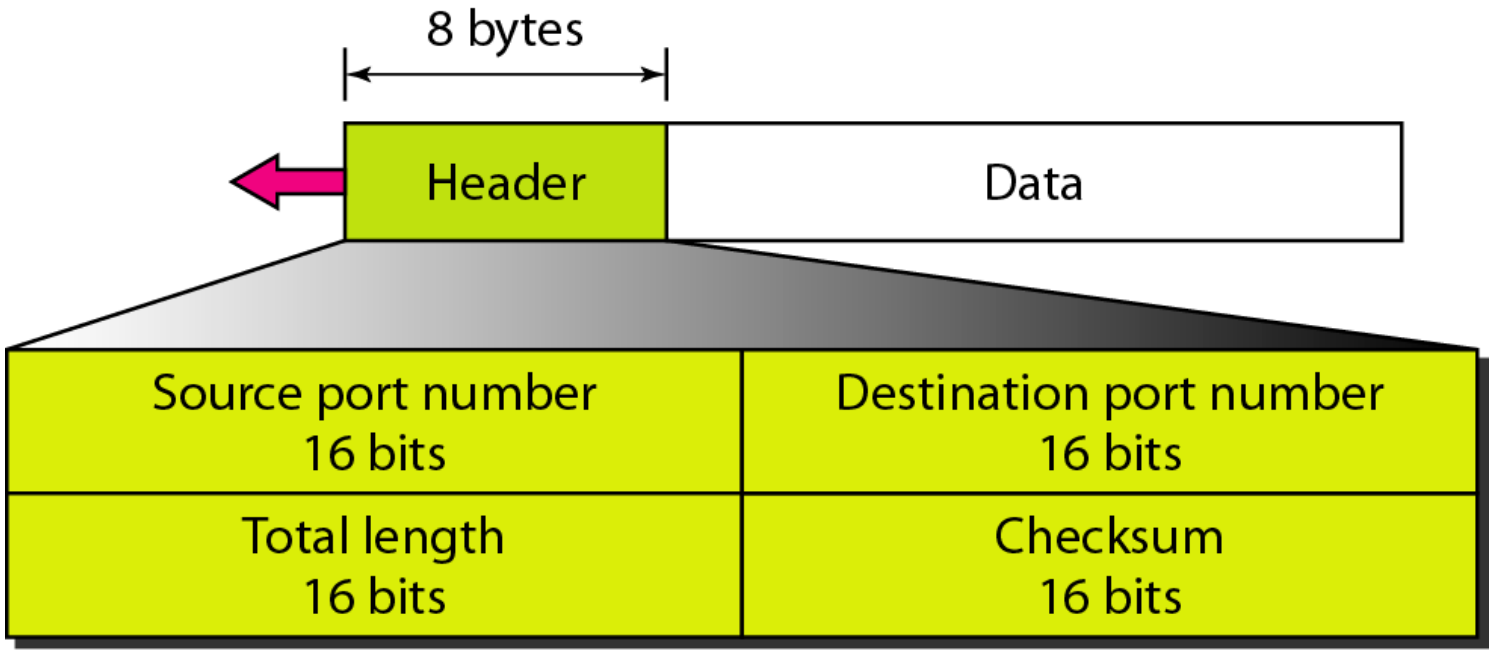
USER DATAGRAM PROTOCOL (UDP)

- *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.*
- *No flow or error control is there.*

Well-known ports used with UDP

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)

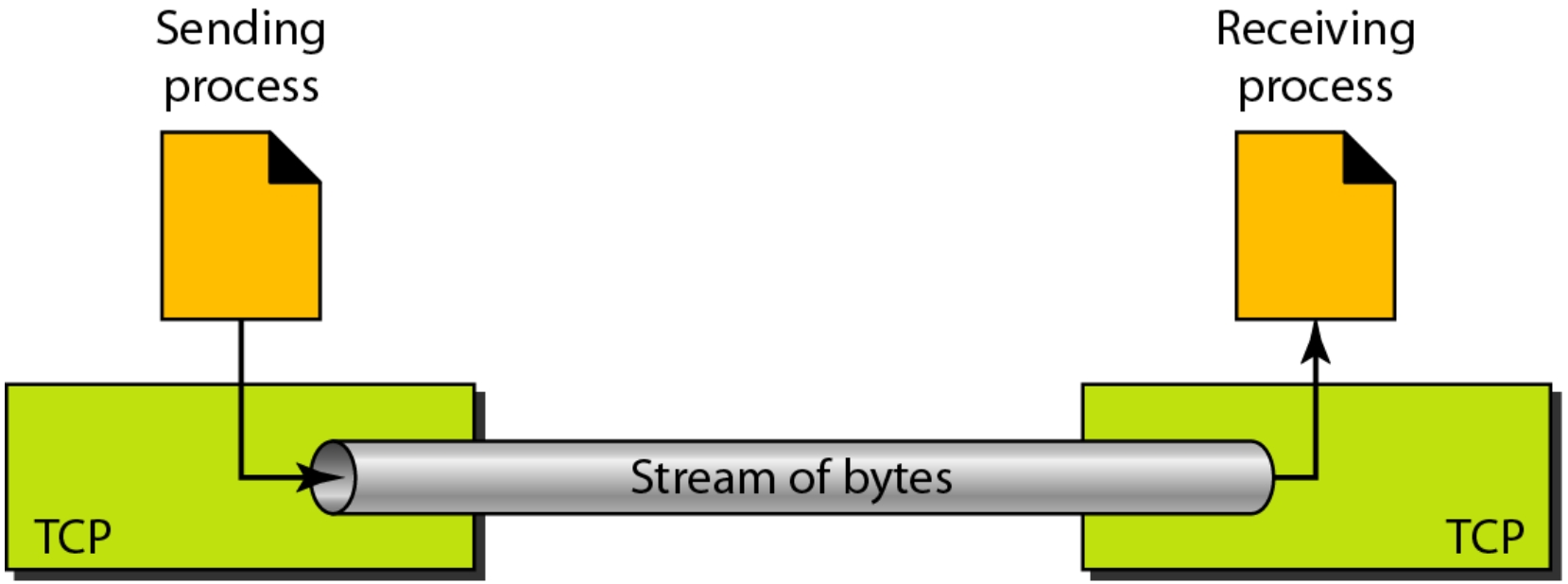
User datagram format



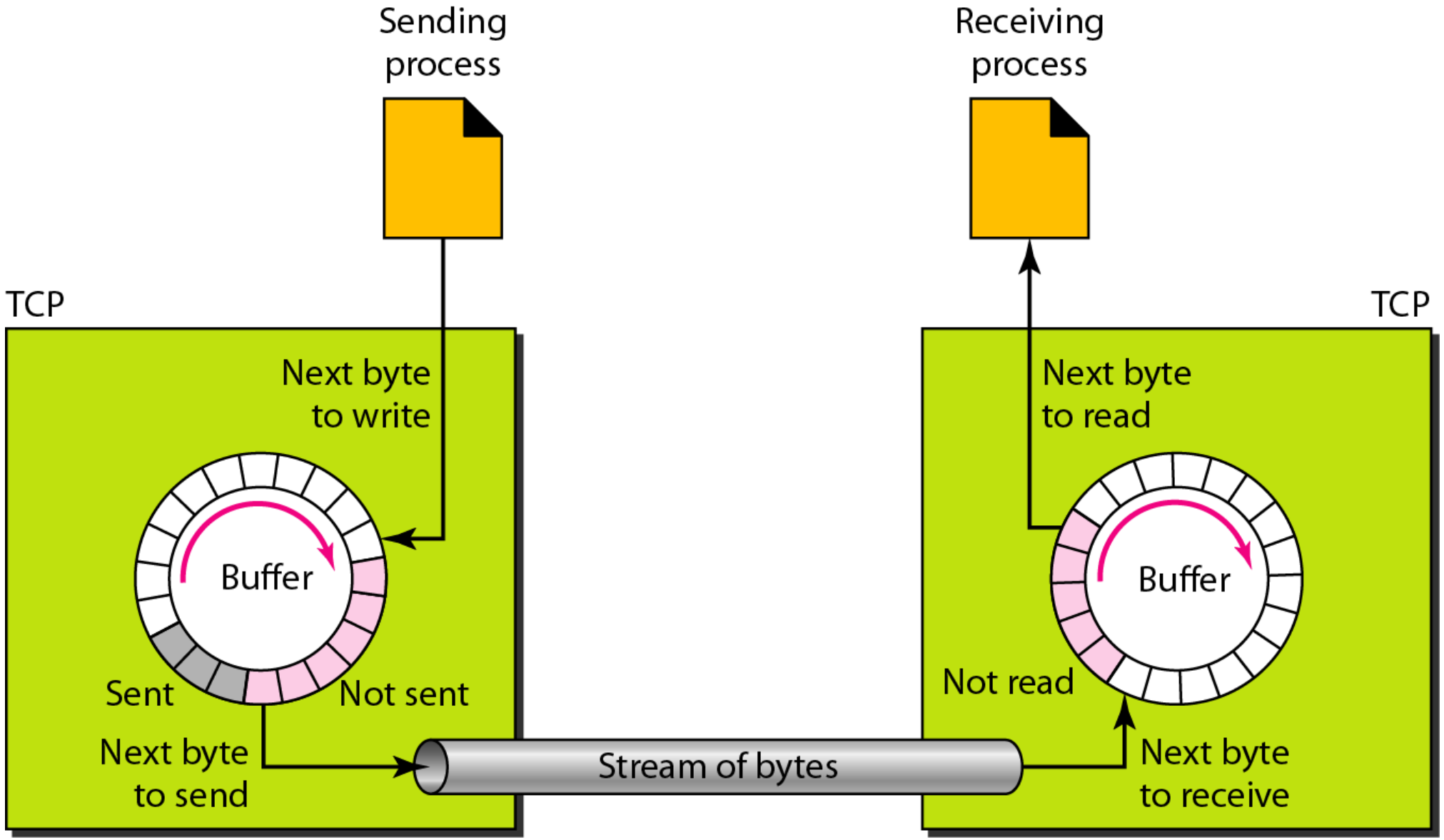
TRANSMISSION CONROL PROTOCOL (TCP)

- *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.*

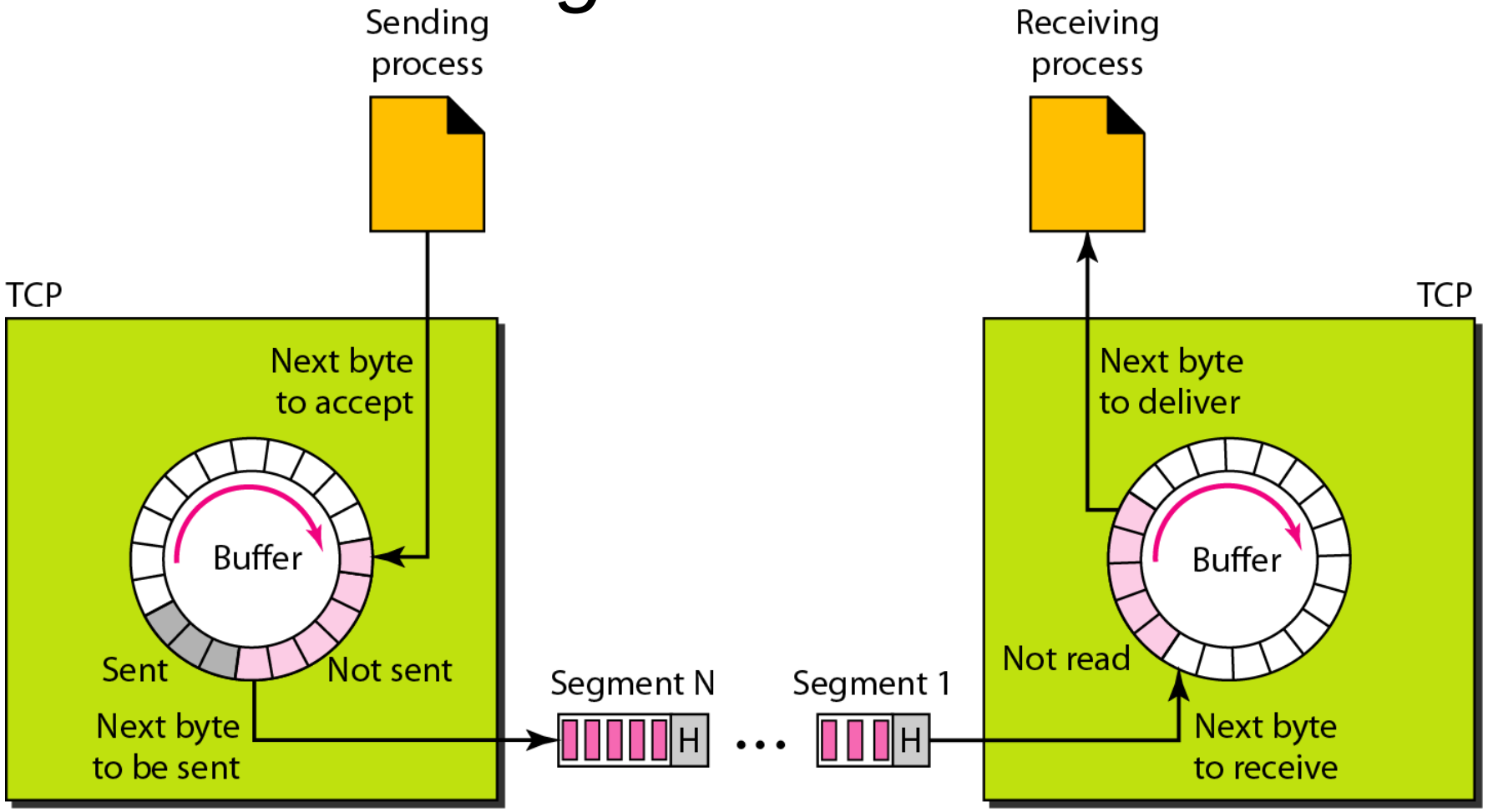
Stream delivery



Sending and receiving buffers



TCP segments



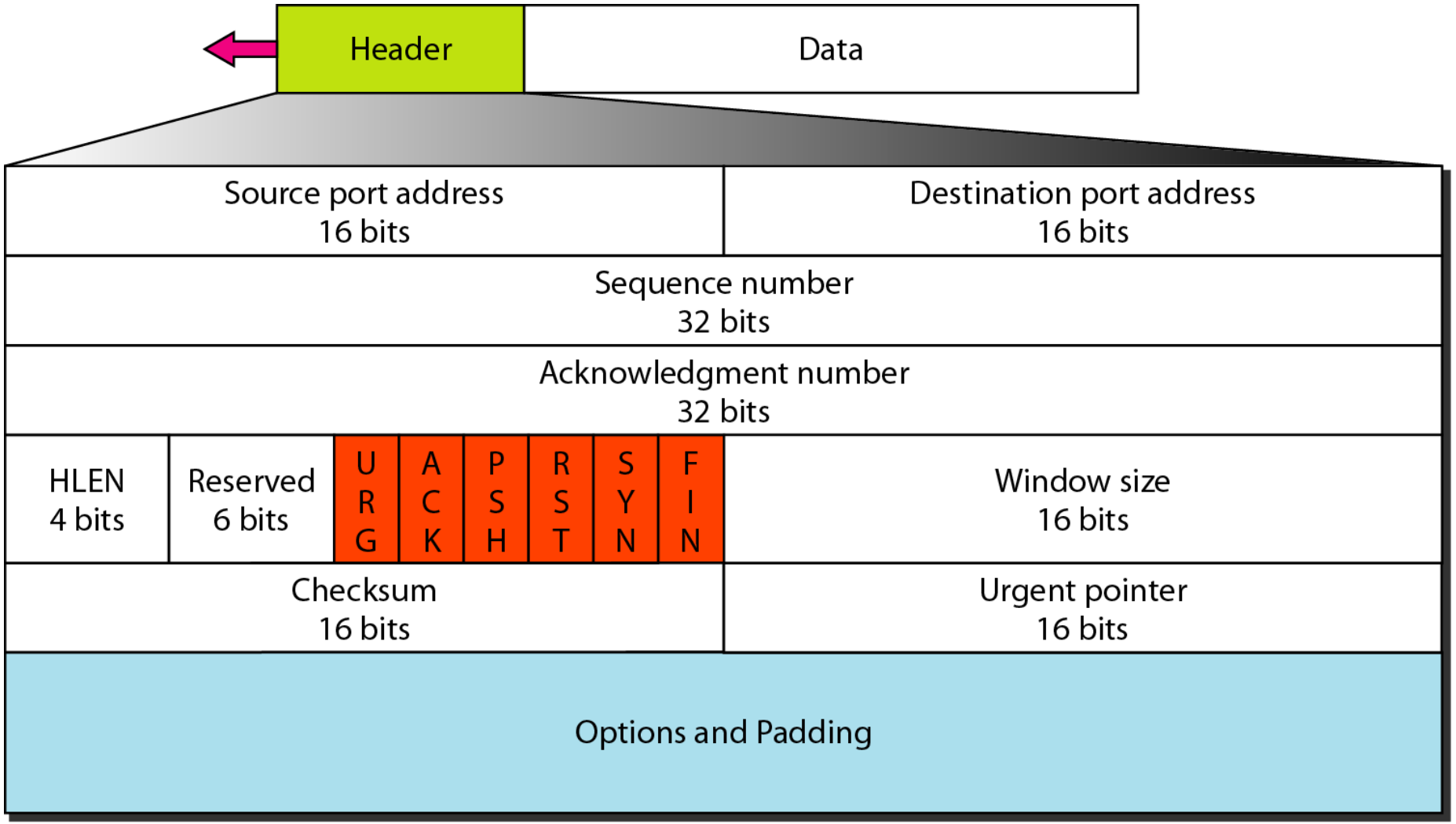
The bytes of data being transferred in each connection are numbered by TCP. The numbering starts with a randomly generated number.

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)**

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

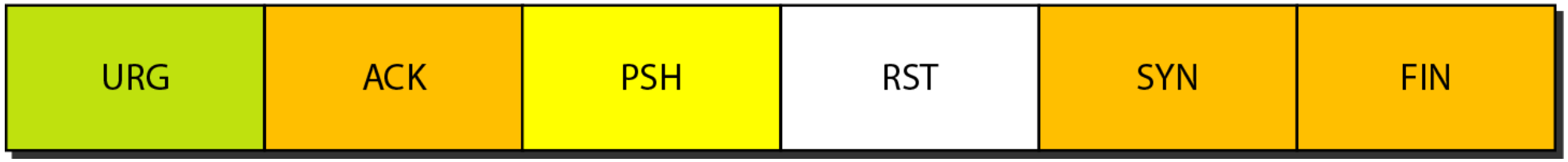
TCP segment format



Flags

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

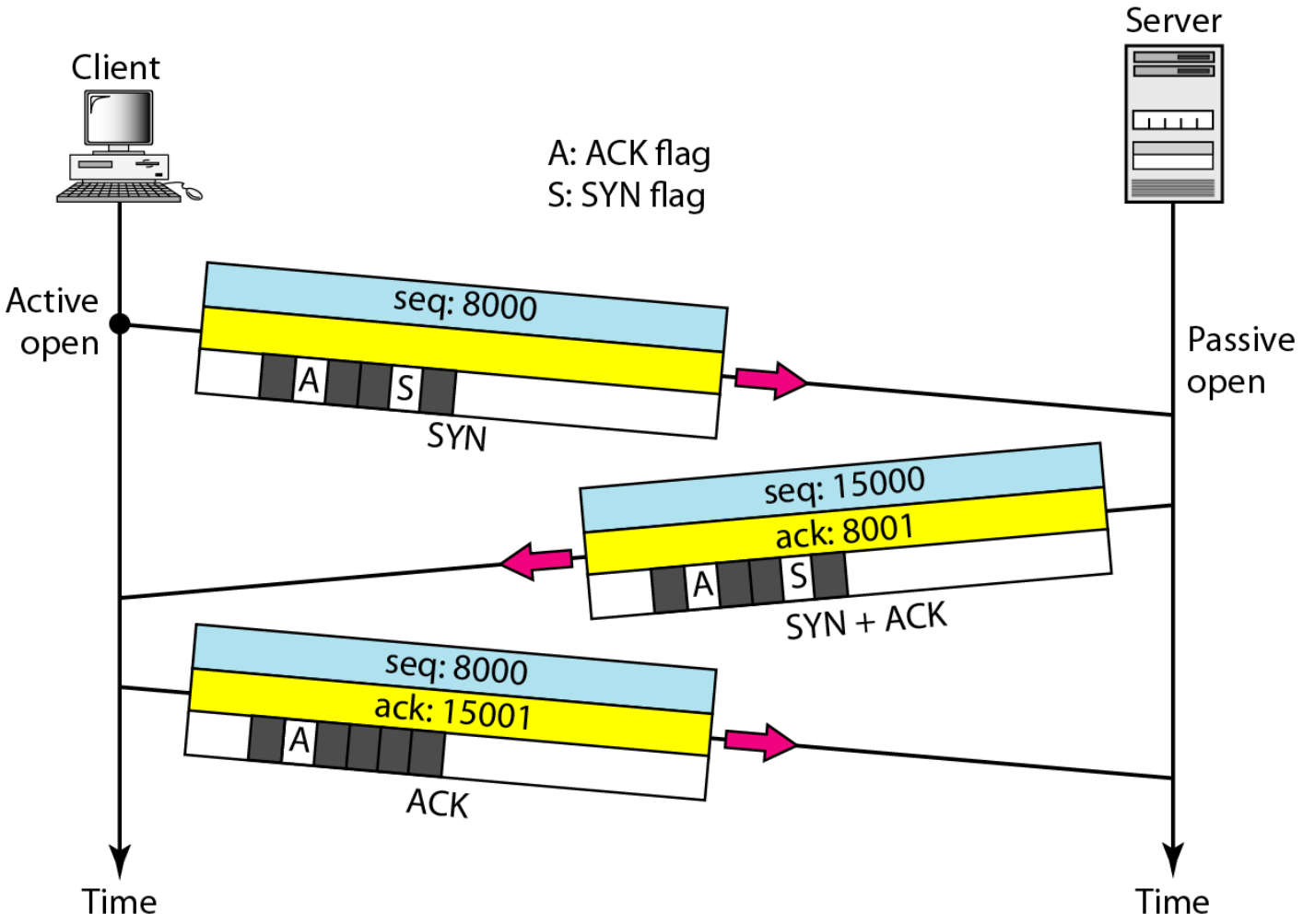
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection



Description of flags in the control field

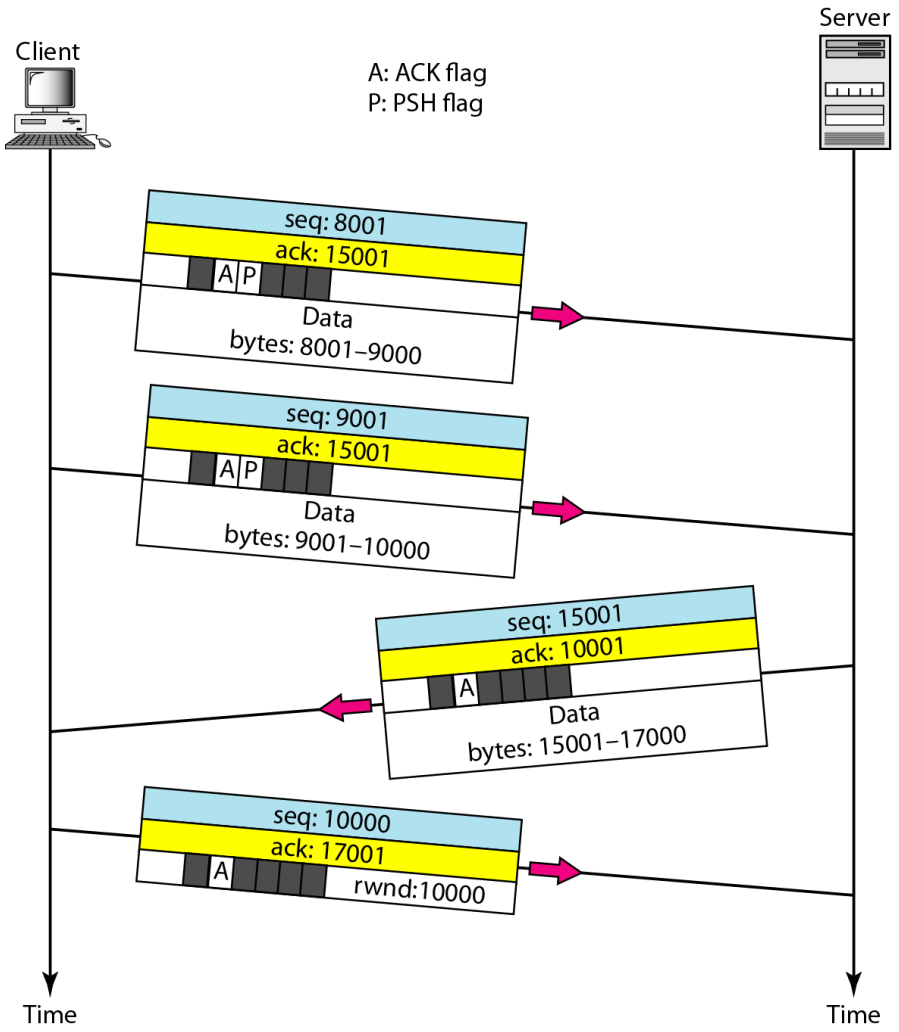
Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

TCP CONNECTION ESTABLISHMENT MECHANISM (3 WAY HANDSHAKING)

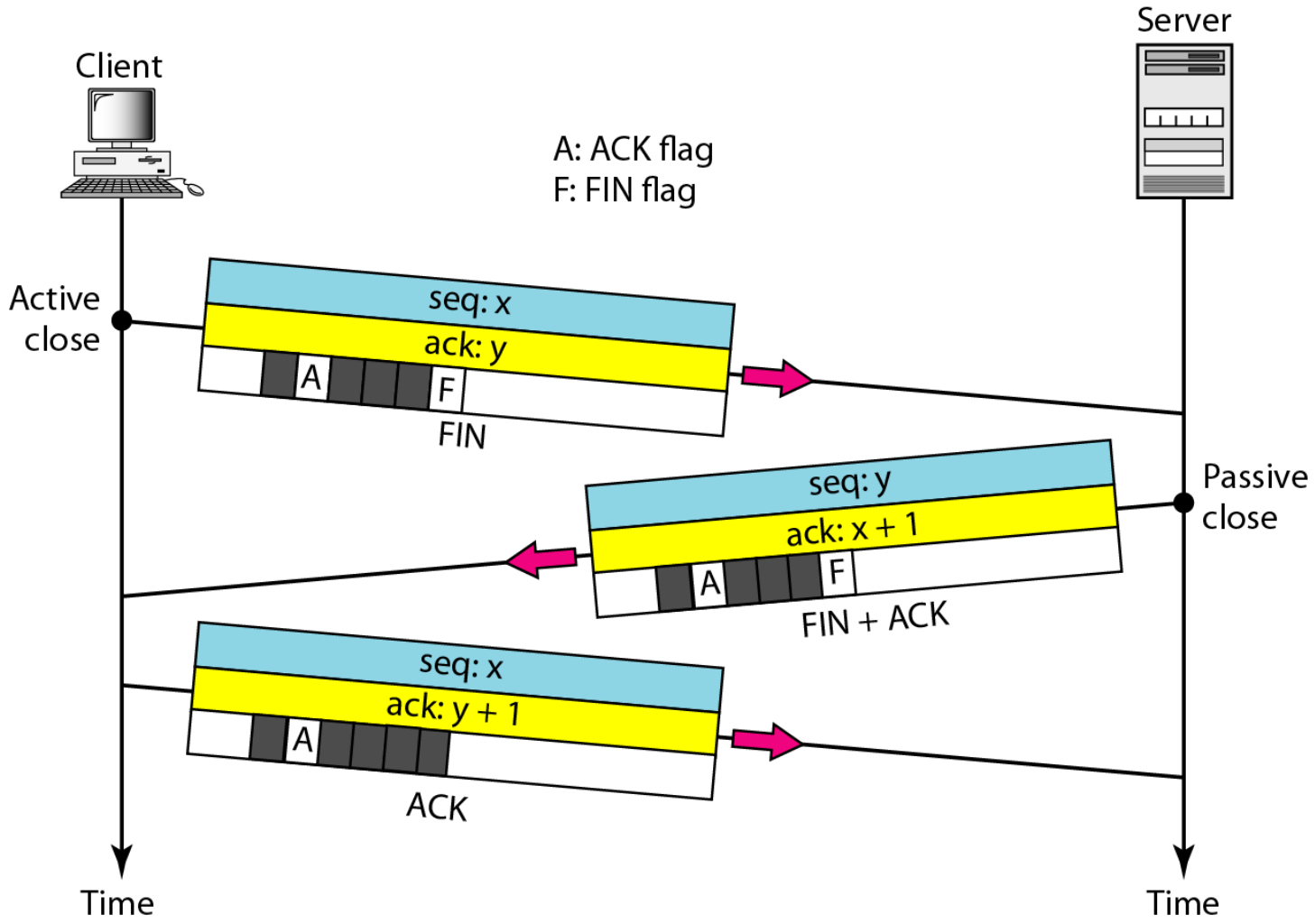


A SYN segment cannot carry data, but it consumes one sequence number.

Data transfer

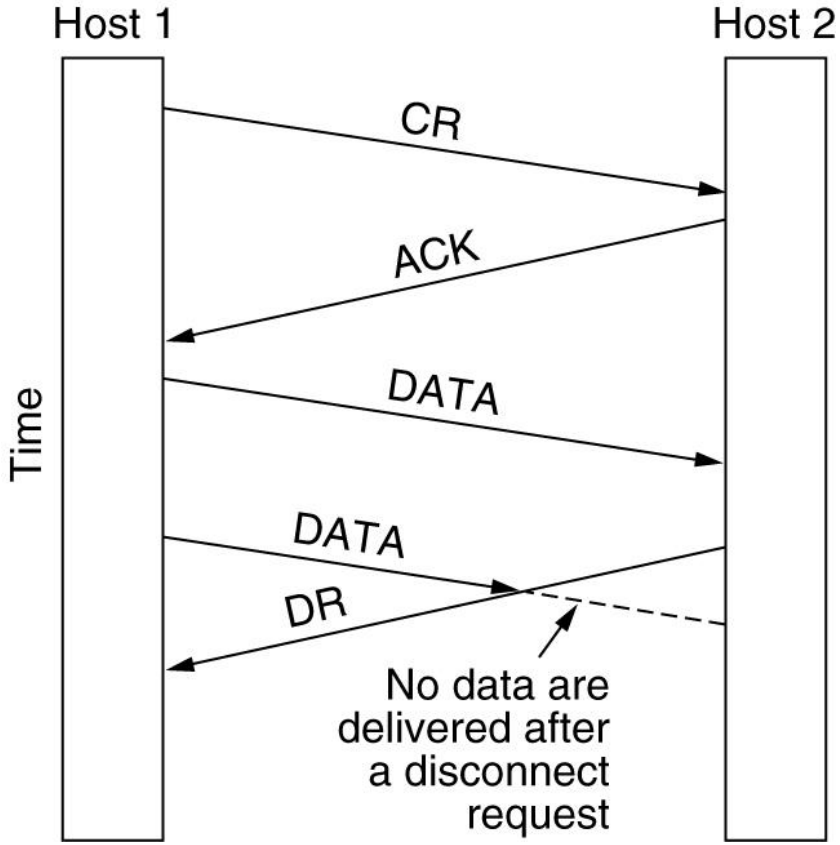


Connection termination using three-way handshaking



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

Connection Release

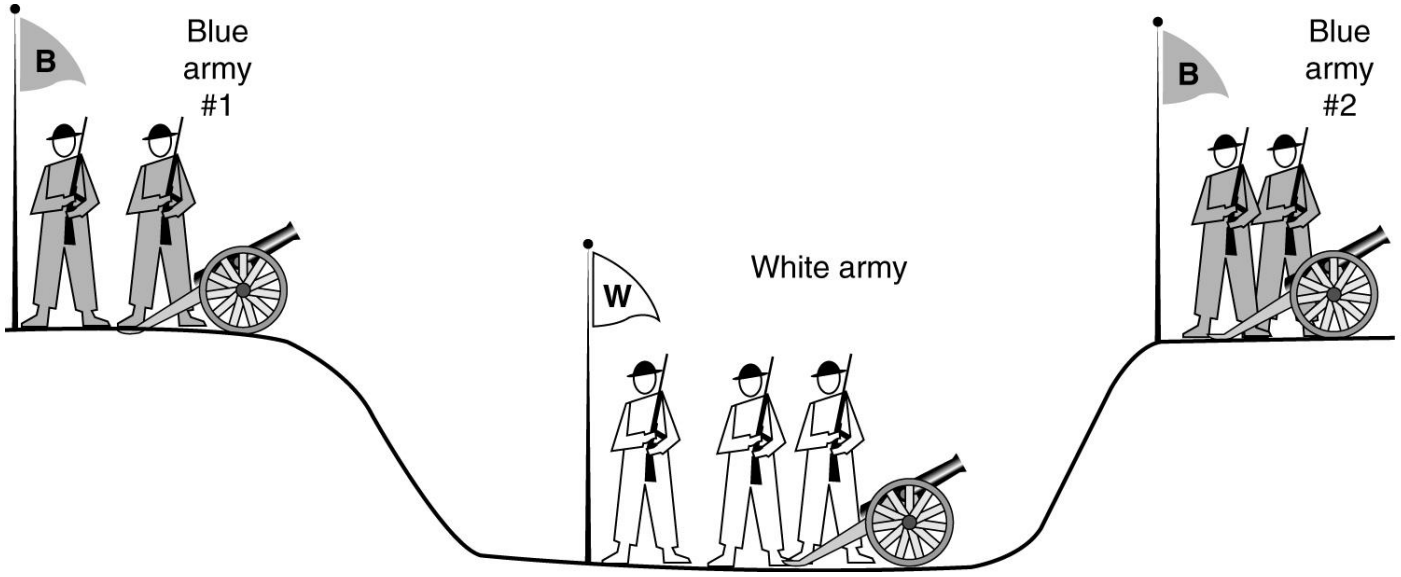


Abrupt disconnection with loss of data.

Connection Release

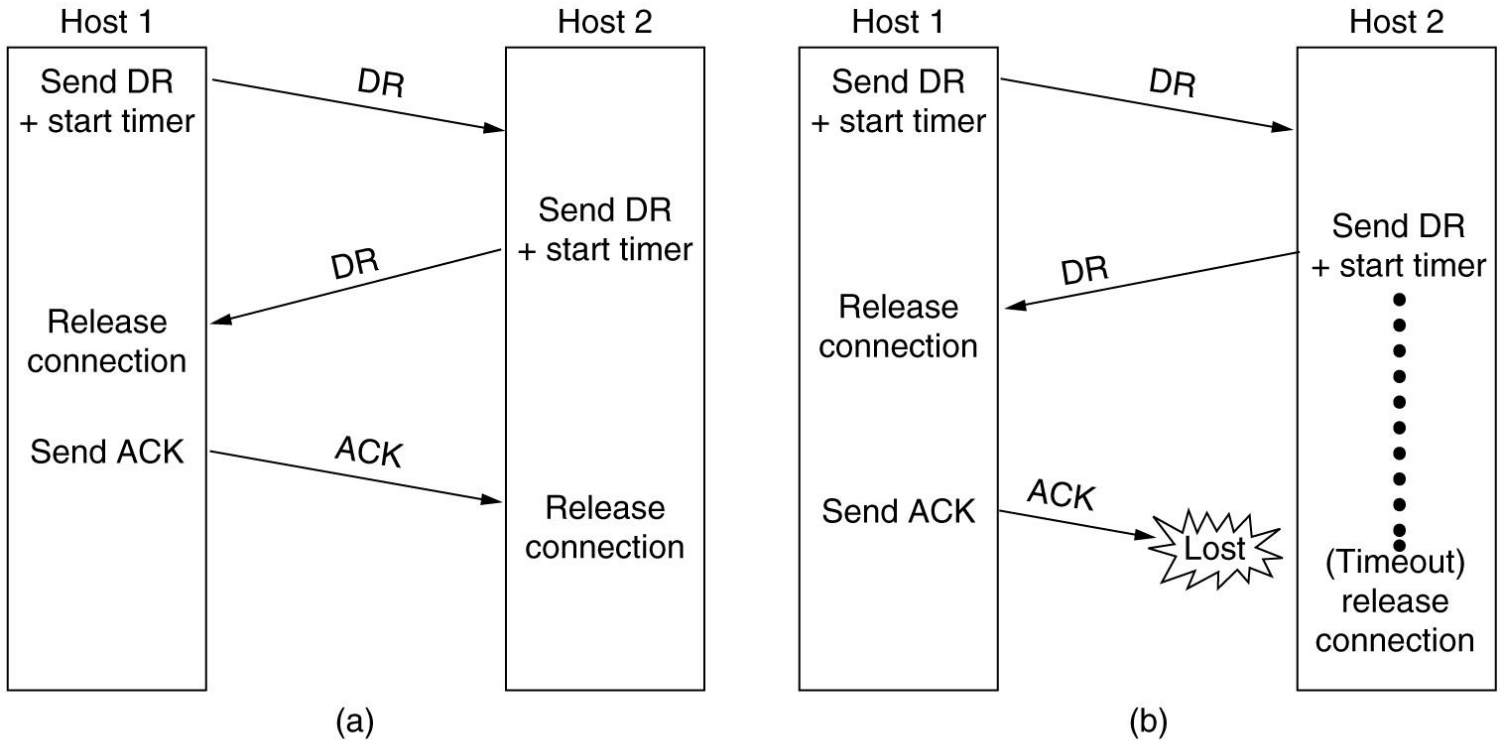
The two-army problem.

Simultaneous attack by blue army
Communication is unreliable

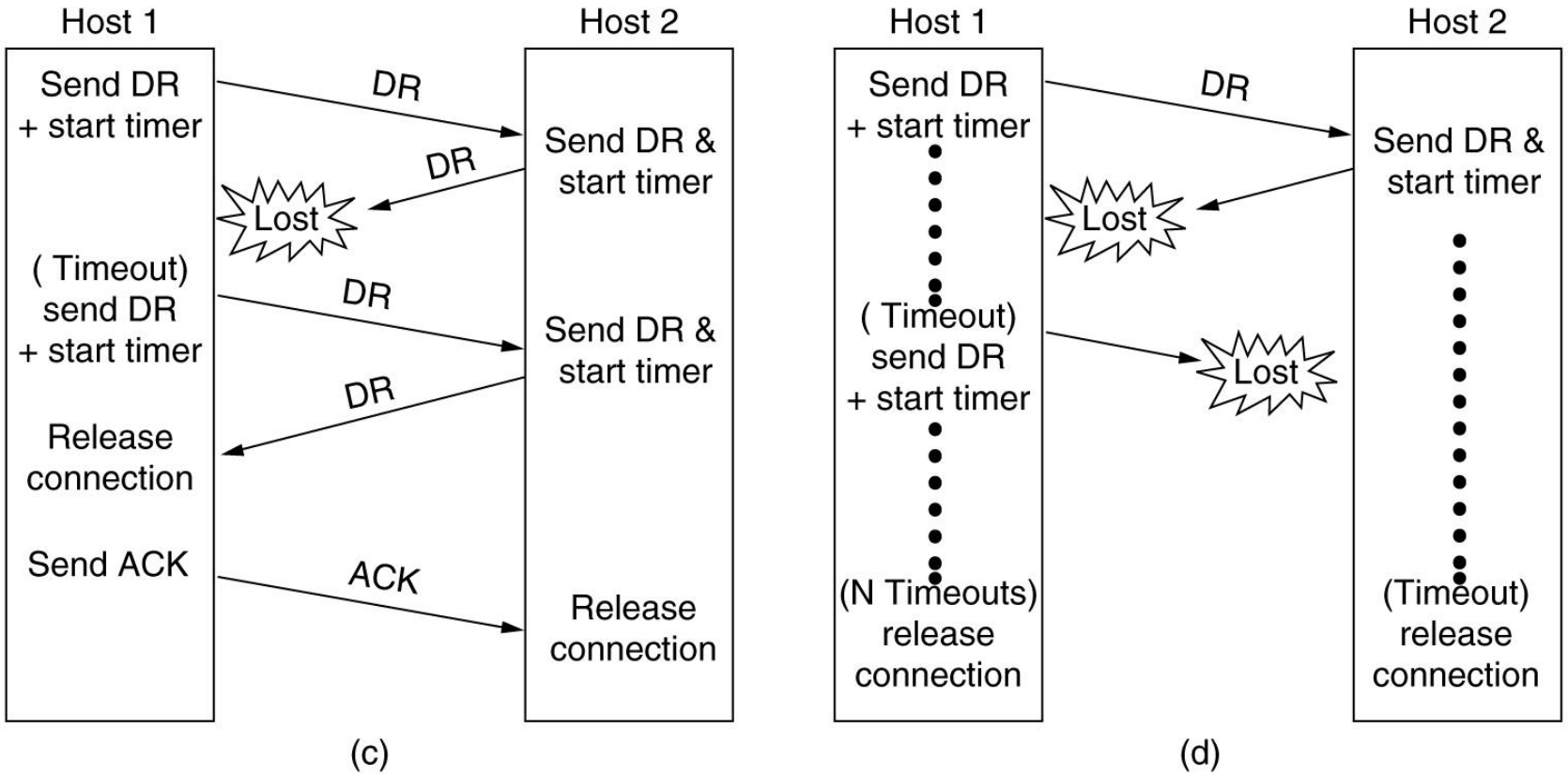


Connection Release

Four protocol scenarios for releasing a connection. (a) Normal case of a three-way handshake. (b) final ACK lost.

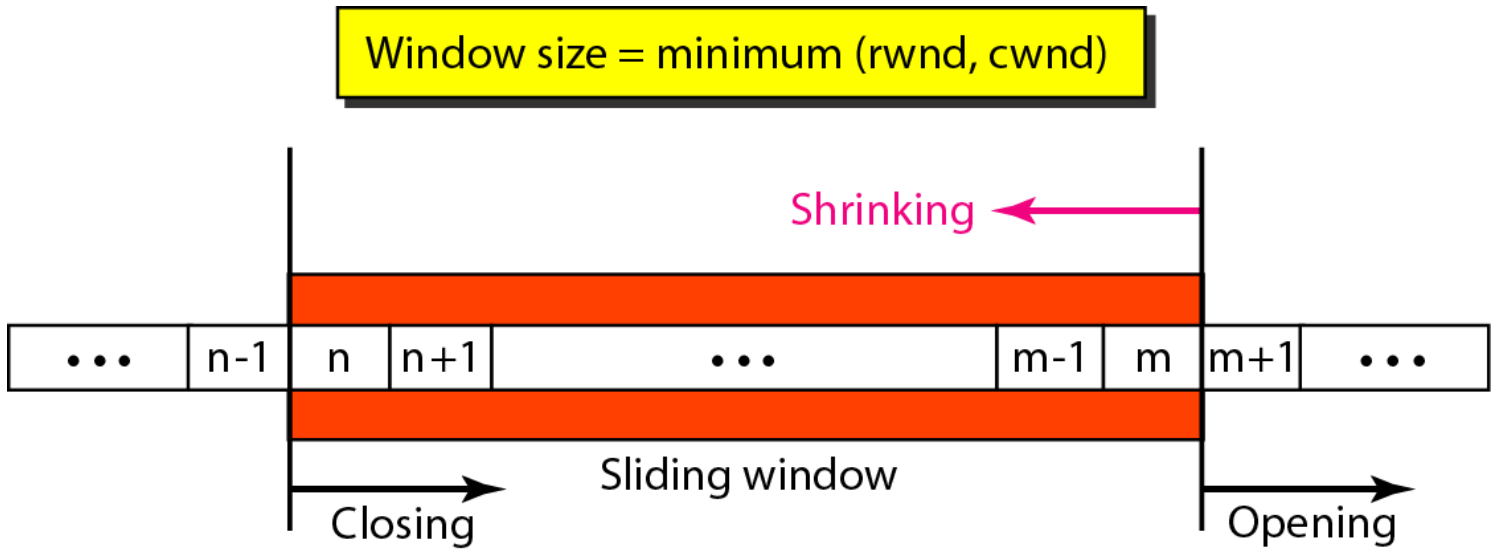


Connection Release



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

Sliding window



rwnd = receiver window size
cwnd = congestion window size

SILLY WINDOW SYNDROM INCREASES OVERHEAD (BY SENDING SMALL SEGMETS)

➢ Sending data in very small segments

Syndrome created by the Sender

- Sending application program creates data slowly (e.g. 1 byte at a time)
- Wait and collect data to send in a larger block
- How long should the sending TCP wait?
- Solution: Nagle's algorithm
- Nagle's algorithm takes into account (1) the speed of the application program that creates the data, and (2) the speed of the network that transports the data

2. Syndrome created by the Receiver

- Receiving application program consumes data slowly (e.g. 1 byte at a time)
- The receiving TCP announces a window size of 1 byte. The sending TCP sends only 1 byte…
- Solution 1: Clark's solution
- Sending an ACK but announcing a window size of zero until there is enough space to accommodate a segment of max. size or until half of the buffer is empty

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.

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?

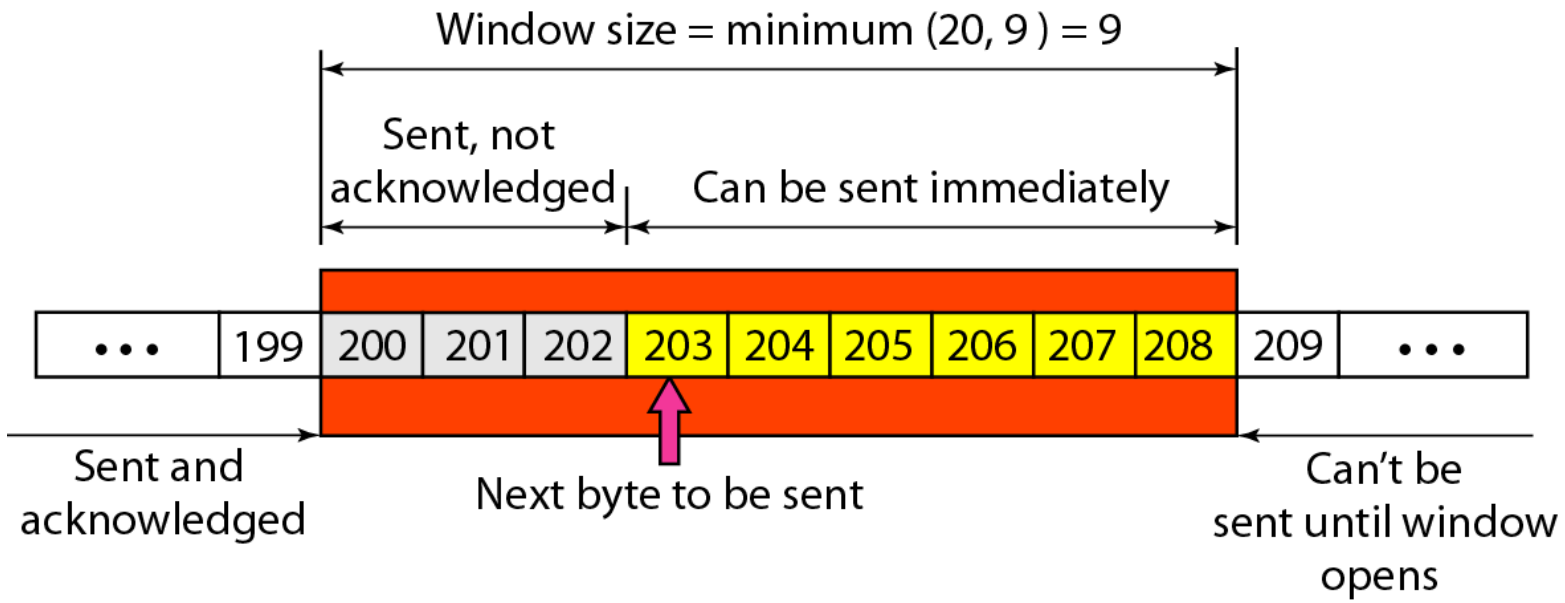
Solution

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.

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.

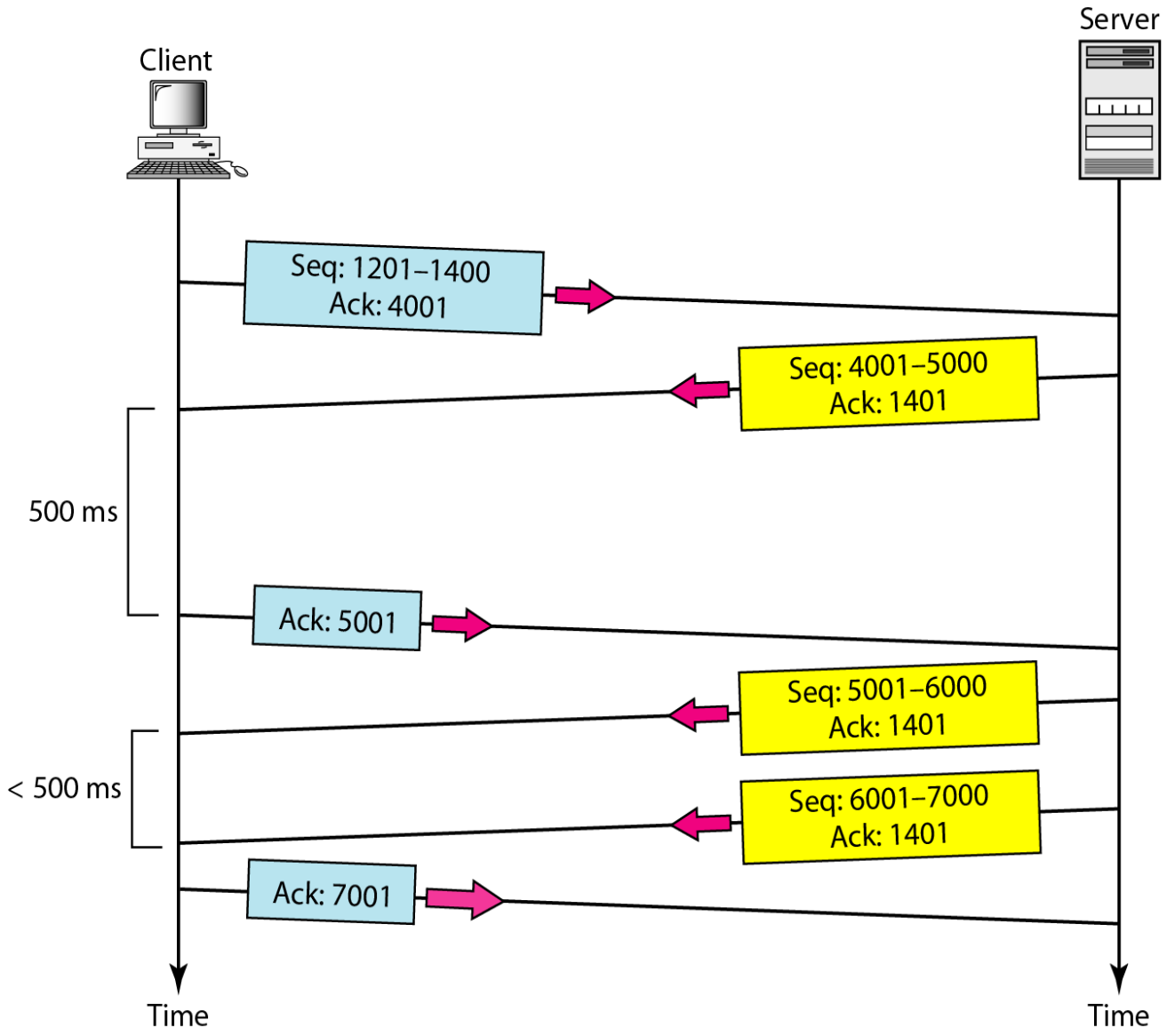


- *The sender has sent bytes up to 202.*
- *We assume that cwnd is 20 (in reality this value is thousands of bytes).*
- *The receiver has sent an acknowledgment number of 200 with an rwnd of 9 bytes (in reality this value is thousands of bytes).*
- *The size of the sender window is the minimum of rwnd and cwnd, or 9 bytes.*
- *Bytes 200 to 202 are sent, but not acknowledged.*
- *Bytes 203 to 208 can be sent without worrying about acknowledgment.*
- *Bytes 209 and above cannot be sent.*

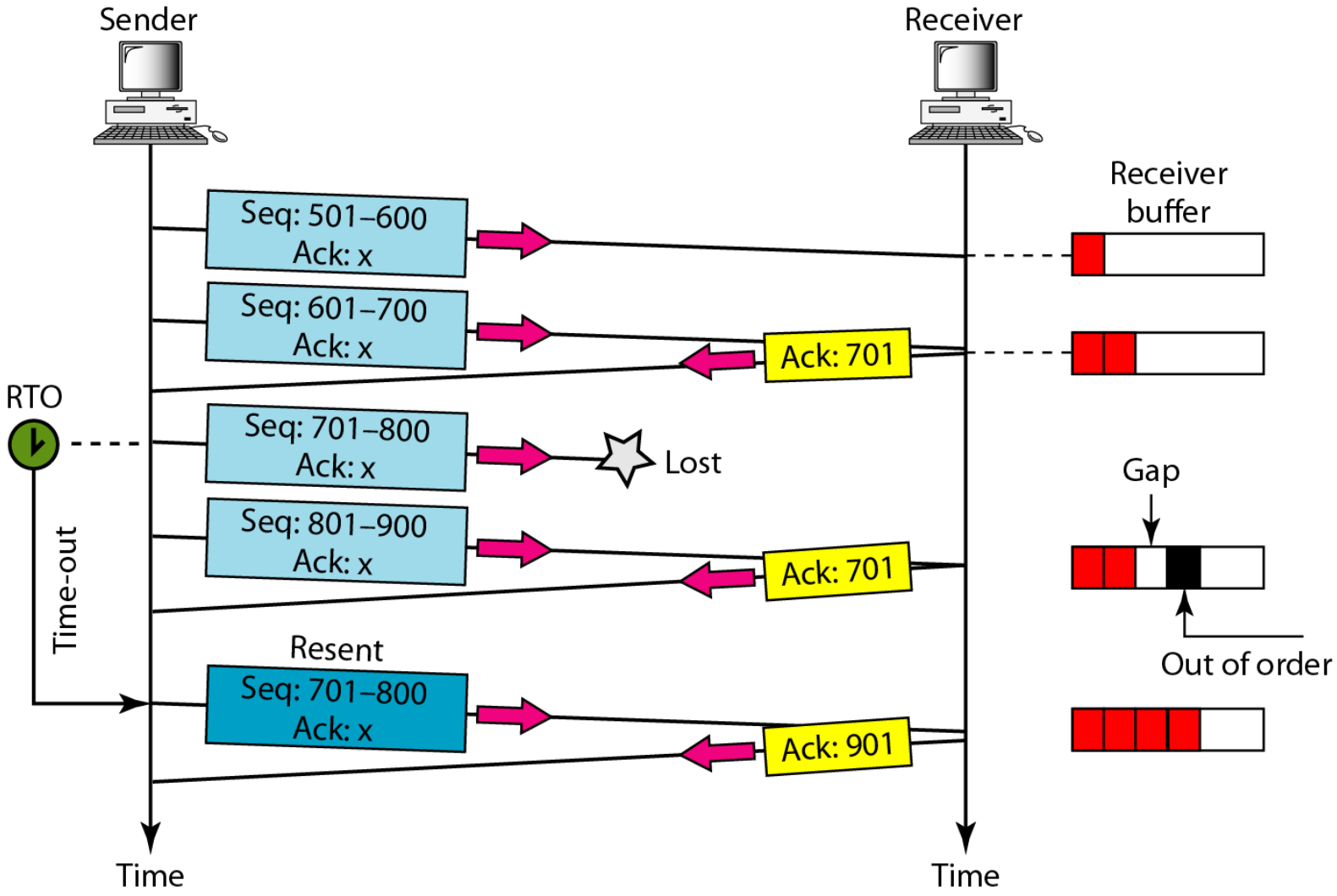
Some points about TCP sliding windows:

- 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.

Normal operation



Lost segment



Fast retransmission