Transmission Control Protocol (TCP)

Slides compiled by: Sanghamitra De

TCP

- TCP lies between the application layer and the network layer.
- TCP serves as the intermediary between the application programs and the network operations.

TCP Services

- **Process-to-Process Communication**
- **Stream Delivery Service**
  - ✓ ***Sending and Receiving Buffers*** *(*circular array of 1-byte locations*)*
  - ✓ ***Segments***
- **Full-Duplex Communication**
- **Multiplexing and Demultiplexing**
- **Connection-Oriented Service**
- **Reliable Service**

Table 15.1 *Well-known Ports used by TCP*

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

Table 15.1 *Well-known Ports used by TCP (continued)*

Port	*Protocol*	*Description*
19	Chargen	Returns a string of characters
20 and 21	FTP	File Transfer Protocol (Data and Control)
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

Figure 15.2 *Stream delivery*

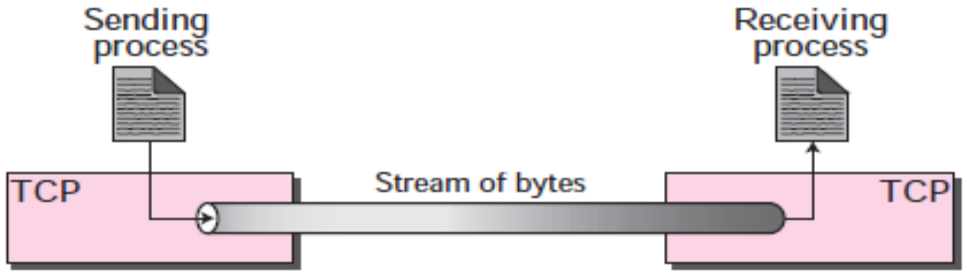


Figure 15.3 *Sending and receiving buffers*

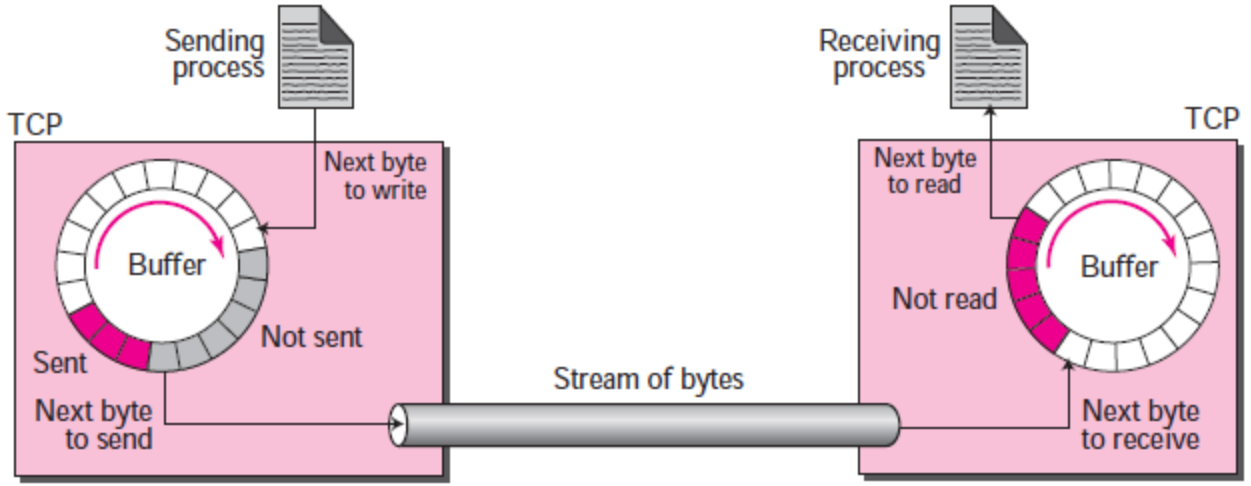
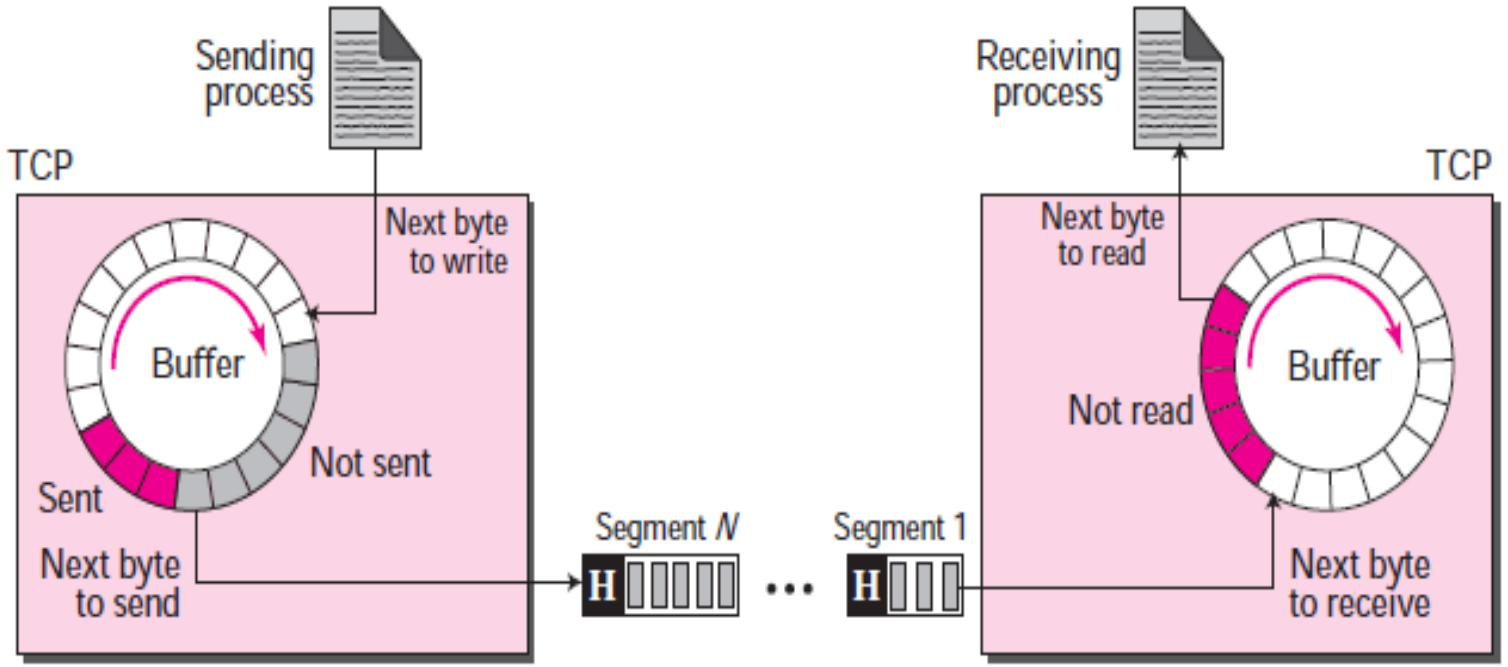


Figure 15.4 *TCP segments*

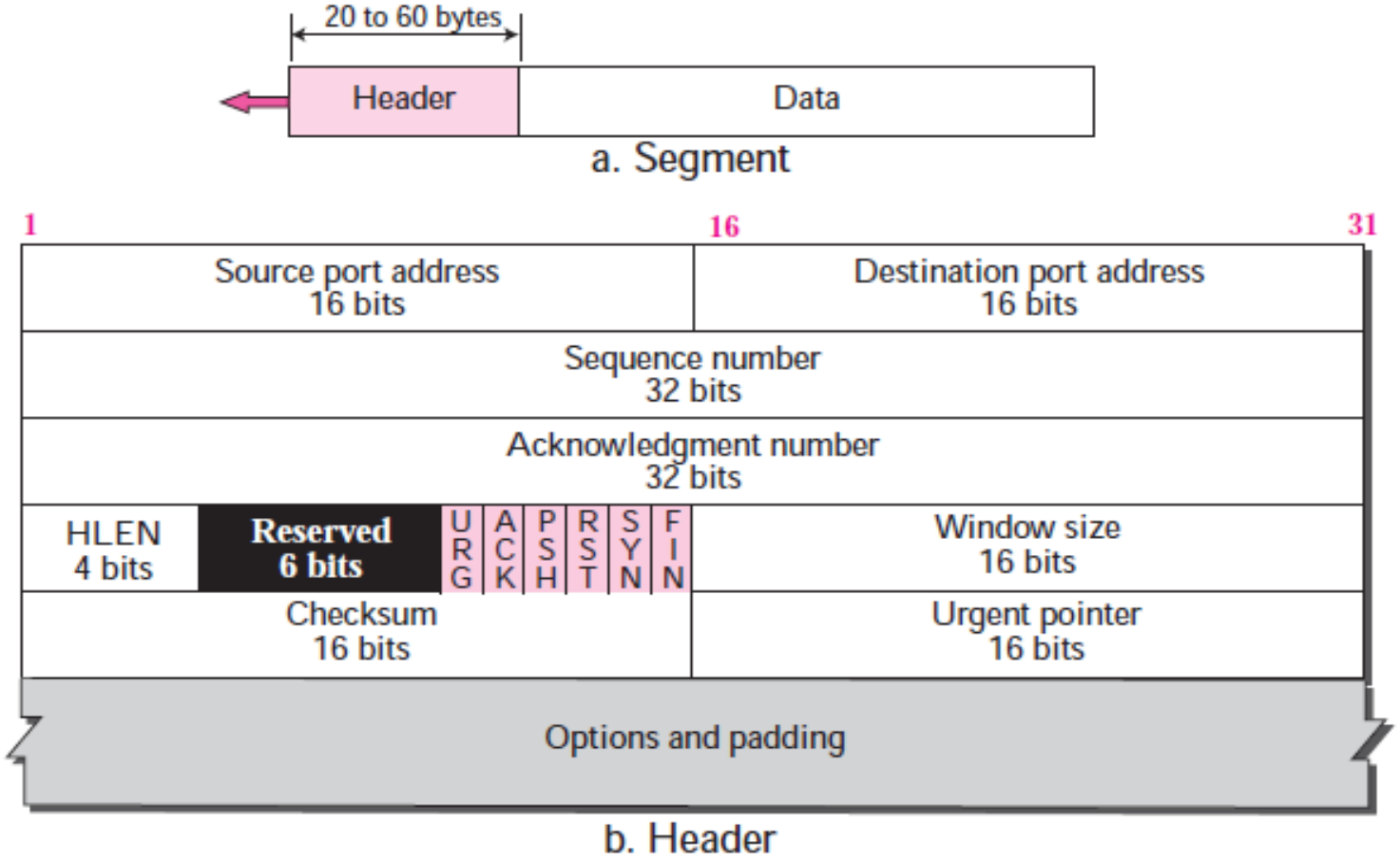


TCP Features

- **Numbering System**
  - ✓ ***Byte Number***
  - ✓ ***Sequence Number***
  - ✓ ***Acknowledgment Number***
- **Flow Control**
- **Error Control**
- **Congestion Control**

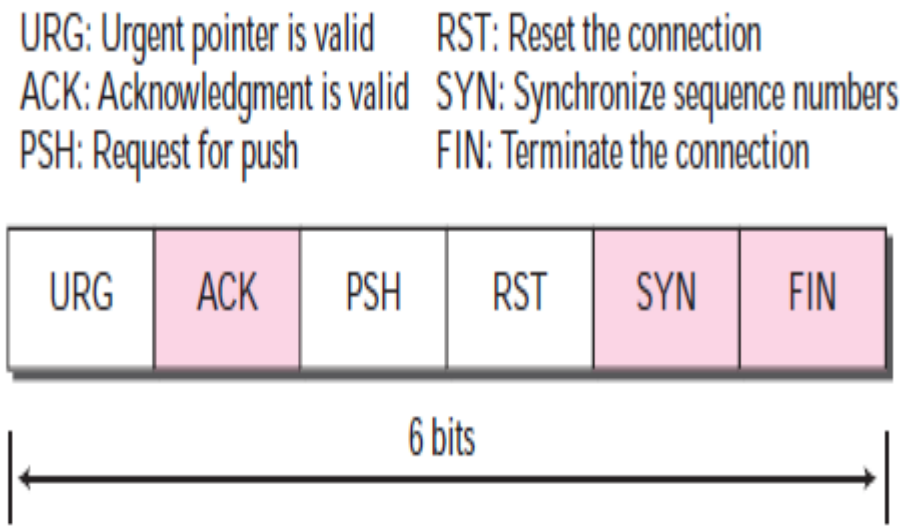
TCP Segment Format

Figure 15.5 *TCP segment format*



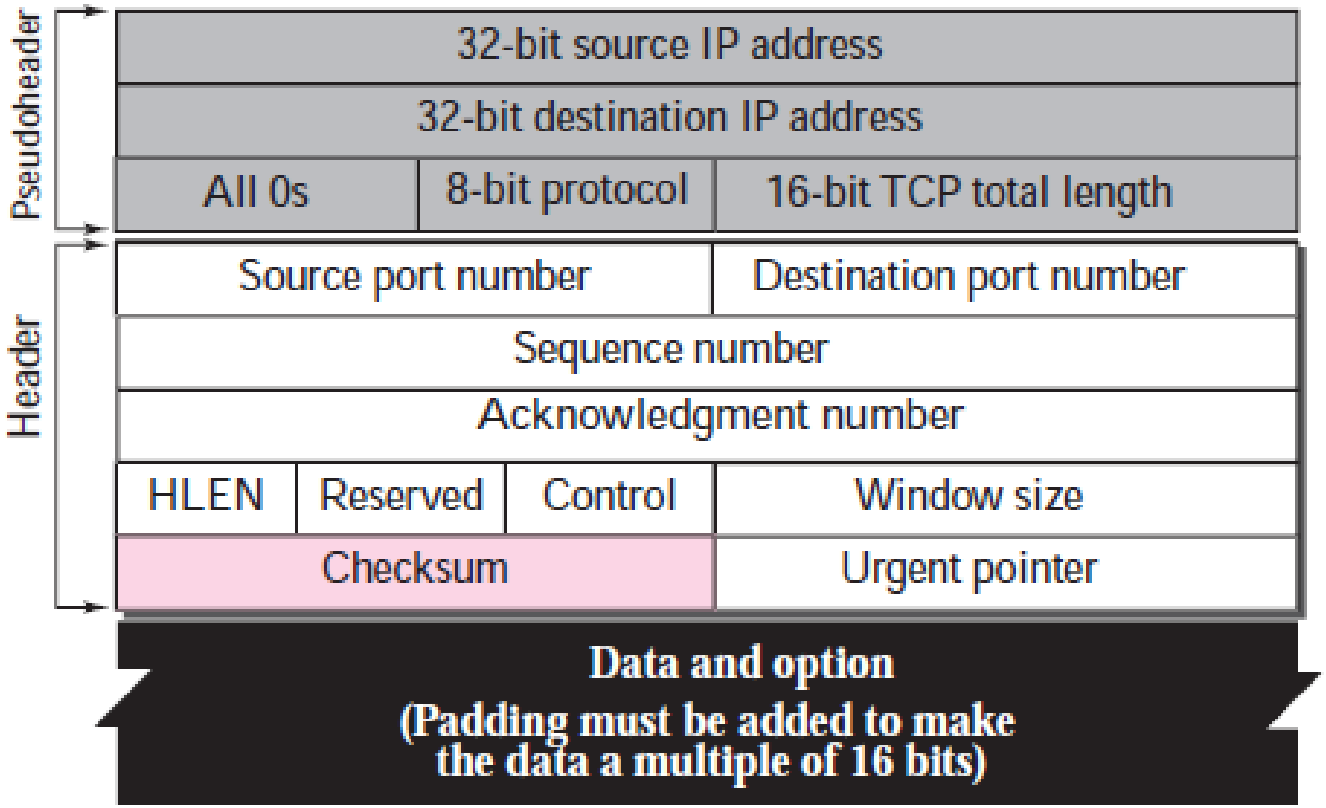
Control field in TCP Segment Format

Figure 15.6 *Control field*



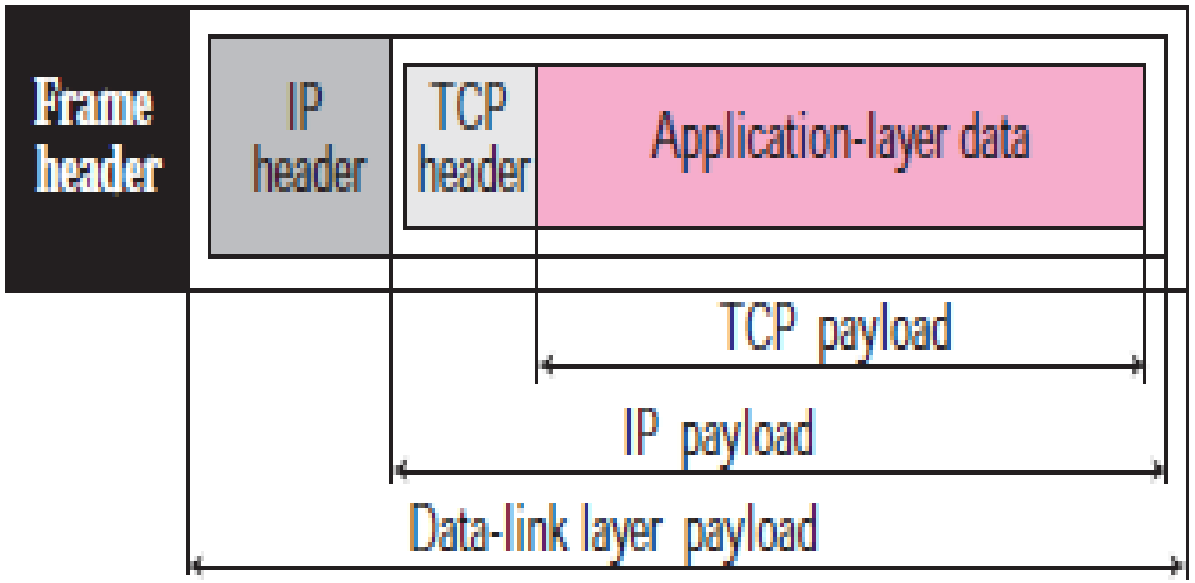
TCP Segment Format

Figure 15.7 *Pseudoheader added to the TCP datagram*



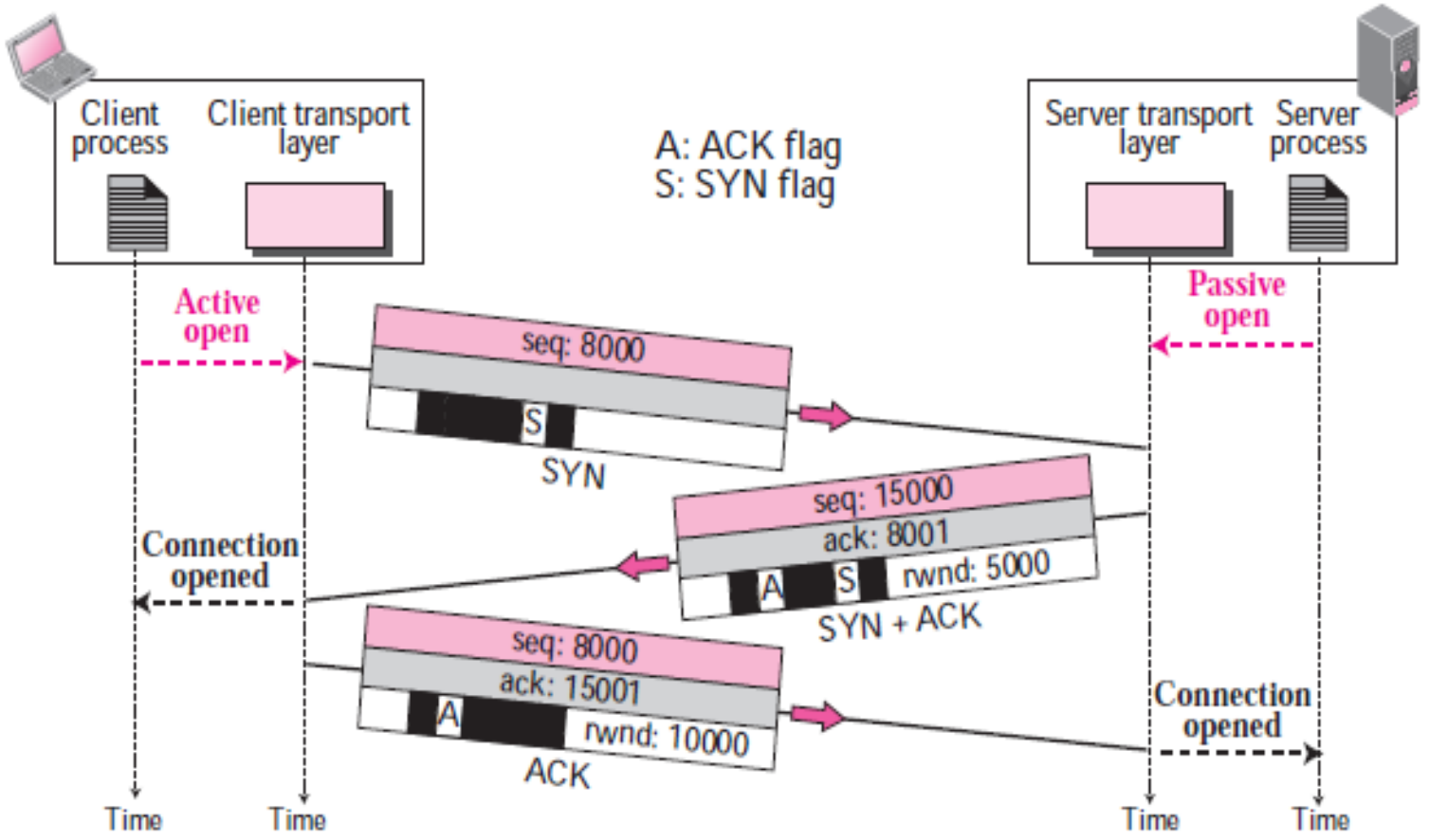
TCP Segment Encapsulation

Encapsulation



TCP Connection Establishment

Figure 15.9 *Connection establishment using three-way handshaking*

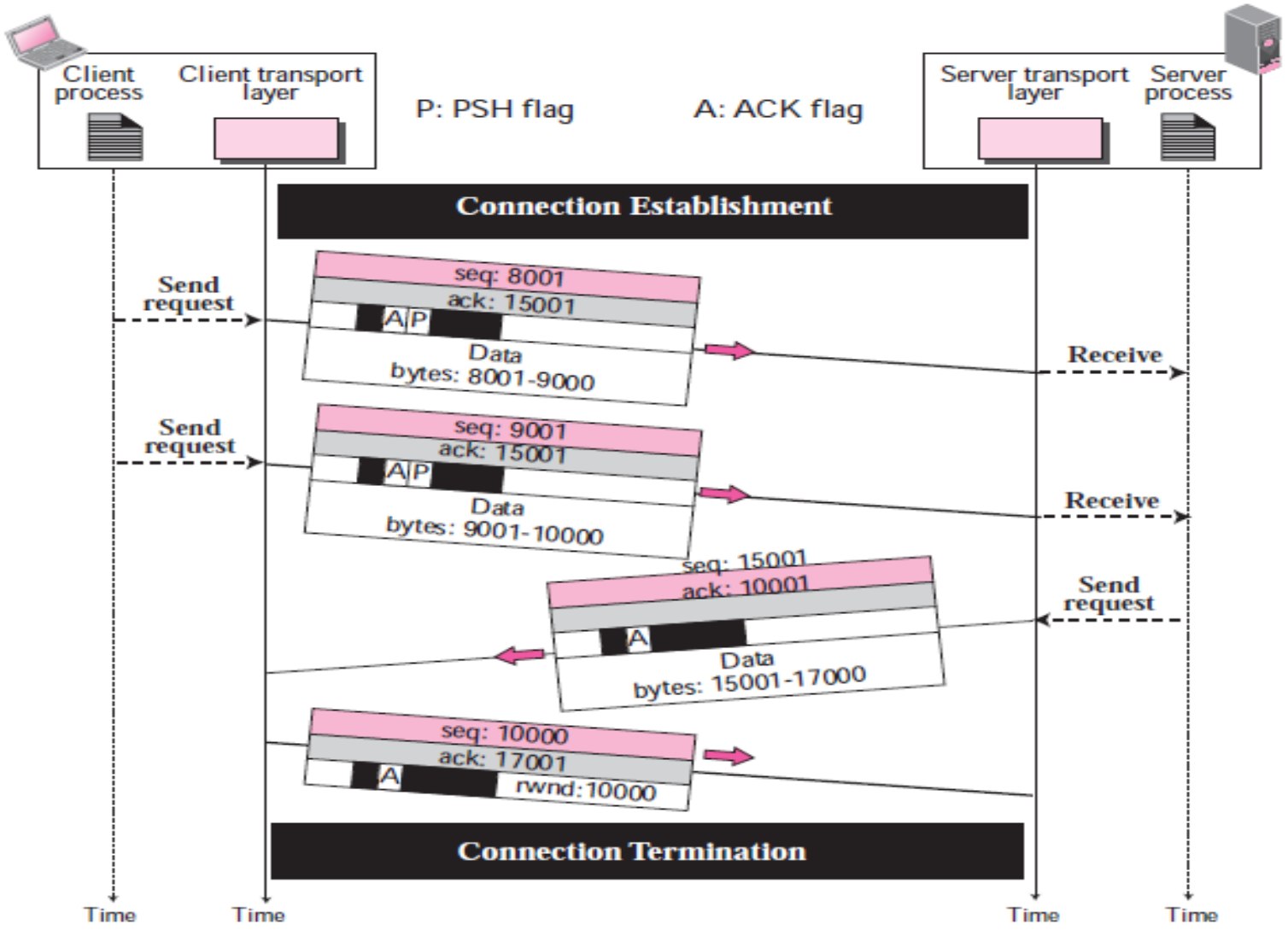


Uncommon Issues in 3-way handshake

- ***Simultaneous Open***
- ***SYN Flooding Attack***

TCP Data Transfer

Figure 15.10 *Data transfer*

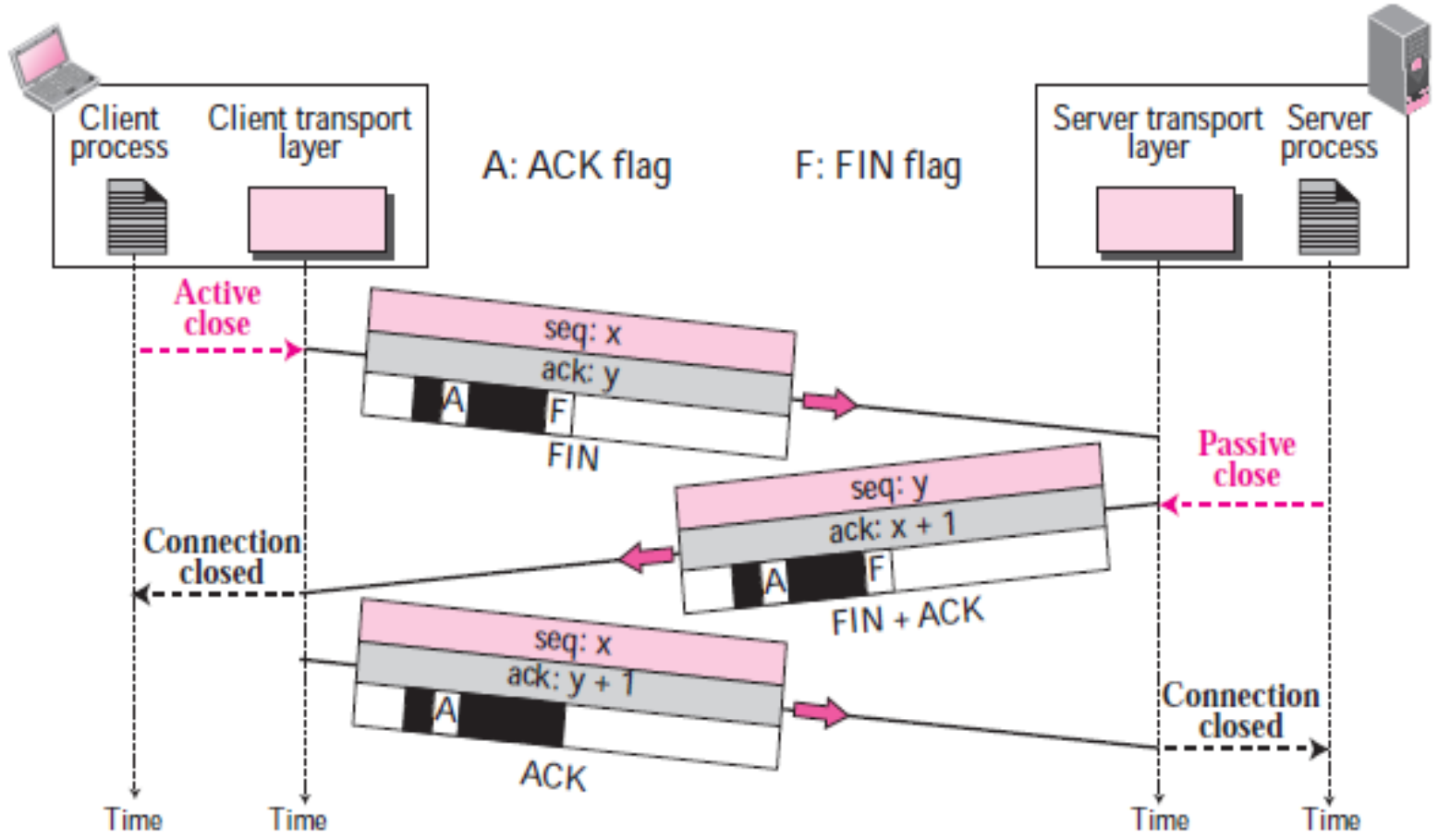


TCP Data Transfer

- ***Pushing Data***
- ***Urgent Data***

TCP Connection Termination

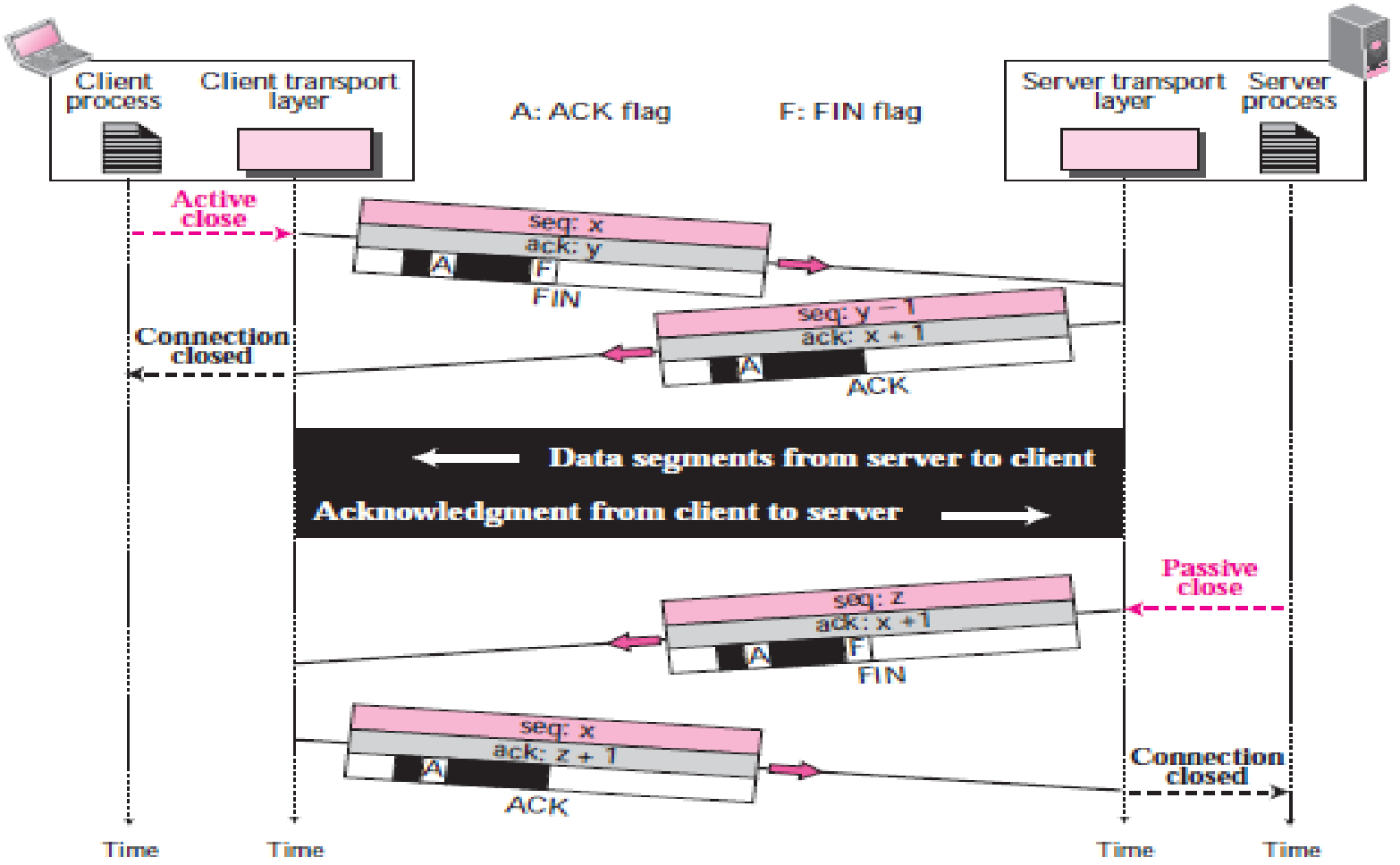
Figure 15.11 *Connection termination using three-way handshaking*



Half Close

- One end can stop sending data while still receiving data.
- This is called a halfclose.
- Either the server or the client can issue a half-close request.
- It can occur when the server needs all the data before processing can begin. E.g. sorting.

Figure 15.12 *Half-close*



Although the client has received sequence number y − 1 and is expecting y, the server sequence number is still y − 1. When the connection finally closes, the sequence number of the last ACK segment is still x, because no sequence numbers are consumed during data transfer in that direction.

Connection Reset

- Denying a Connection
  - ✓ TCP on one side has requested a connection to a non-existent port.
- Aborting a Connection
  - ✓ TCP may want to abort an existing connection due to an abnormal situation.
- Terminating an Idle Connection
  - ✓ TCP on one side may discover that the TCP on the other side has been idle for a long time.

TCP Message types

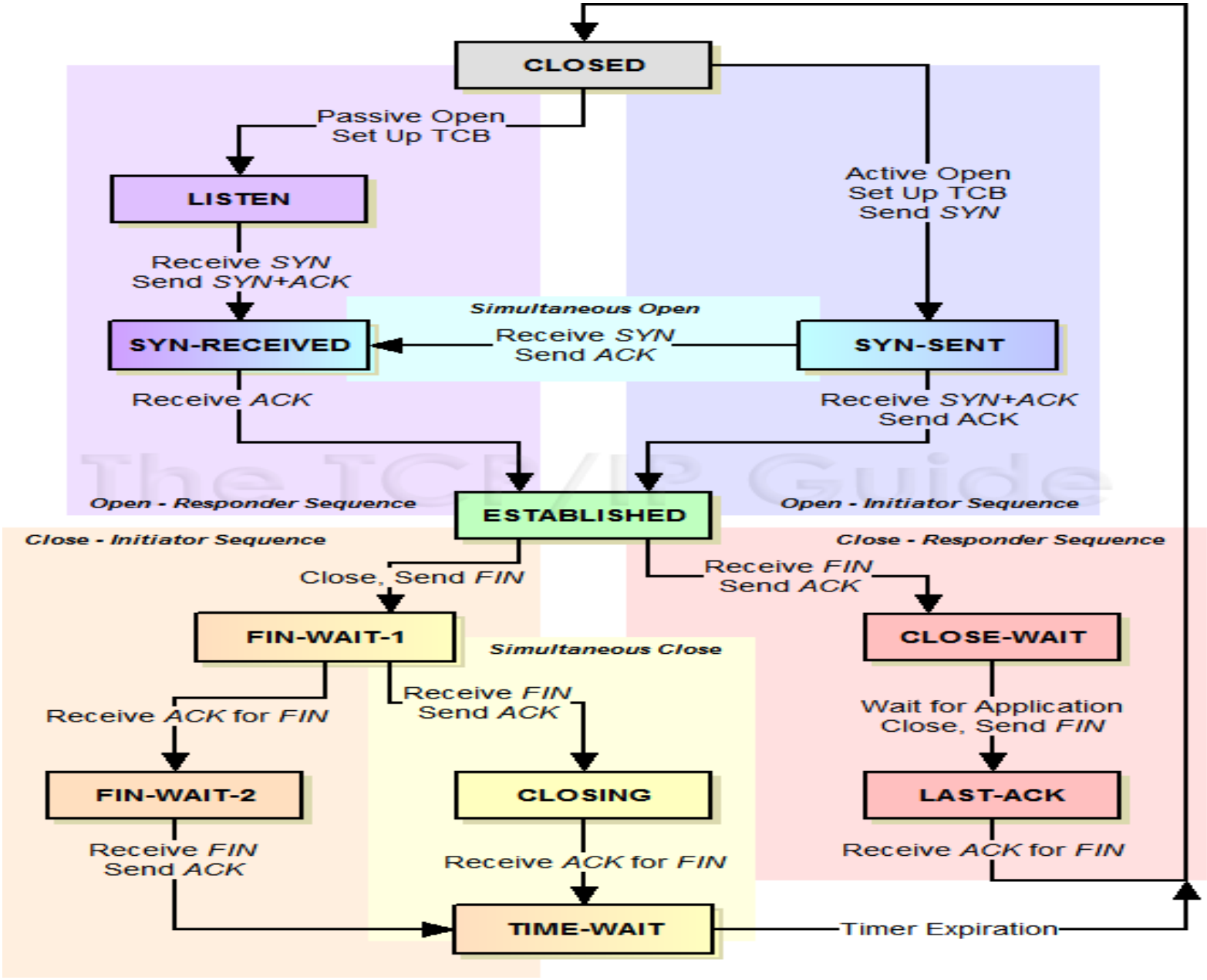
- ***SYN:*** A *synchronize* message, used to initiate and establish a connection. It is so named since one of its functions is to synchronizes sequence numbers between devices.

- ***FIN:*** A *finish* message, which is a TCP segment with the *FIN* bit set, indicating that a device wants to terminate the connection.

- ***ACK:*** An *acknowledgment*, indicating receipt of a message such as a *SYN* or a *FIN*.

TCP States & State Transition Diagram

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


CLOSED
Passive Open
Set Up TCB
Active Open
Set Up TCB
Send SYN
LISTEN
Receive SYN
Send SYN+ACK
Simultaneous Open
SYN-RECEIVED
Receive SYN
Send ACK
SYN-SENT
Receive ACK
Receive SYN+ACK
Send ACK
Open - Responder Sequence
Open - Initiator Sequence
ESTABLISHED
Close - Initiator Sequence
Close - Responder Sequence
Close, Send FIN
Receive FIN
Send ACK
FIN-WAIT-1
Simultaneous Close
CLOSE-WAIT
Receive ACK for FIN
Receive FIN
Send ACK
Wait for Application
Close, Send FIN
FIN-WAIT-2
CLOSING
LAST-ACK
Receive FIN
Send ACK
Receive ACK for FIN
Receive ACK for FIN
TIME-WAIT
Timer Expiration

Flow Control in TCP

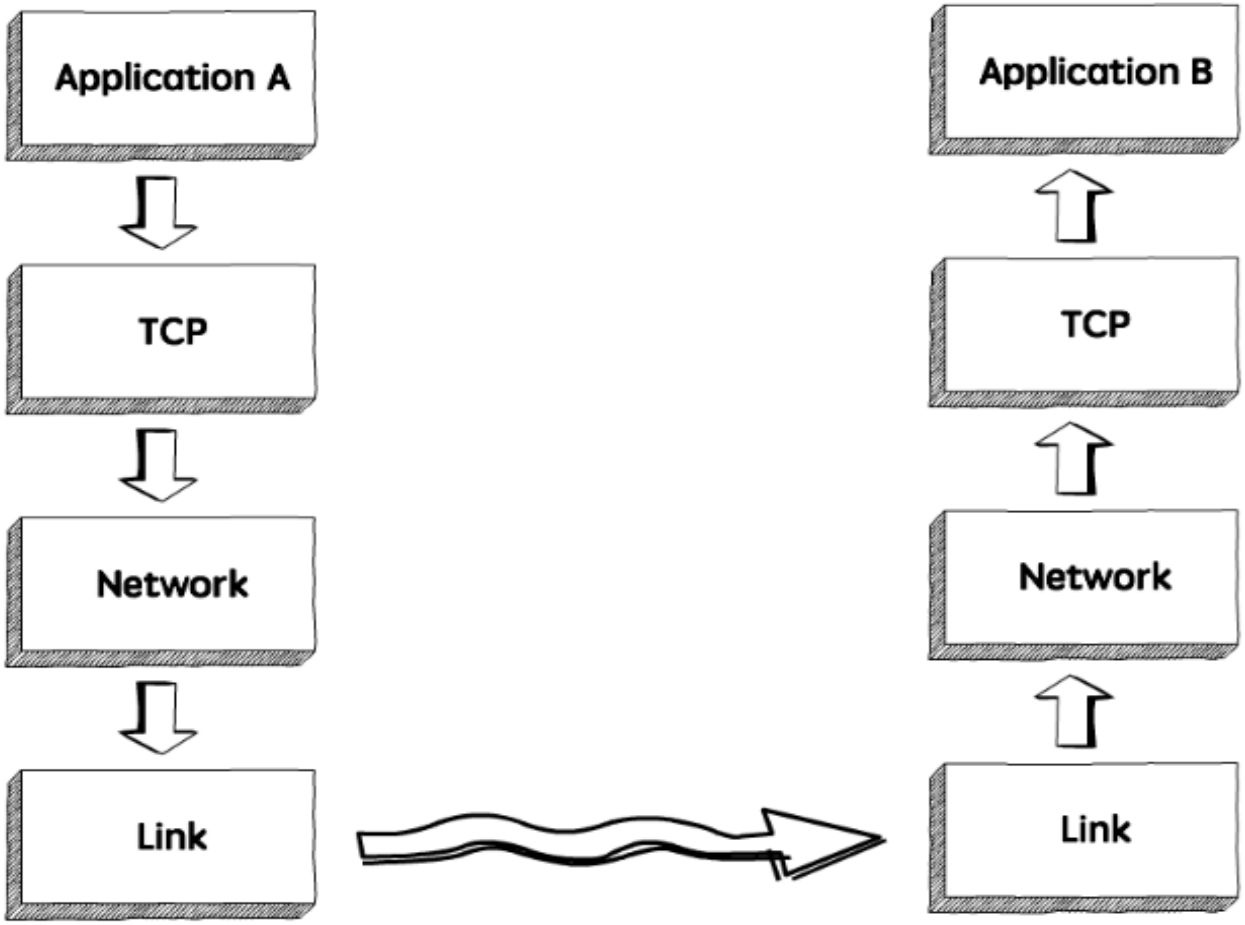
- TCP's flow control is a mechanism to ensure the sender is not overwhelming the receiver with more data than it can handle.
- A sliding window is used to make transmission more efficient as well as to control the flow of data so that the destination does not become overwhelmed with data.
- TCP sliding windows are byte-oriented & is of variable size.
- *Receive Window (rwnd)*, is the spare room in the receive buffer.
- *Congestion Window* (cwnd) is a TCP state variable that limits the amount of data the TCP can send into the network before receiving an ACK.
- With every ACK message the receiver advertises its current receive window.

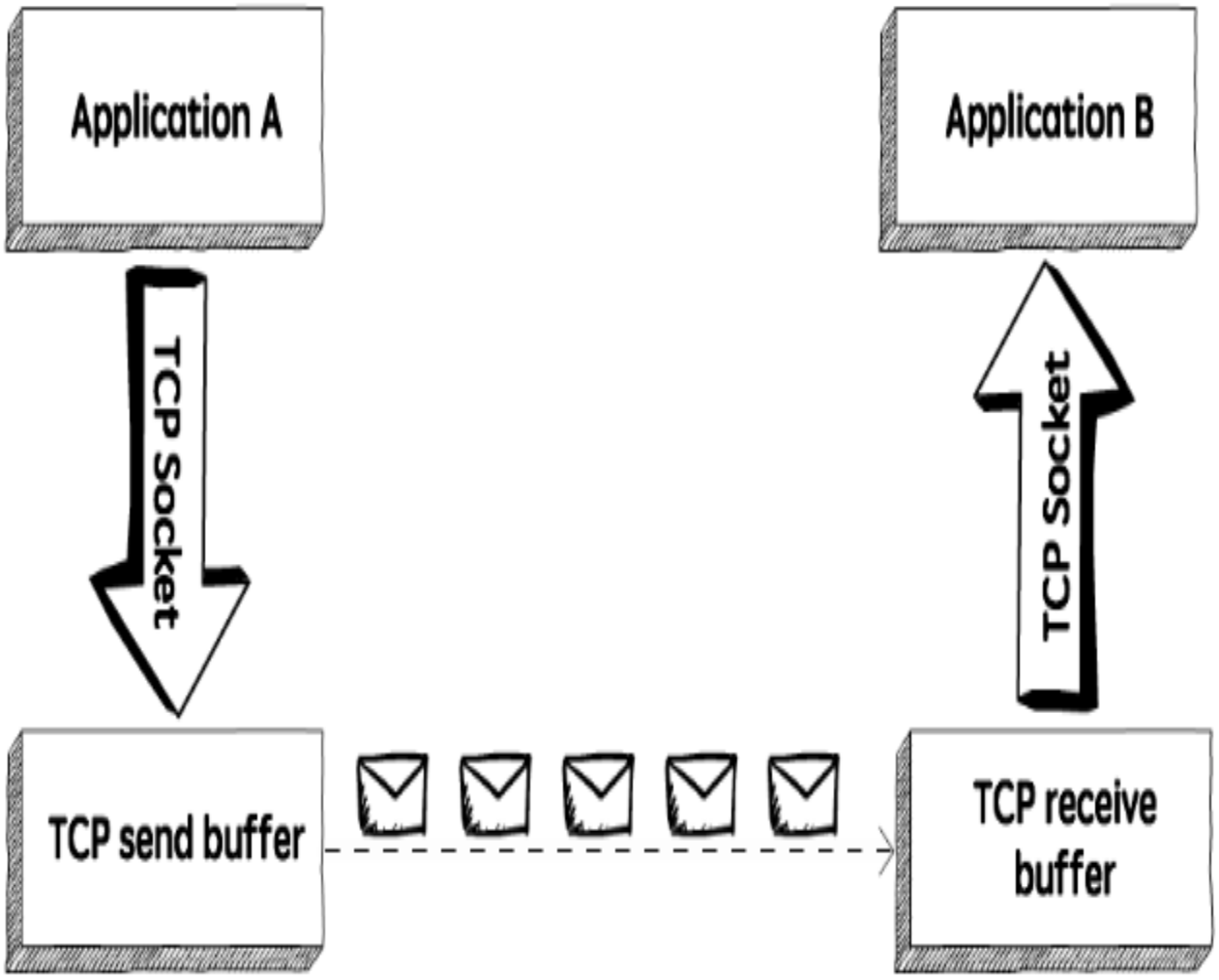
Flow Control in TCP

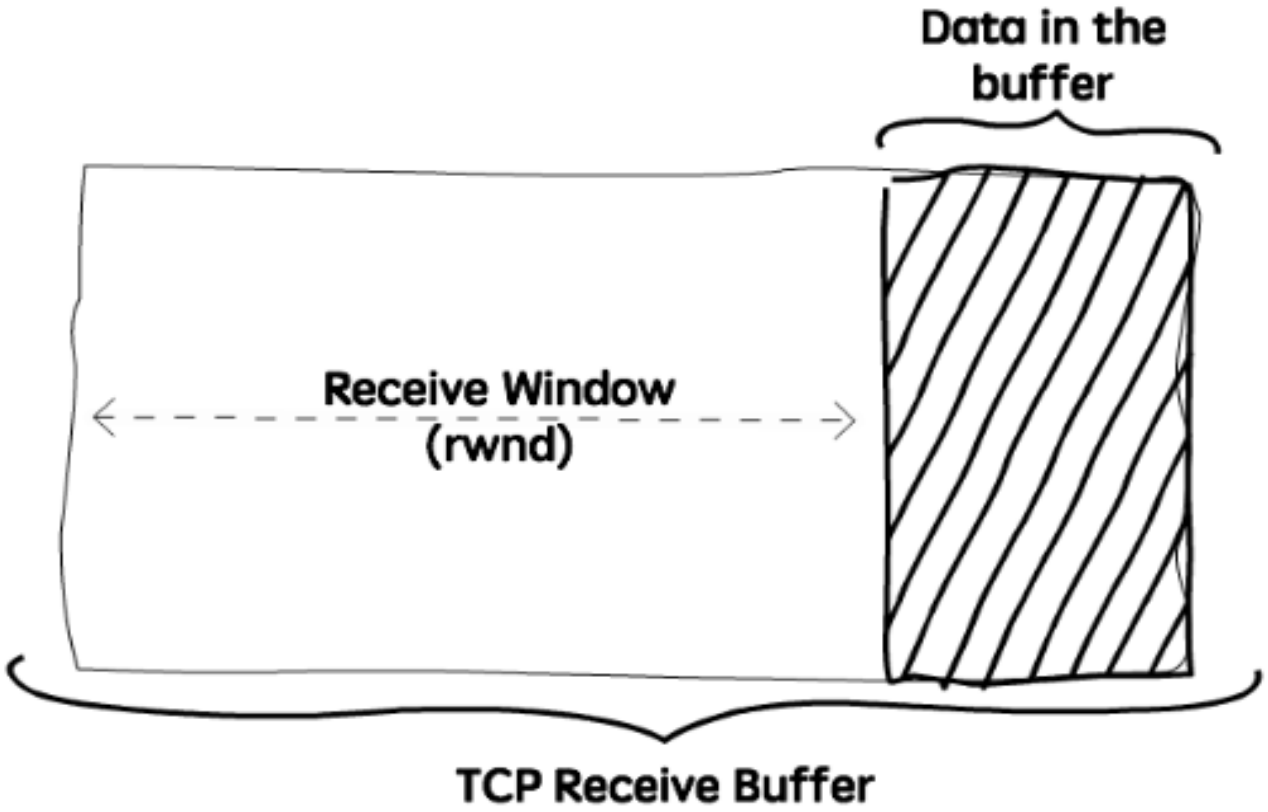
- The size of the window at one end is determined by the lesser of two values: *receiver window (rwnd)* or *congestion window (cwnd)*.

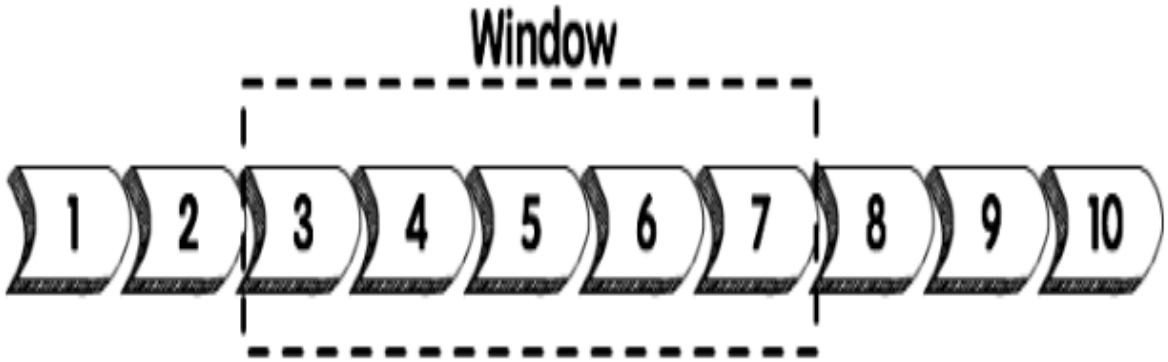
  window size= min(rwnd, cwnd)

- Sliding window in TCP spans a portion of the buffer containing bytes received from the process.
- Bytes inside the window are the bytes that can be in transit & can be sent without worrying about acknowledgment.
- The imaginary window has two walls: left and right.
- Window activities: *opened*, *closed*, or *shrunk*.

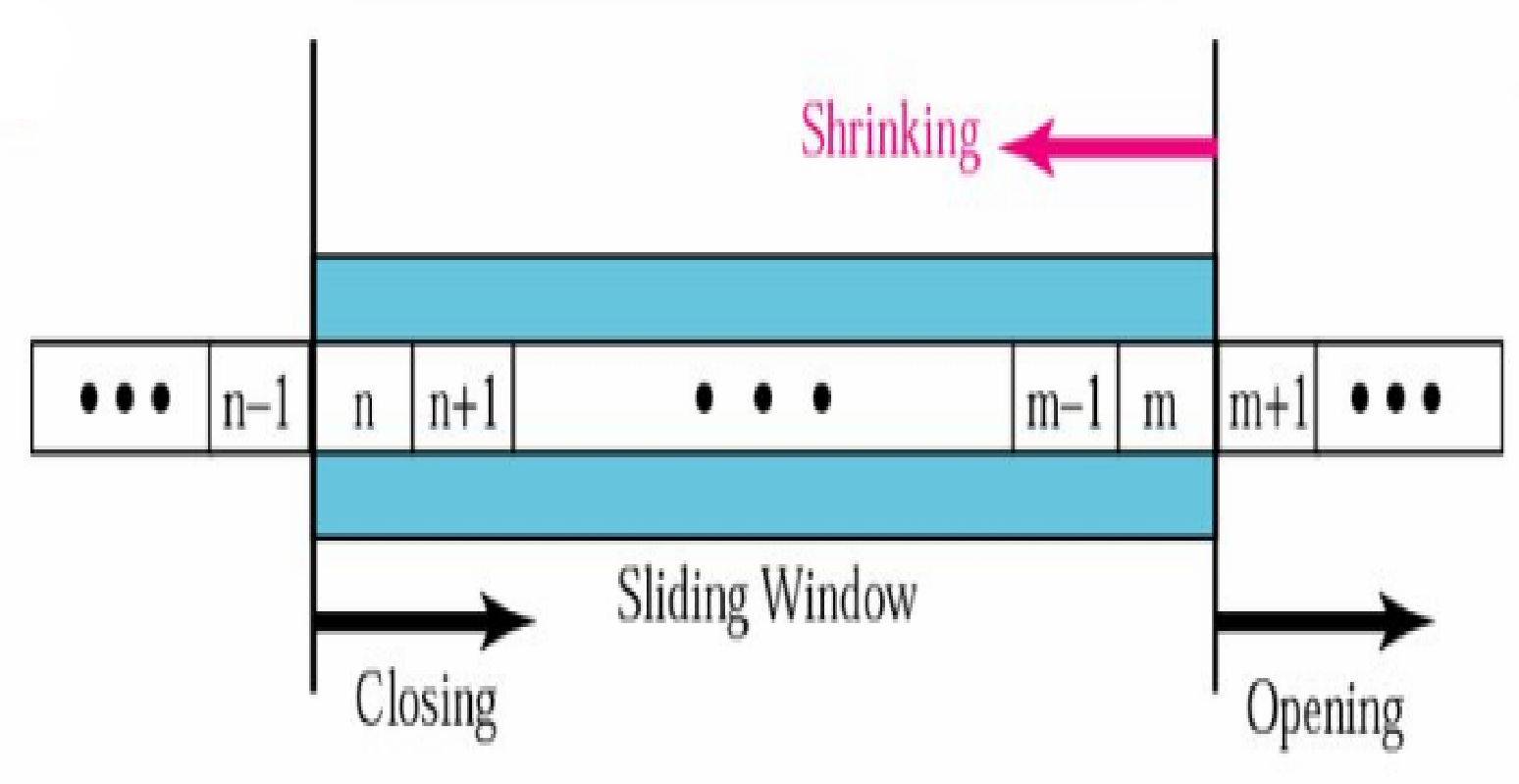

Application A
TCP
Network
Link
Application B
TCP
Network
Link

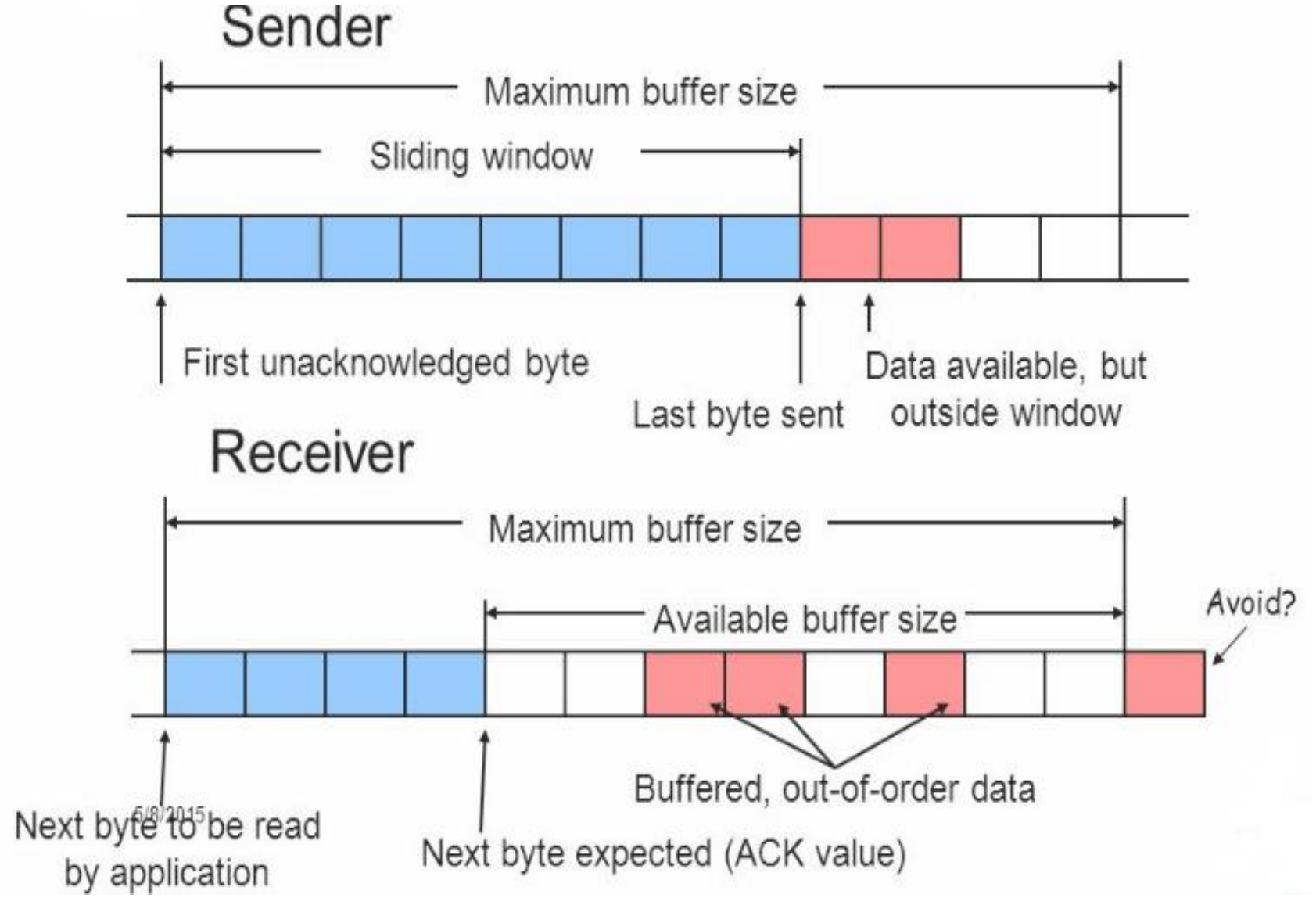

Application A
Application B
TCP Socket
TCP Socket
TCP send buffer
TCP receive buffer


Data in the buffer
Receive Window (rwnd)
TCP Receive Buffer



Example of a sliding window. As soon as packet 3 is acked, we can slide the window to the right and send the packet 8.

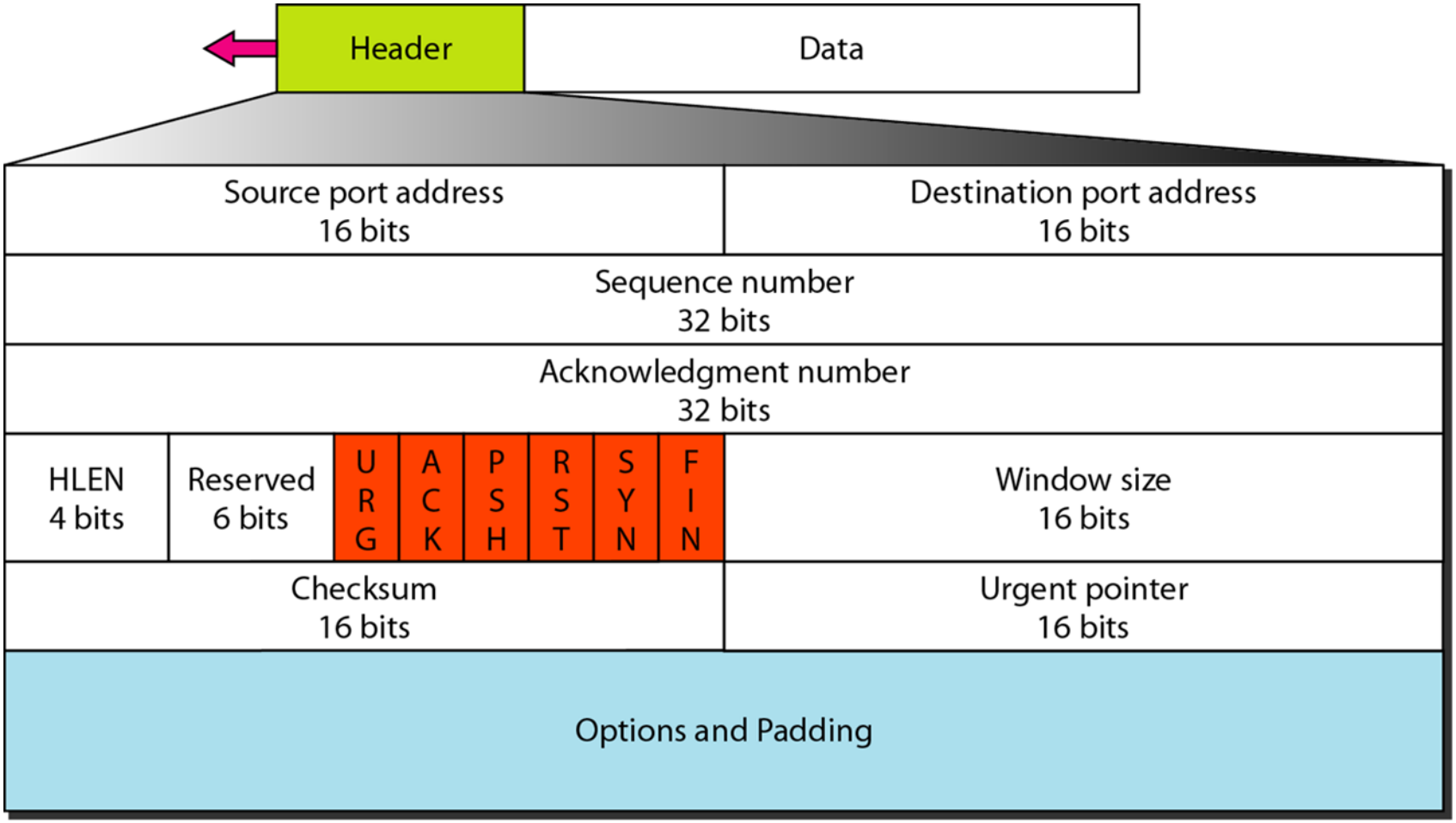

Shrinking
n-1
n
n+1
m-1
m
m+1
Sliding Window
Closing
Opening


Sender
Maximum buffer size
Sliding window
First unacknowledged byte
Last byte sent
Data available, but outside window
Receiver
Maximum buffer size
Available buffer size
Avoid?
Buffered, out-of-order data
Next byte to be read by application
Next byte expected (ACK value)

Error Control in TCP

- TCP is a reliable transport layer protocol
- TCP provides reliability using error control
- Mechanisms:
  - ✓ Checksum
  - ✓ Acknowledgment
  - ✓ Time-out
- Detects corrupt segments, lost segments, out-of-order segments, and duplicated segments. Also corrects errors after detection

TCP segment format



Checksum

- TCP uses a 16-bit checksum that is mandatory in every segment.
- Checksum field in each segment required to check for a corrupted segment
- If segment is corrupted, it is discarded by the destination TCP and is considered as lost.
- TCP checksums are calculated over the entire segment, both the header and the data.
- The ENTIRE segment is divided into 16-bit pieces and add up all of them.

Acknowledgment

- TCP uses acknowledgments to confirm the receipt of data segments.
- Control segments that carry no data but consume a sequence number are also acknowledged.
- ACK segments are never acknowledged.

NOTES

- The bytes of data being transferred in each connection are numbered by TCP.
- The numbering starts with a randomly generated number.
- The value in the sequence number field of a segment defines the number of the first data byte contained in that segment.
- 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.

Sequence Numbers

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)**

- http://www.tcpipguide.com/free/t_TCPOperationalOverviewandtheTCPFiniteStateMachineF-2.htm
- https://www.brianstorti.com/tcp-flow-control/