Transport Layer (TCP)

Lecture 7 | CSE421 – Computer Networks

Department of Computer Science and Engineering
School of Data & Science

TCP Services

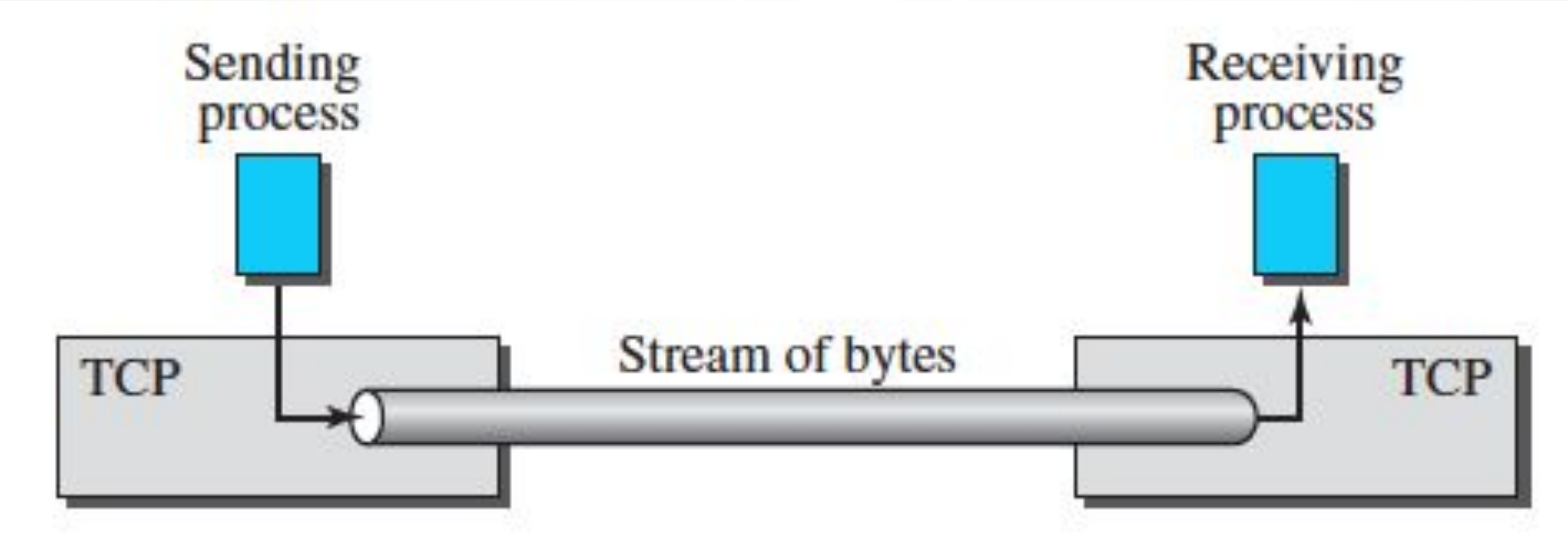
- Stream delivery
- Segmenting and Reassembling segments
- Multiplexing segments
- Full Duplex Service
- Identifying and tracking the segments of different applications
- Connection Oriented Service
- Reliable service

Objectives (Part 1)

- Stream delivery
- Segmenting and Reassembling segments
- Multiplexing segments
- Full Duplex Service
- Identifying and tracking the segments of different applications.

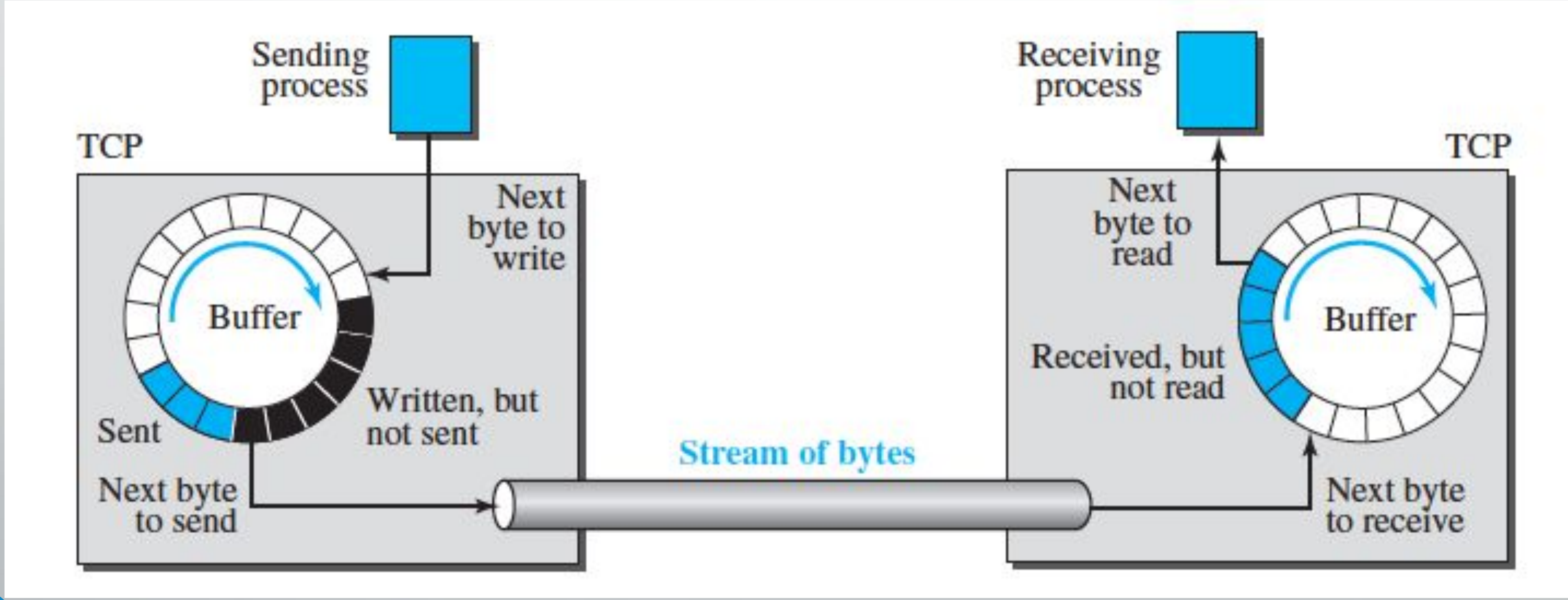
Stream Delivery

- TCP, unlike UDP, is a **stream-oriented** protocol
- TCP allows the sending process to deliver data as a stream of bytes
- And allows the receiving process to obtain data as a stream of bytes



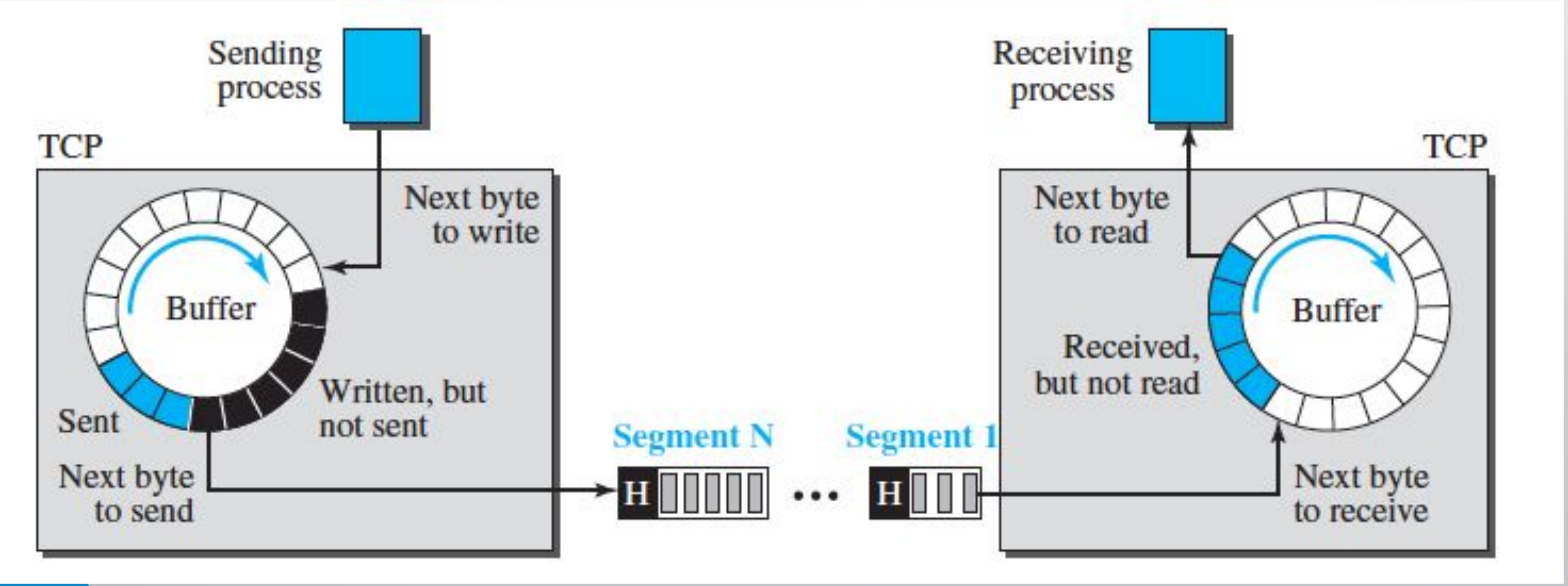
Buffers

- The sending and the receiving processes may not necessarily write or read data at the same rate
- So TCP will need to store the data in a place before it can process it.
- That storage space is known as **buffer**

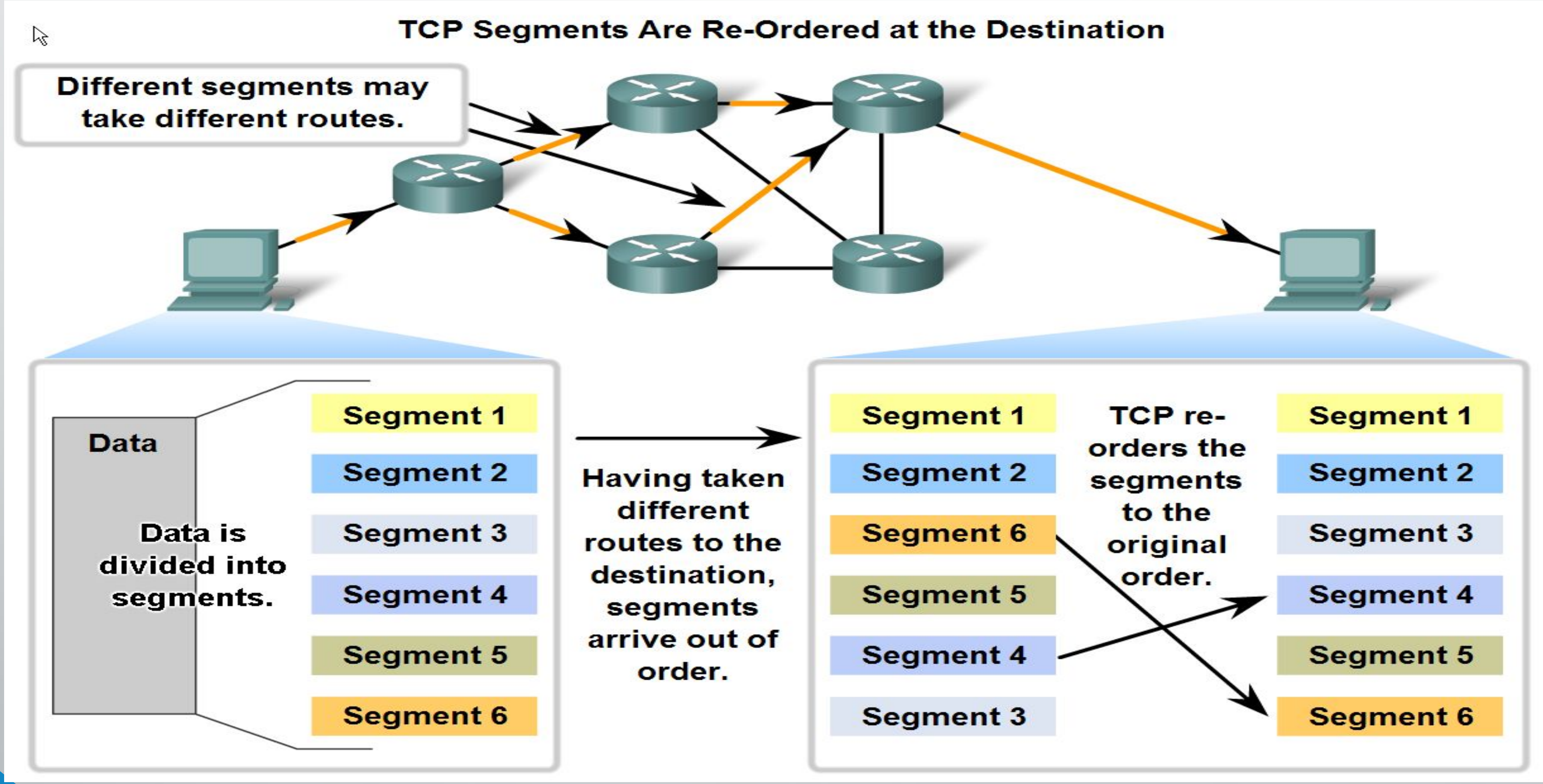


Segments

- The network layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes.
- So TCP groups a number of bytes together into a packet called a **segment .**

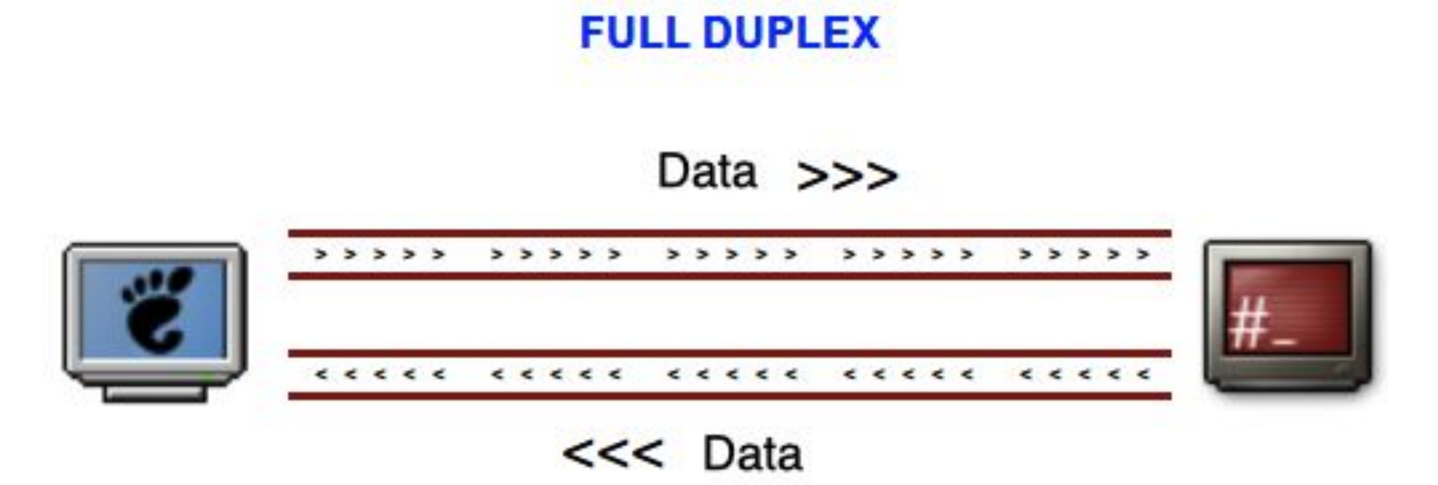


TCP Segment Reassembly



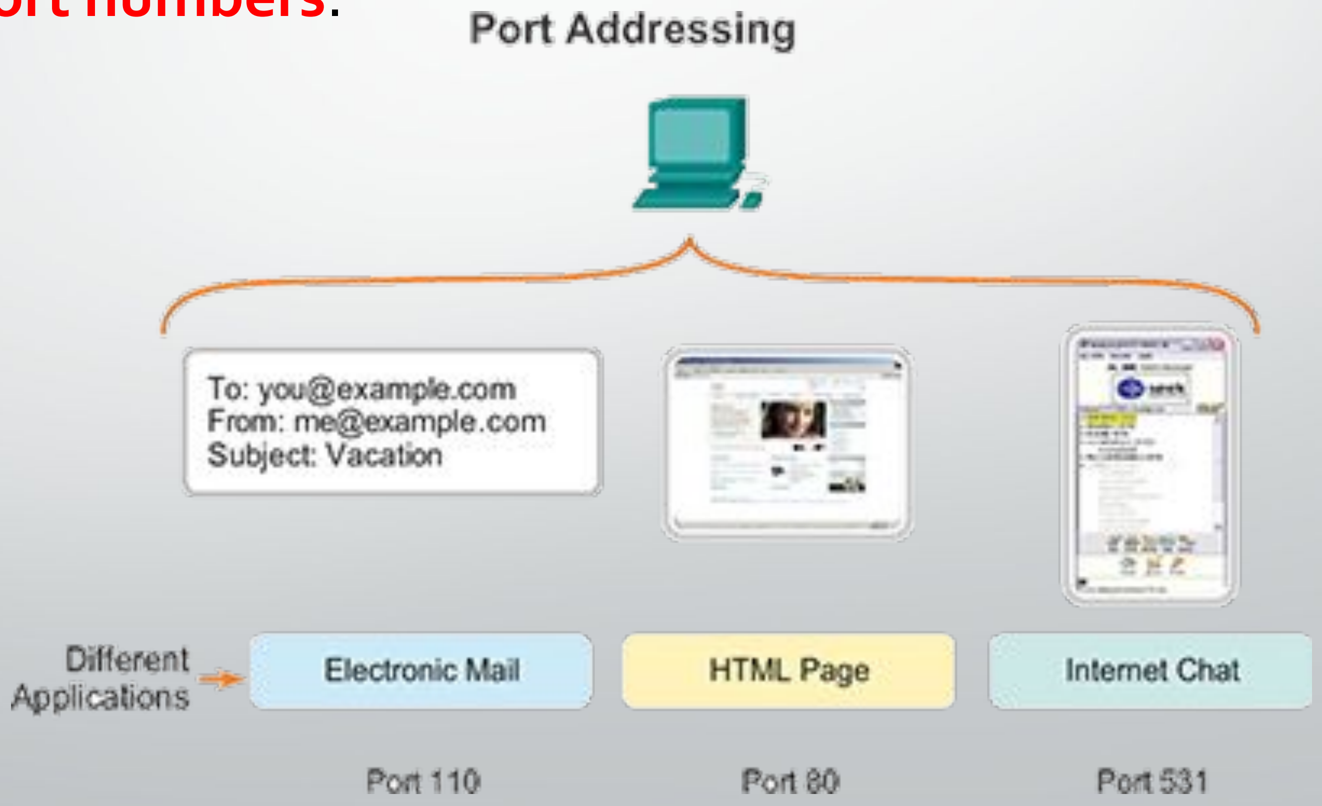
Full Duplex

- Each TCP connection supports a pair of byte streams, one flowing in each direction.
- Exchanging data (sending and receiving) between two entities at the same time.



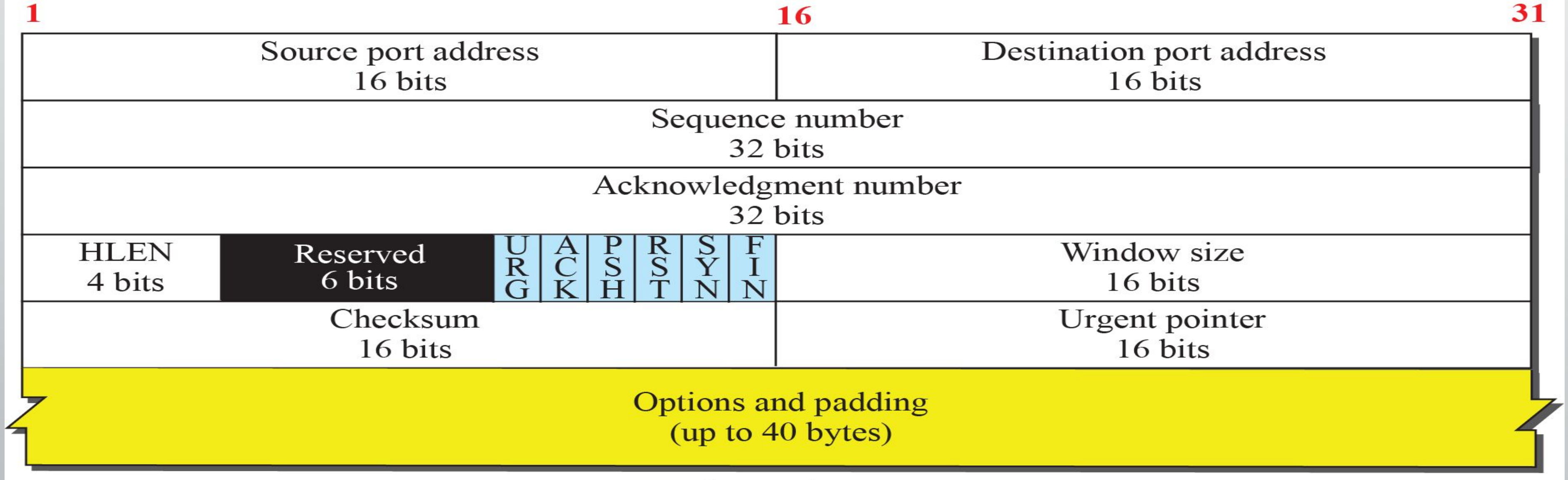
Identifying and tracking the segments

- TCP and UDP-based services must keep track of the various applications communicating.
- To differentiate the segments and datagrams for each application, both TCP and UDP uses **port numbers**.



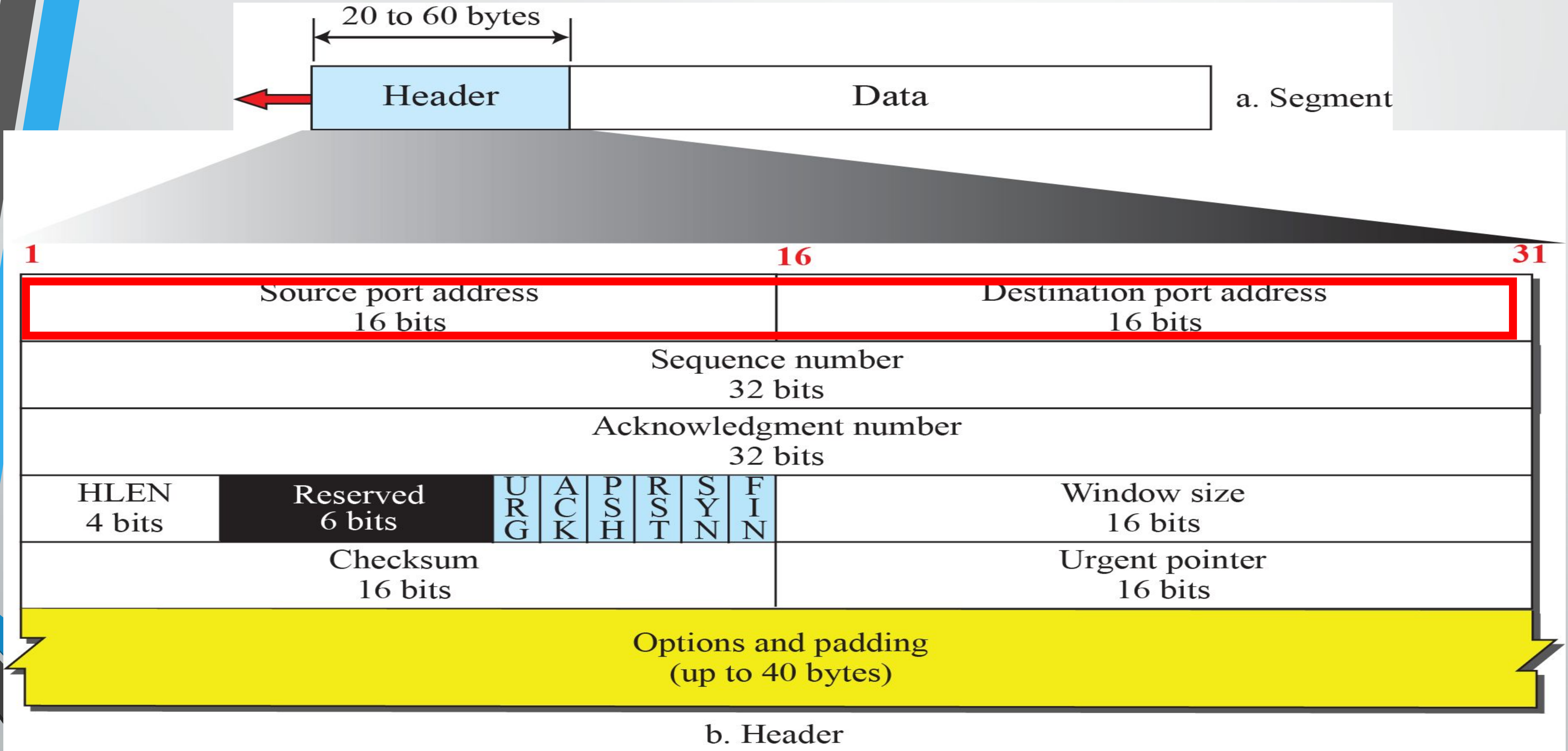
Objectives (Part 2)

- TCP Segment Header
  - All header fields



b. Header

TCP Segment Header



Byte Number

- The bytes of data being transferred in each connection are numbered by TCP.
- The numbering starts with an arbitrarily generated number.
- An arbitrary number between **0 and $2^{32} - 1$** for the number of the first byte.

- For example if the number of the first byte happens to be **1067** and the total data to be sent is **3000 bytes** .
- What is the byte number for the first byte of data and last byte of data?

First Byte Number	→	Last Byte Number
1067		4066

Sequence Numbers

- The sequence number of the first segment is **the ISN (initial sequence number)**, which is a random number (byte number).

- The sequence number of any other segment is the sequence number of the previous segment plus the number of bytes **(real or imaginary)** carried by the previous segment.

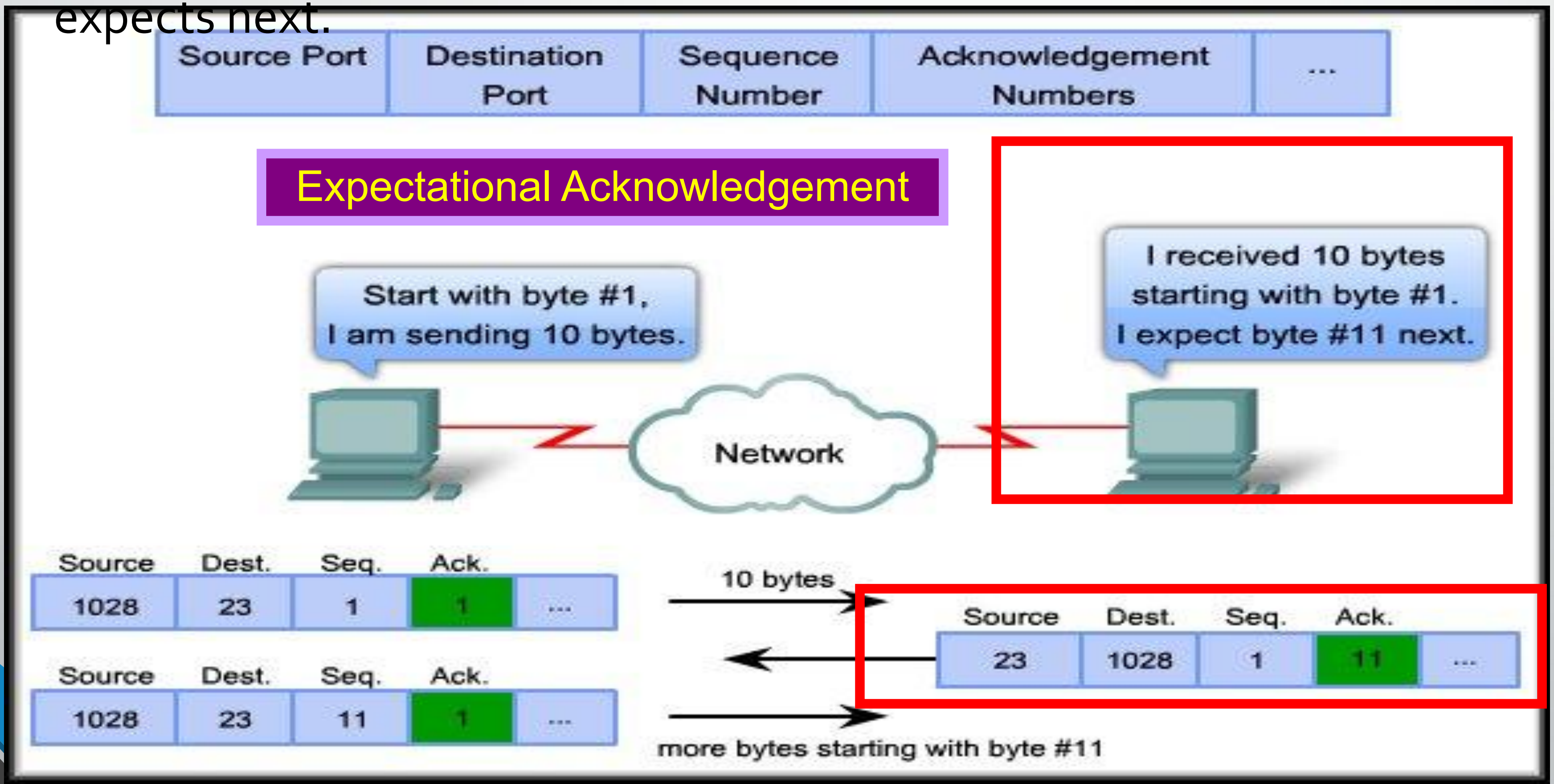
- Suppose a TCP connection is transferring a file of 5,000 bytes. The first byte is numbered 10,001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1,000 bytes?

- **Solution:**

Segment 1	→	Sequence Number:	10001	**Range:**	10001	to	11000
Segment 2	→	Sequence Number:	11001	**Range:**	11001	to	12000
Segment 3	→	Sequence Number:	12001	**Range:**	12001	to	13000
Segment 4	→	Sequence Number:	13001	**Range:**	13001	to	14000
Segment 5	→	Sequence Number:	14001	**Range:**	14001	to	15000

Acknowledgement Number

- If receiving host TCP receives uncorrupted data, then...
- It sends an acknowledgement by giving the sequence number of the byte that it expects next.



Acknowledgement Number

- The value of the acknowledgment field in a segment defines the **number of the next byte** the receiver expects to receive.
- The acknowledgment number is **cumulative**.

- For example if the sender receives **1001** as the acknowledgement number.
- What does it mean?

Received all data up to 1000, tells the sender that it ready to receive the next data from 1001 byte number.

Note :This does not indicate receiver has received 1000 bytes of data. Why?

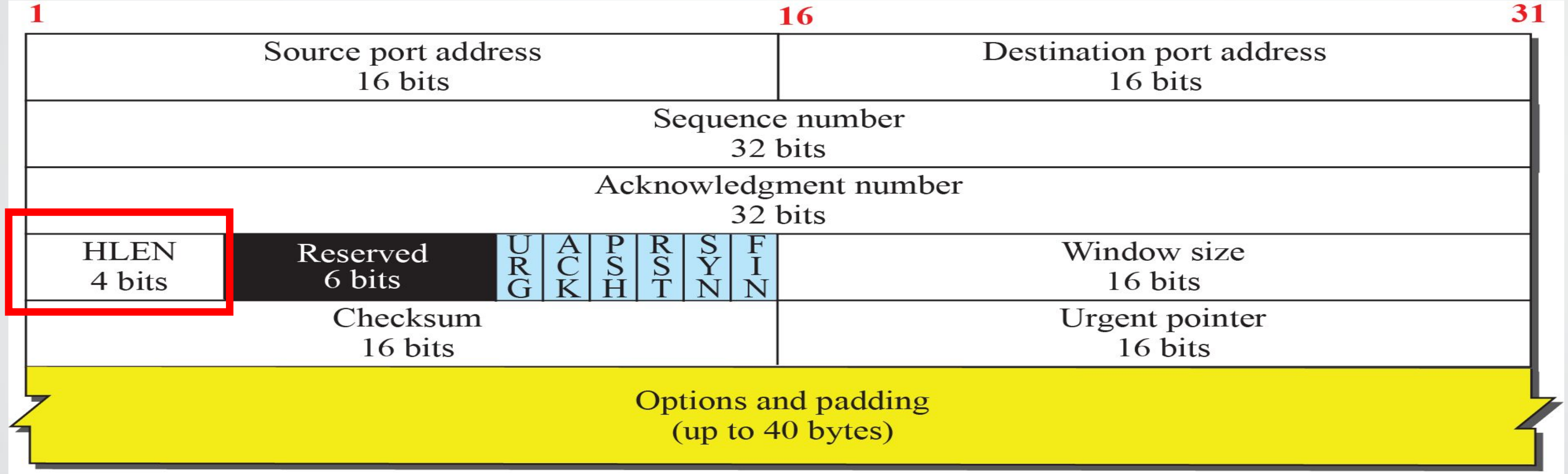
Acknowledgement Number

- The acknowledgment number is **cumulative**.
- Receiver acknowledges multiple data segments in one acknowledgement.

Sender

Receiver

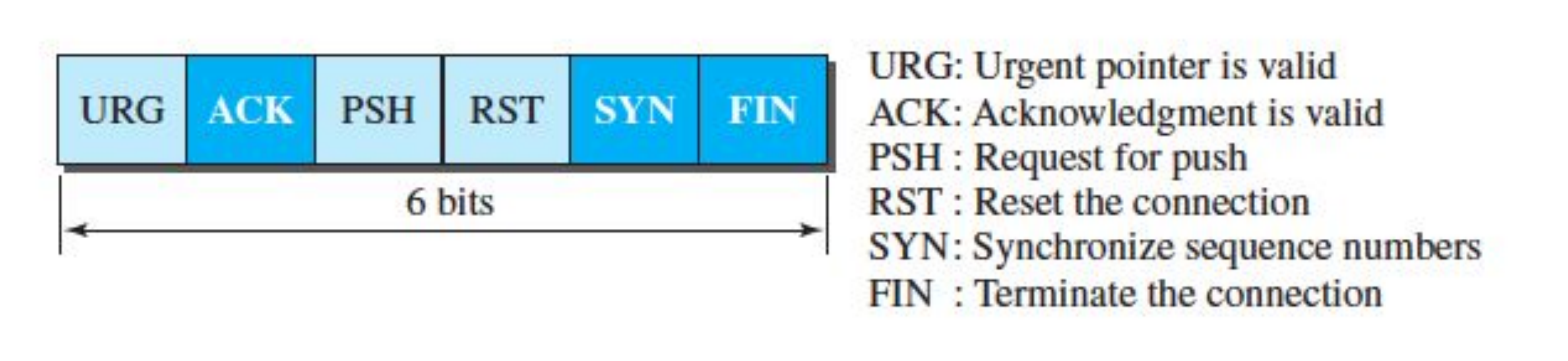
Header Length



b. Header

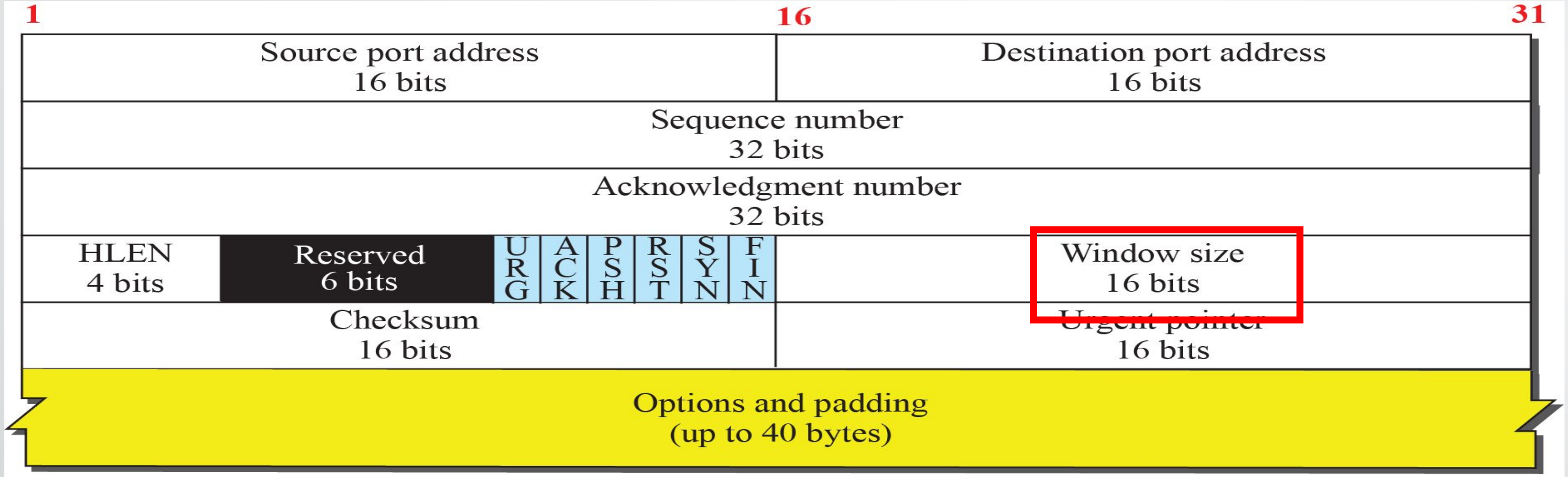
- Header Length :
  - Indicates the number of 4-byte words
  - The length of the header can be between 20 and 60 bytes

Control Bits



- Control Bits:
  - This field defines 6 different control bits or flags
  - One or more of these bits can be set at a time
  - These bits help indicate connection establishment and termination, flow control

Window Size

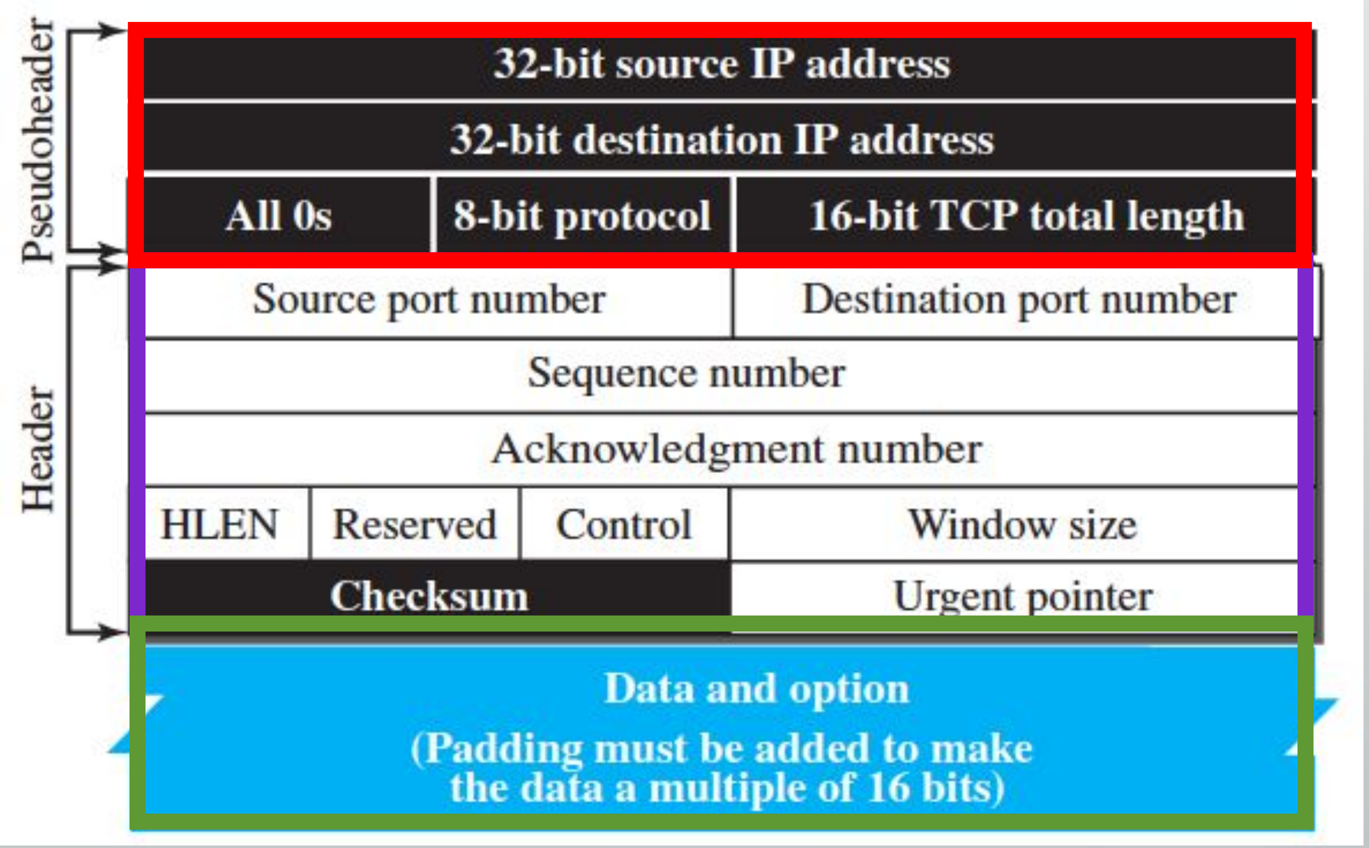


b. Header

- Window Size:
  - This field defines size of data in bytes of the sending TCP process
  - The maximum size of the window is 65,535 bytes
  - Normally referred to as the receiving window (rwnd)
  - The sender must obey the dictation of the receiver in this case

Checksum

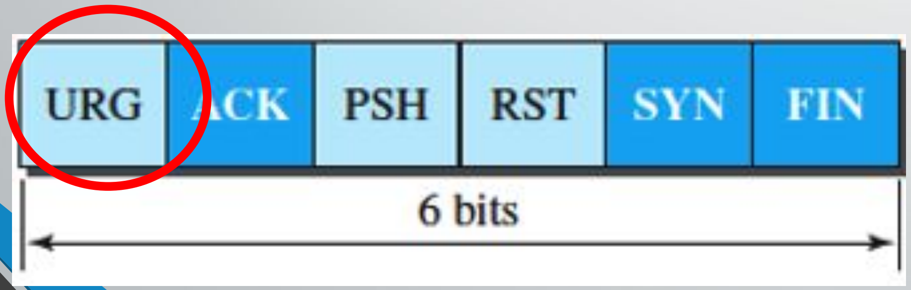
- This 16 bits field is used to check if the segment got corrupted (intentionally or unintentionally) while segment was on traveling in order to reach the destination.
- Mandatory in TCP

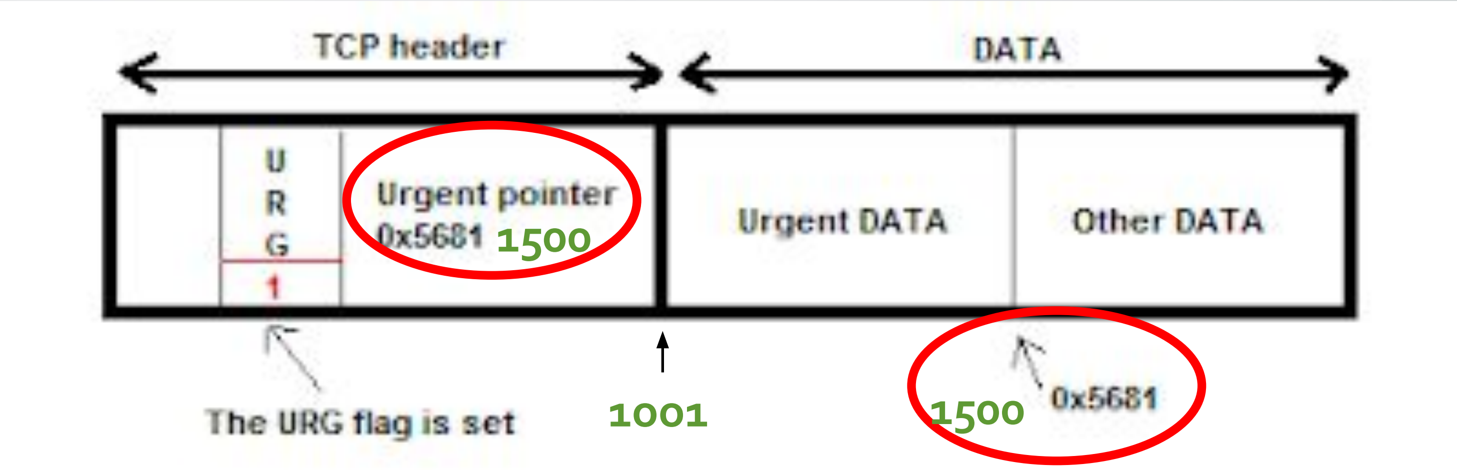


- **TCP Header**
- **TCP Body**
- **Pseudo IP Header**

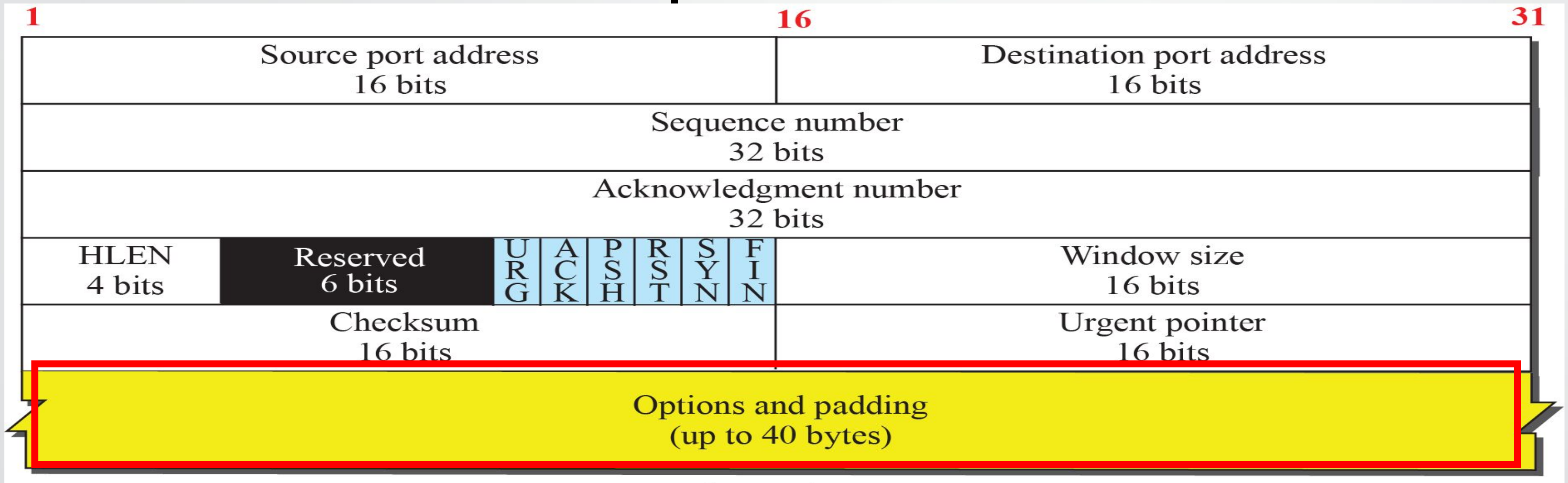
Urgent Pointer

- This 16-bit field, which is valid only if the urgent flag is set.
- Used when the segment contains urgent data.
- It defines a value that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.





Options



b. Header

- There can be up to 40 bytes of optional information in the TCP header
- Provides a way to deal with limitations of the original header
- For example :
  - MSS (Maximum Segment Size) is defined as the largest block of data that a sender using TCP will send to the receiver

Objectives (Part 3)

- TCP Services
  - Connection Establishment
  - Pushing Data
  - Connection Termination

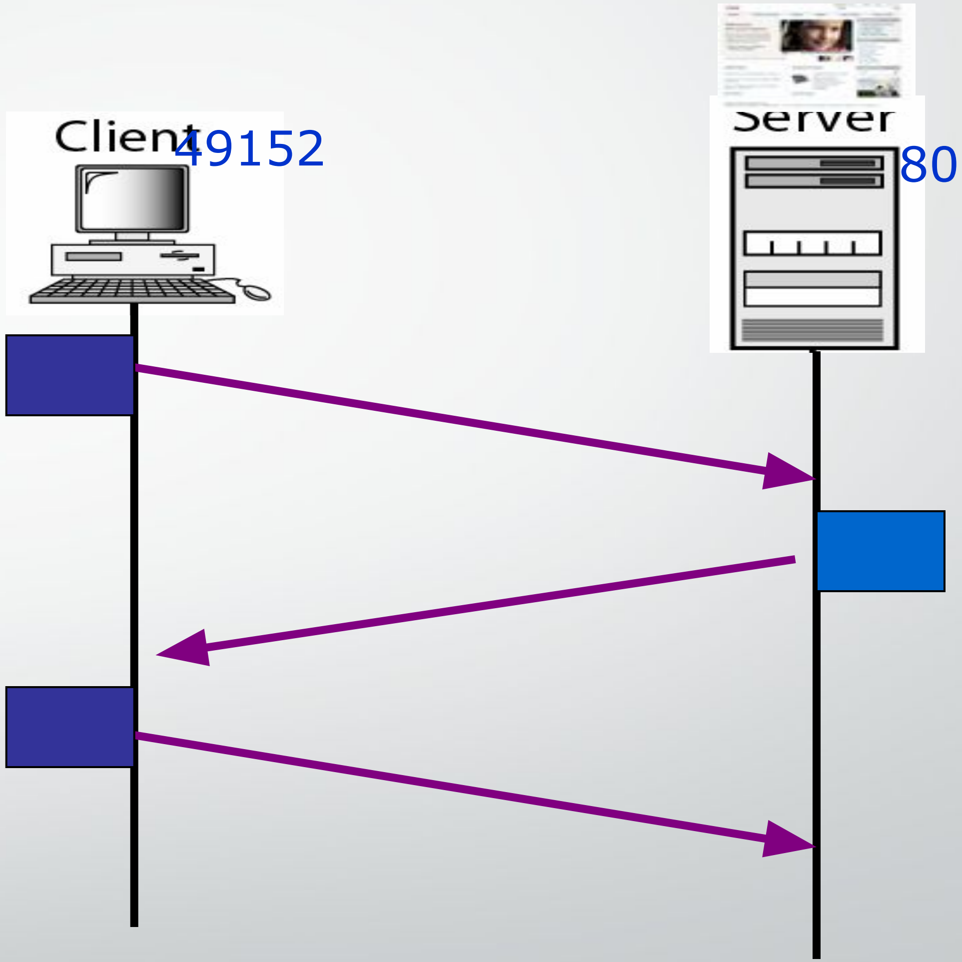
Connection Establishment

- TCP sets up a connection between end hosts before sending data
- This process is known as **"Three-way handshake"**
- After the connection is established the hosts can send data

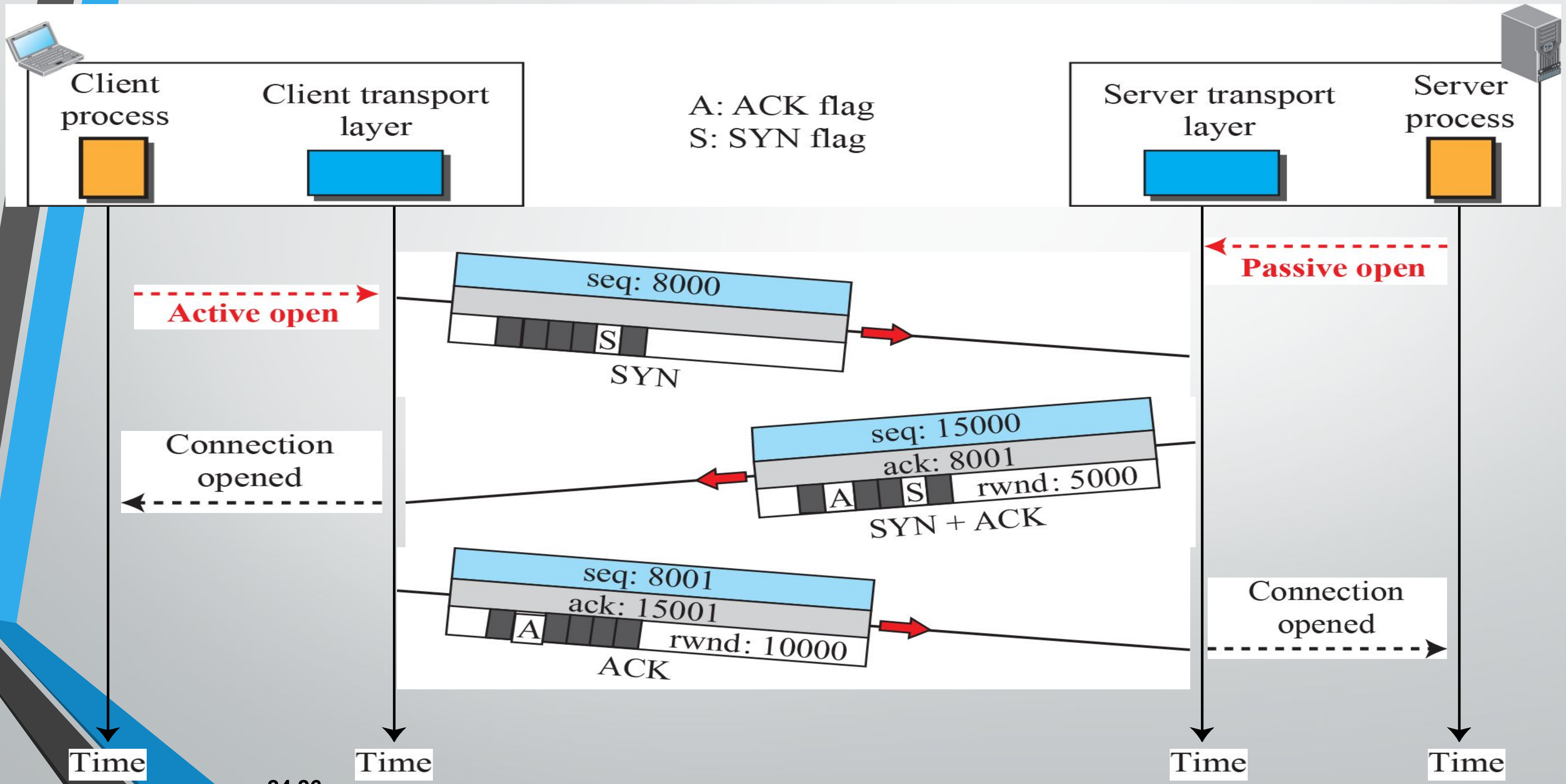
3 Way Handshake : Connection Establishment

Source Port No.						Destination Port No.	
Sequence No.							
Acknowledgement No.							
	U	A	P	R	S	F	Window Size
Others							

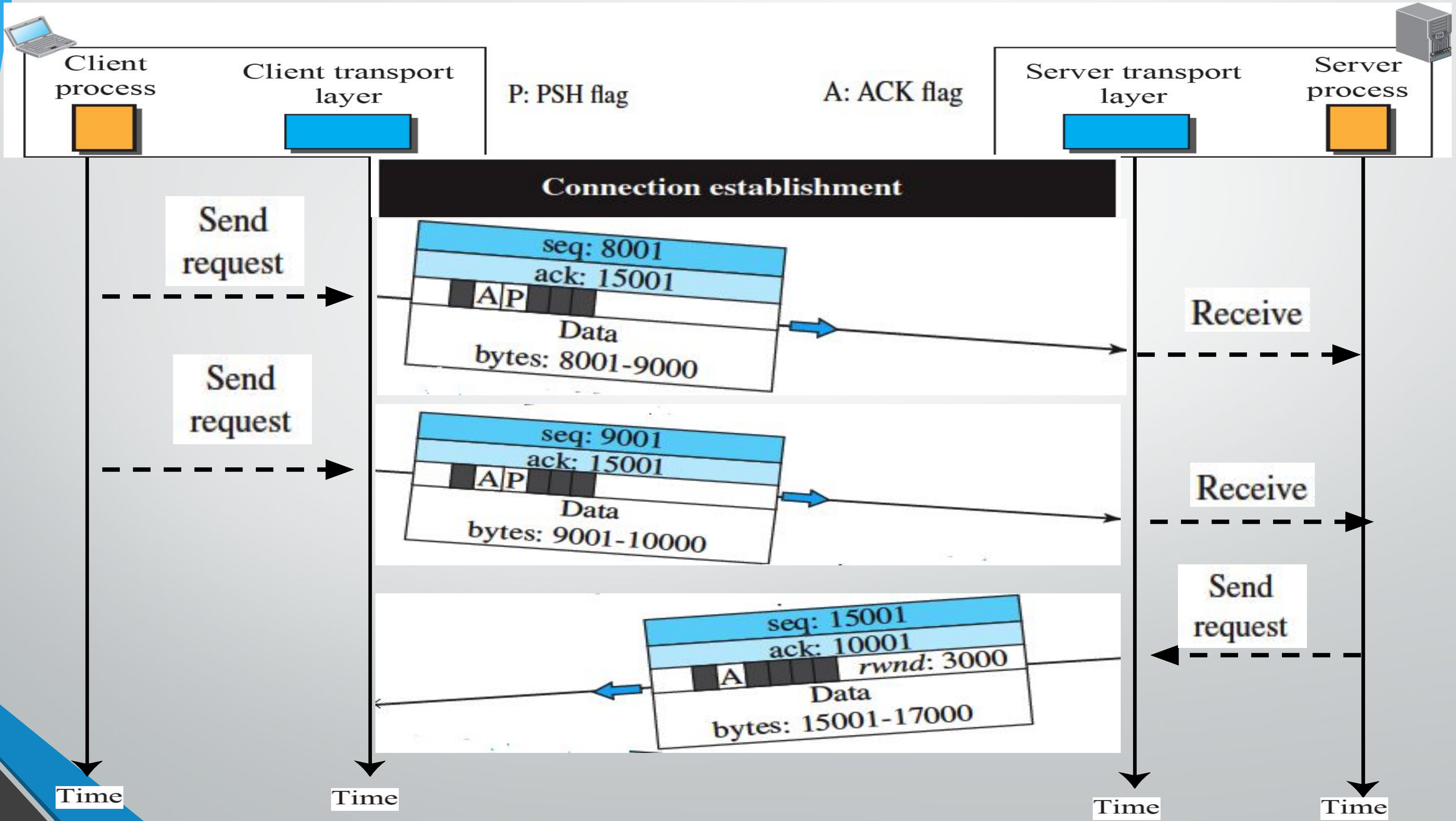
49152 80						80 49152 80	
1000 101							
0 1011							
		A			S		Window Size
Others							



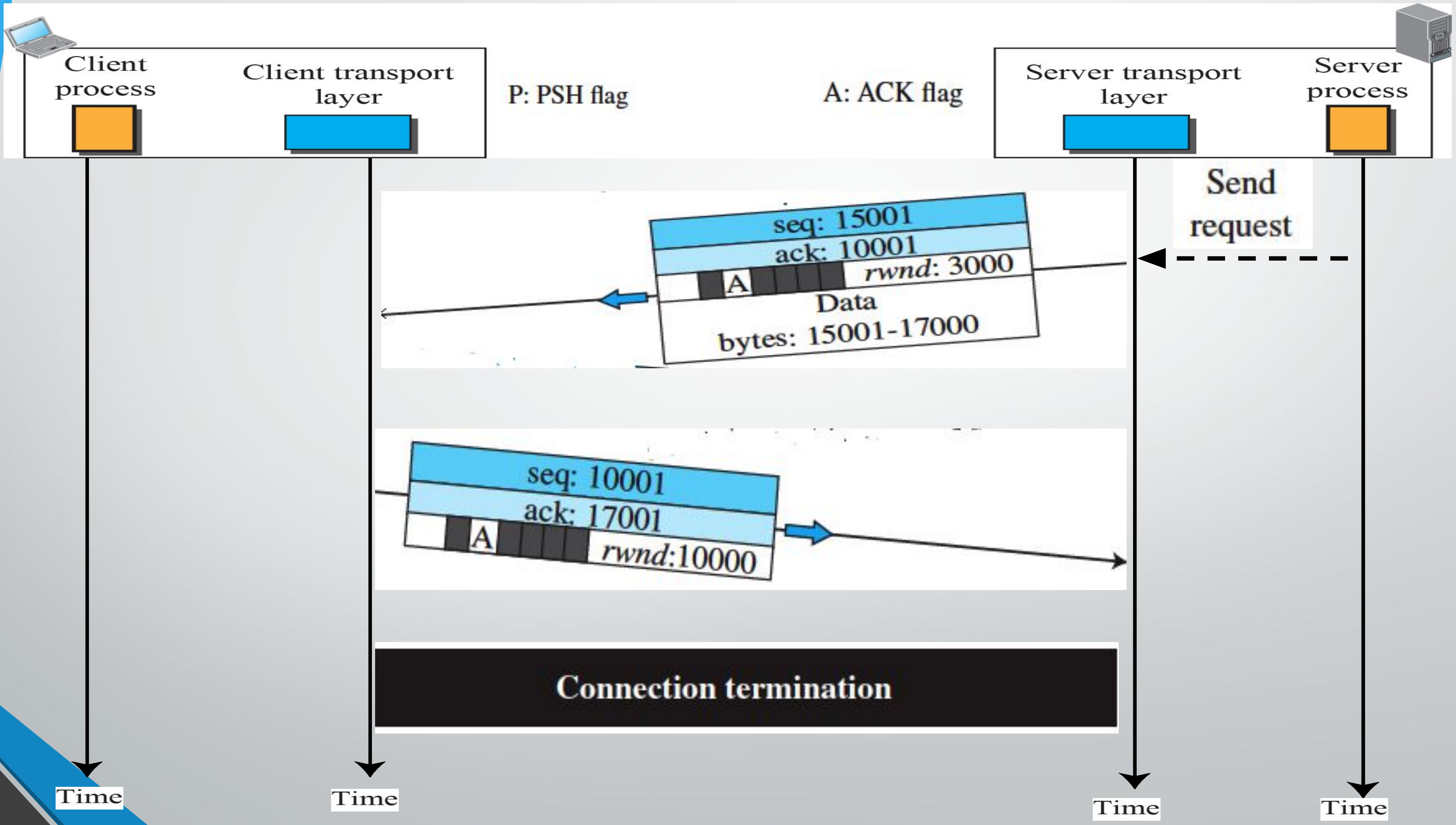
3 Way Handshake : Connection Establishment



Data Transfer

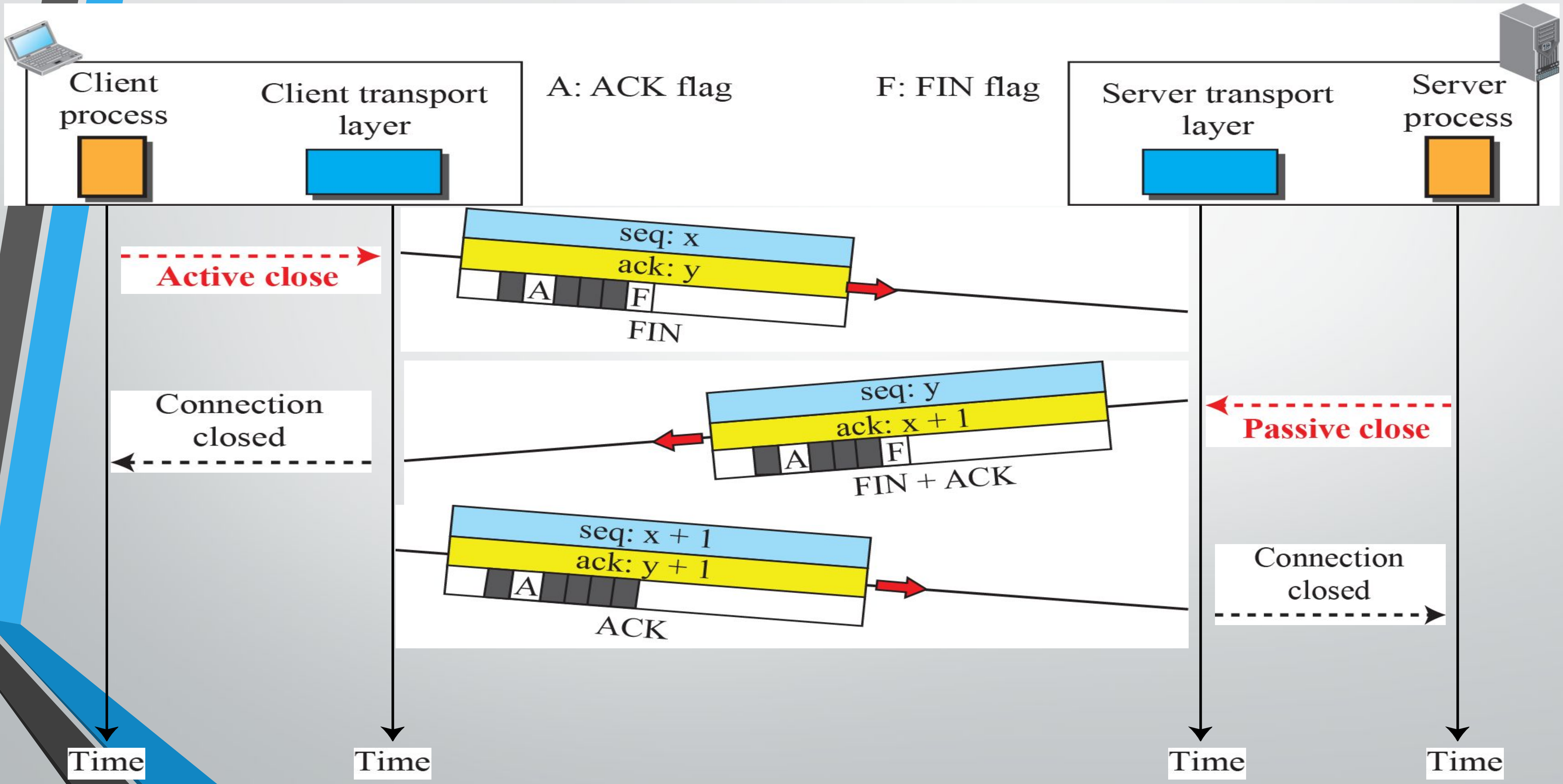


Data Transfer Continued..

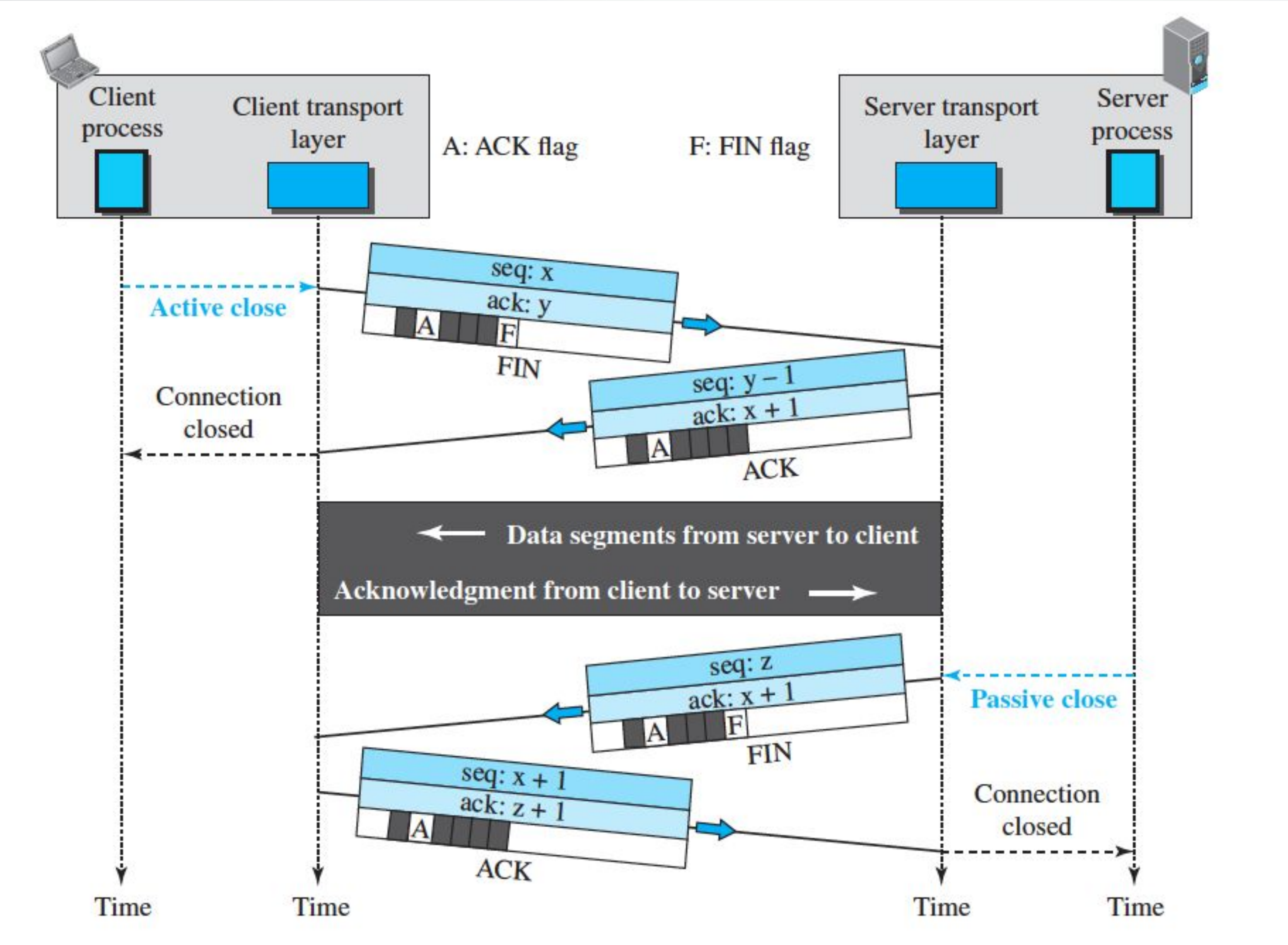


Connection Termination

3 Way Handshake : Connection Termination



Connection Termination :: Half Close



Objectives (Part 4)

- Reliable service
  - Error Control

Reliability in TCP

- TCP provides **reliability** using **error control**
- Error control includes mechanisms for
  - detecting and resending corrupted segments
  - resending lost segments
  - storing out-of order segments until missing segments arrive
  - detecting and discarding duplicated segments.
- Error control in TCP is achieved through
  - Checksum
  - Acknowledgment
  - Time-out and retransmission

Error Control

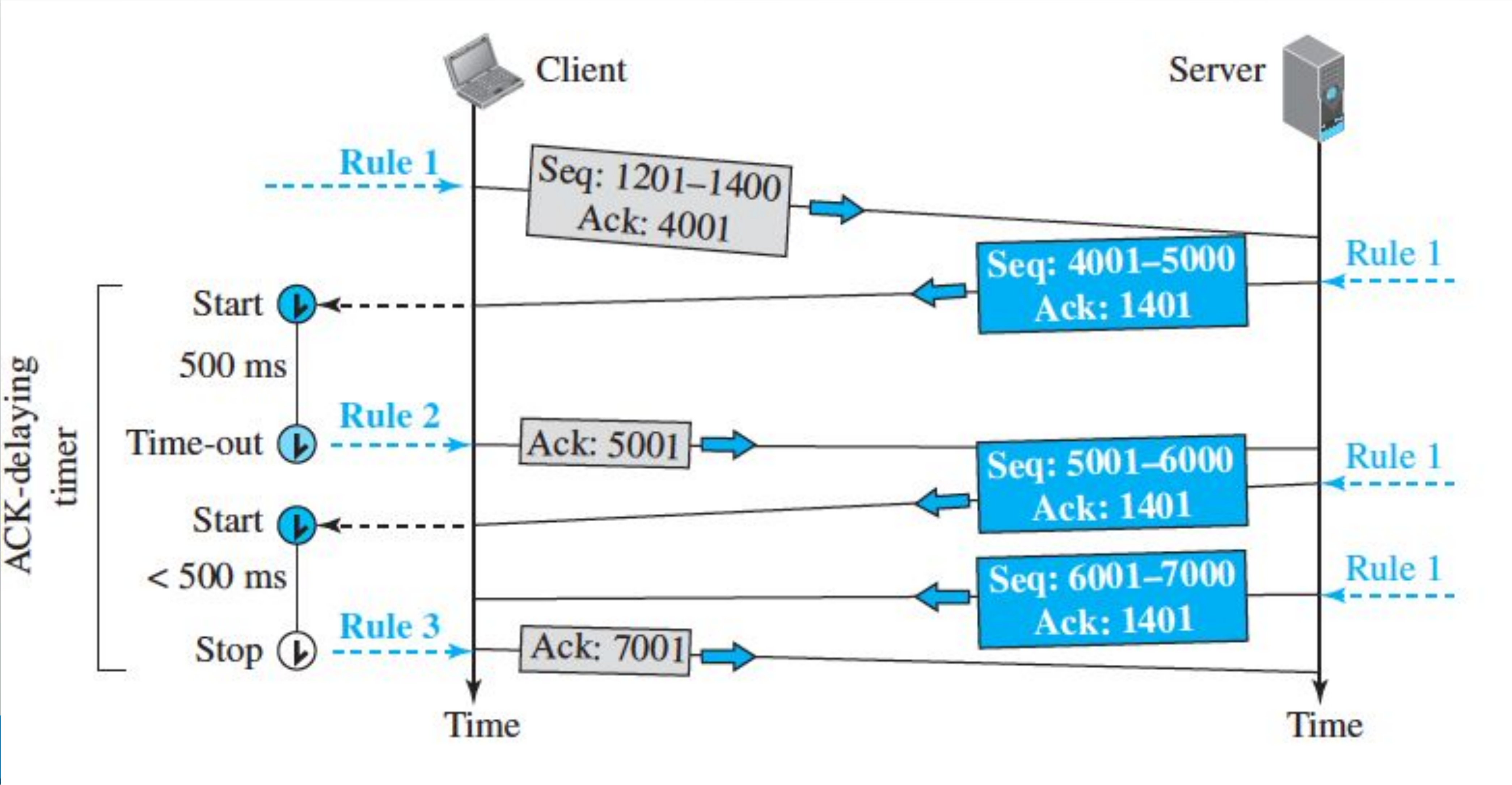
- **Checksum**
  - Each segment includes a checksum field, which is used to check for a corrupted segment
  - If a segment is corrupted, as detected by an invalid checksum, the segment is discarded
- **Acknowledgment**
  - To confirm the receipt of data segments.
  - To confirm control segments that carry no data, but consume a sequence number
- ACK segments do not consume sequence numbers and are not acknowledged.

Types and Rules of Acknowledgment

- Cumulative Acknowledgment (ACK)
- Selective Acknowledgment (SACK)

- **Generating Acknowledgments Rules (1 to 3)**
  - **Rule 1:** When host A sends a data segment to host B, it must include (piggyback) an acknowledgment number.
  - **Rule 2 :** The receiver needs to delay sending an ACK segment if there is only one outstanding in-order segment.
  - **Rule 3:** When a segment arrives with a sequence number that is expected by the receiver, and the previous in-order segment has not been acknowledged, the receiver immediately sends an ACK segment.

Normal Operation



Other Scenarios

- Segment Lost or Corrupted?
- Retransmission of segment ??
- How will the sender know ??
- What about the receiver, not aware of a packet sent?
- **Retransmission** after time out.

Retransmission

- The heart of the error control mechanism is the retransmission of segment
- When a segment is sent, it is stored in a queue until it is acknowledged.
- Retransmission of segment will occur
- **After RTO**
  - The sending TCP maintains one retransmission time-out (RTO) for each connection.
  - When the timer matures TCP resends the segment in the front of the queue
- **After Three Duplicate ACK Segments**
  - If three duplicate acknowledgments (i.e., an original ACK plus three exactly identical copies) arrive for a segment, the next segment is retransmitted without waiting for the time-out.

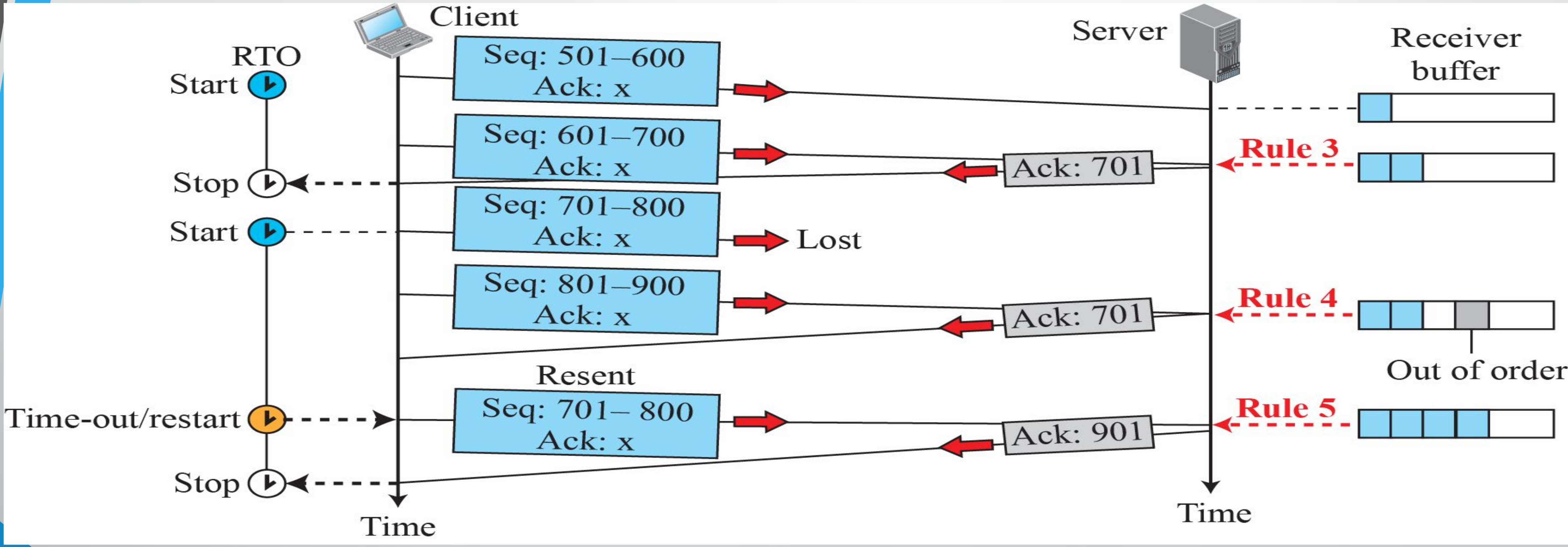
Out of Order Segments

- TCP implementations today do not discard out-of-order segments.
- They store them temporarily .
- Flag them as out-of-order segments until the missing segments arrive.
- Out-of-order segments are never delivered to the process.
- TCP guarantees that data are delivered to the process in order.

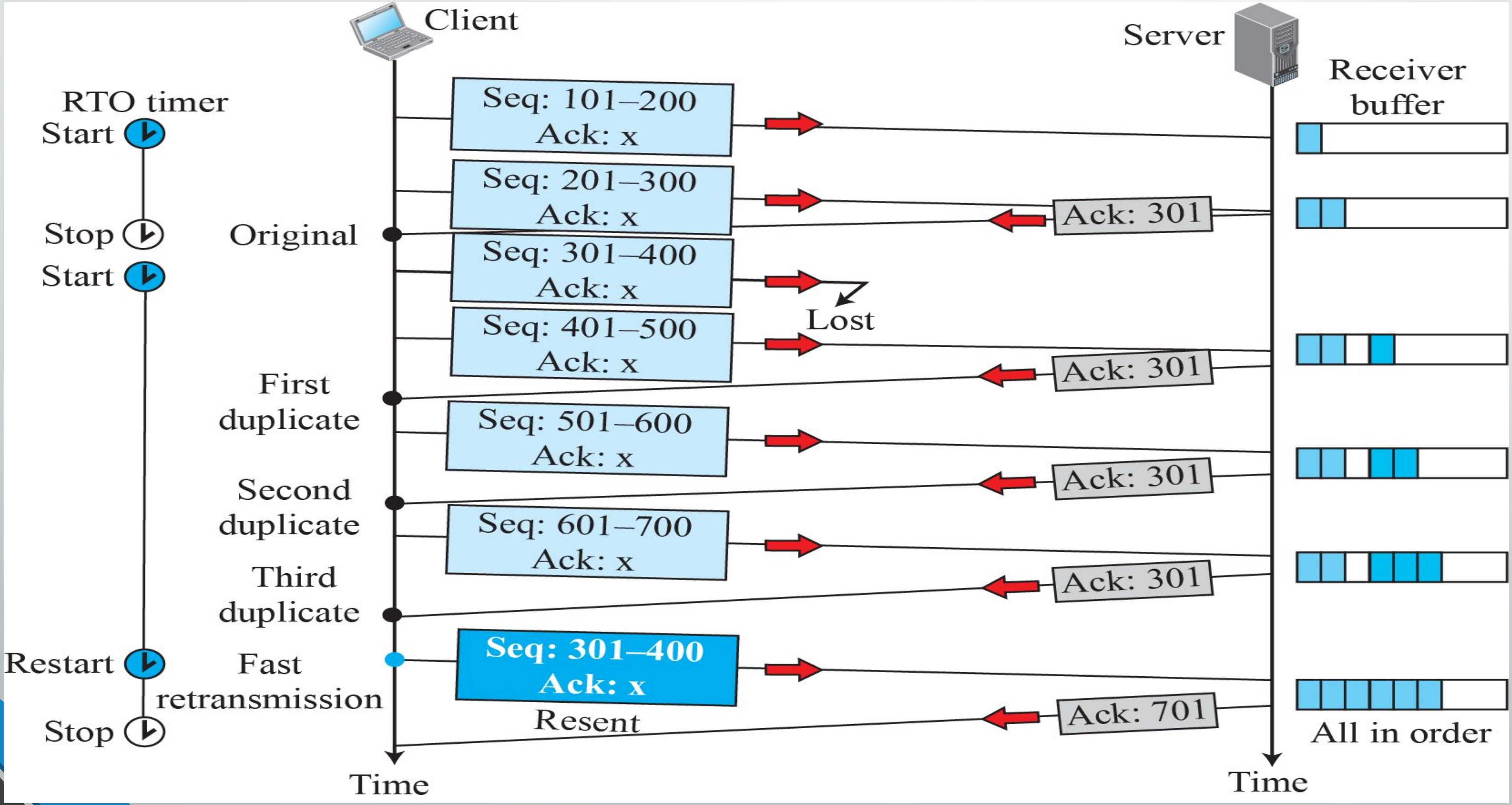
Rules of Acknowledgment contd

- **Generating Acknowledgments Rules (4 to 6)**
- **Rule 4:** When a segment arrives with an out-of-order higher sequence number the receiver immediately sends an ACK segment .
- **Rule 5 :** When a missing segment arrives, the receiver sends an ACK segment to announce the next sequence number expected.
- **Rule 6:** If a duplicate segment arrives, the receiver discards the segment, but immediately sends an acknowledgment indicating the next in-order segment expected.

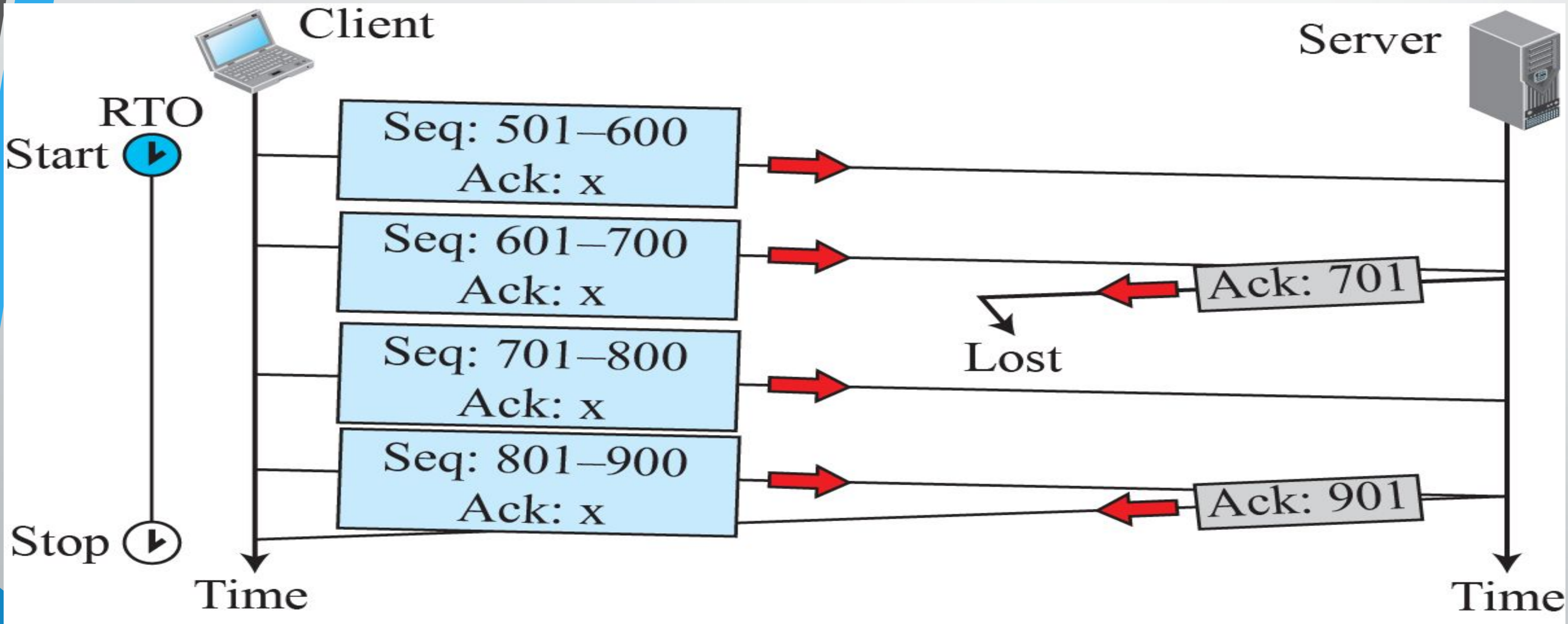
Lost Segment



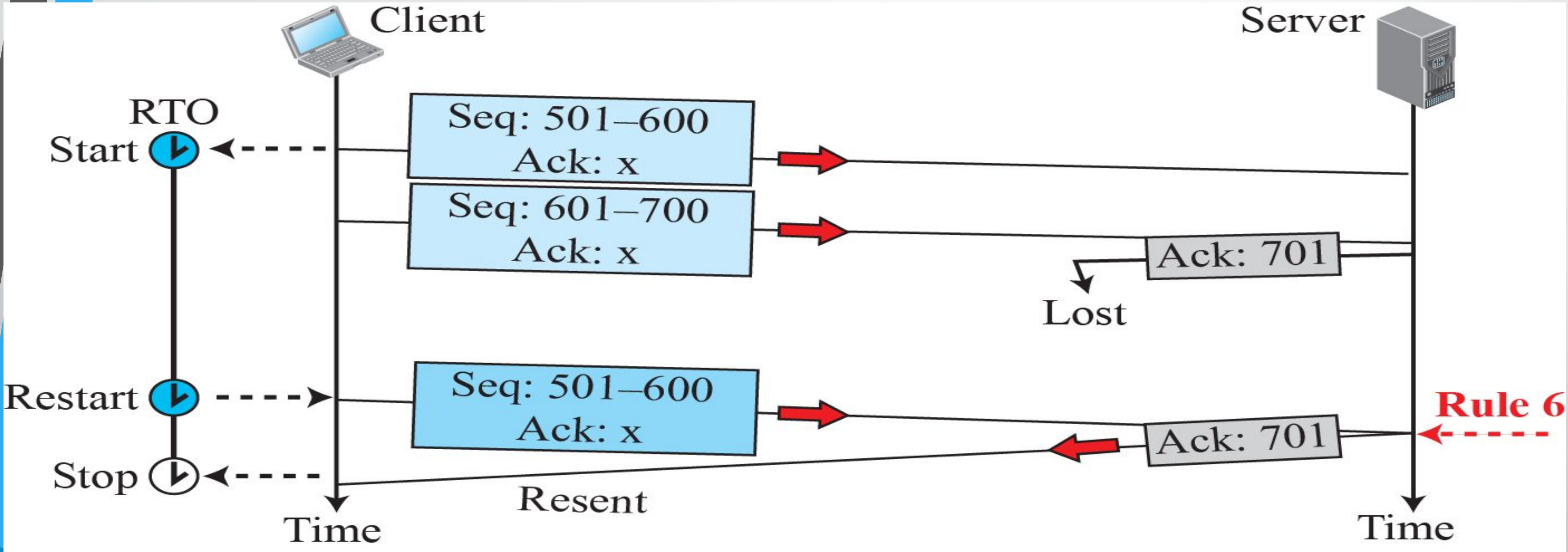
Fast Retransmission :: 3 Acks



Lost Acknowledgement



Lost Ack corrected by resending a segment



Objectives (Part 5)

- Reliable service
  - Flow Control
    - Sliding Window Concept

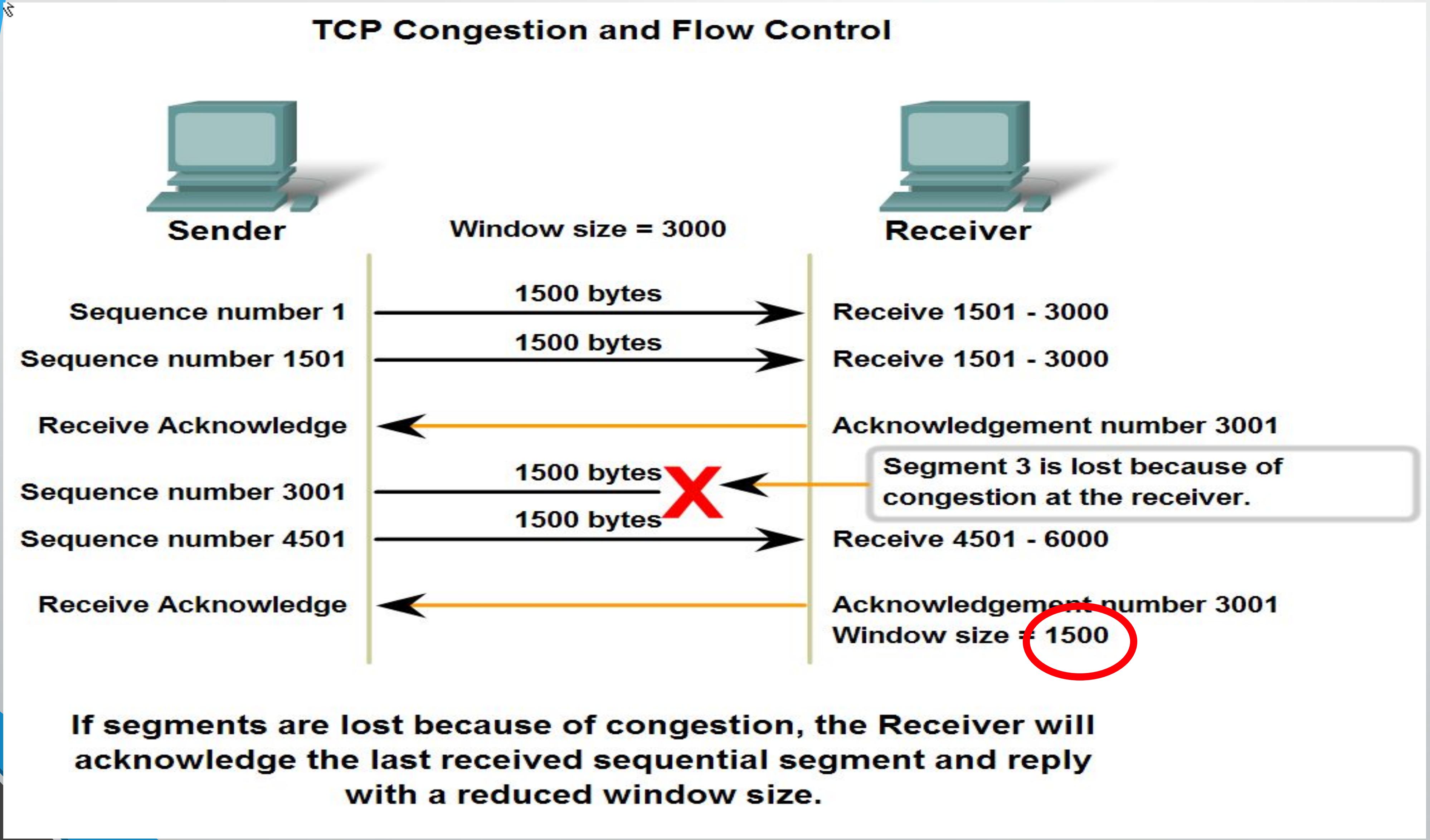
Flow Control

- Transmission Control Protocol (TCP) uses a ***sliding window*** for flow control.

- What is the "**Window** "?
  - Indicates the size of the device's receive buffer for the particular connection.
  - How much data a device can handle from its peer at one time before it is passed to the application process.
  - Set by receiver of data

  - **Example** : The server's window size was **360**. This means the receiver is willing to take **no more than 360 bytes** at a time from the sender.

Flow Control



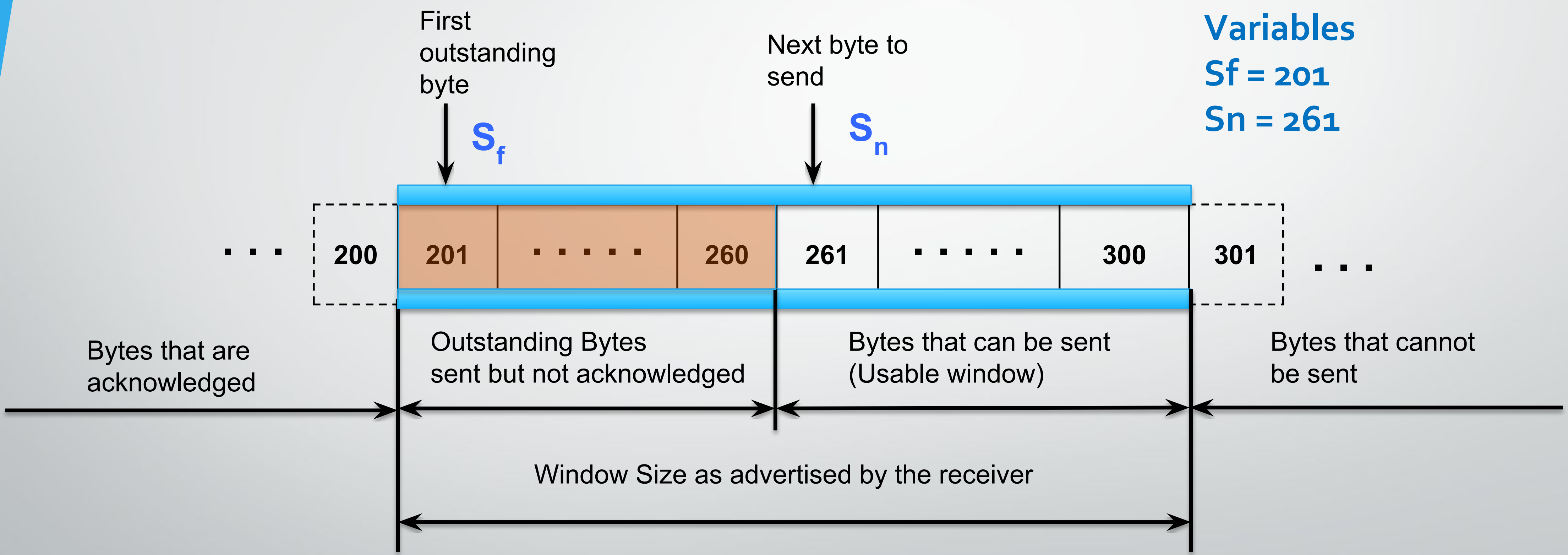
Sender Sliding Window

Window Size = 100 bytes

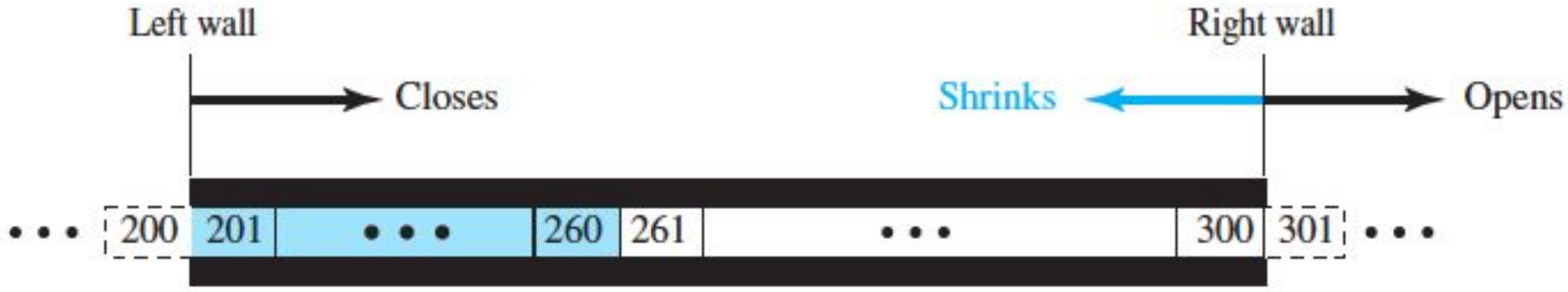
Variables

Sf = 201

Sn = 261



Sliding of Sender Window



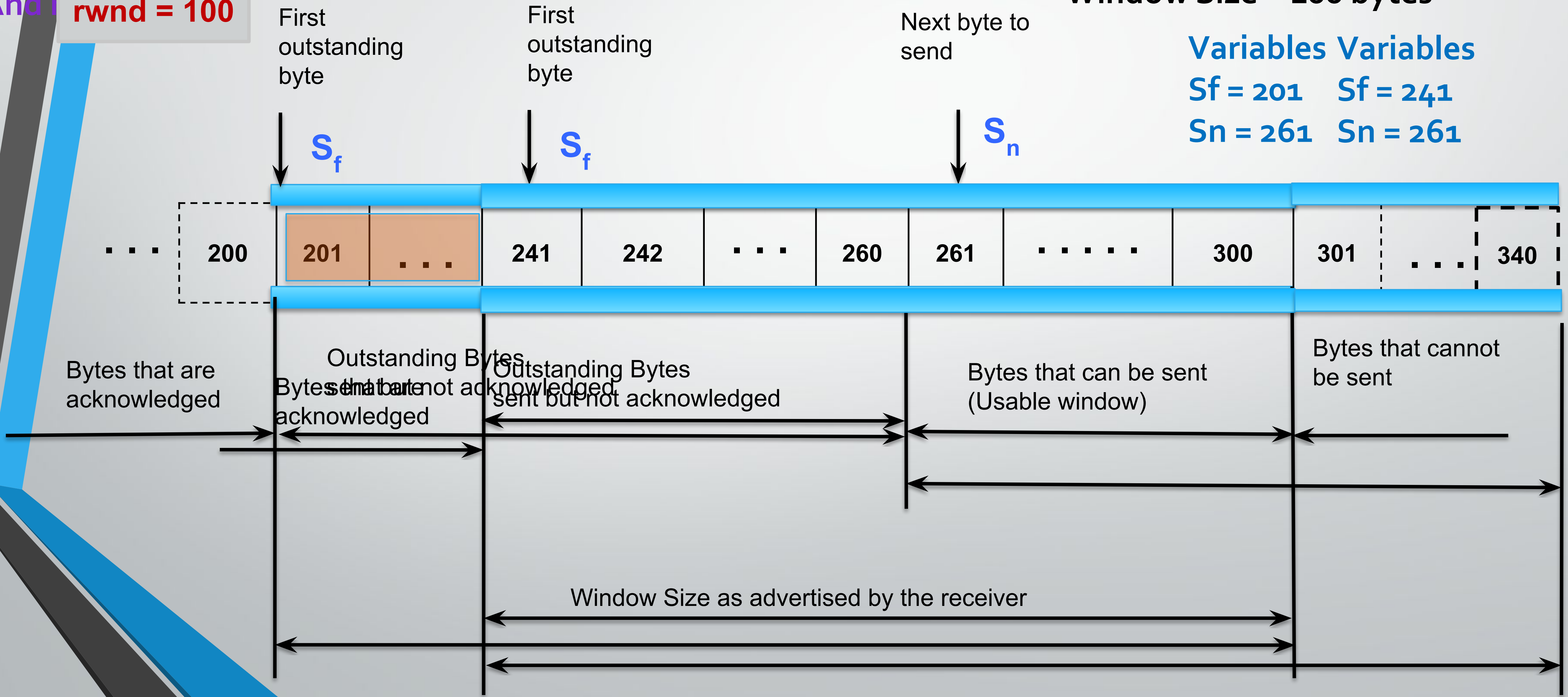
b. Opening, closing, and shrinking send window

Sliding of Sender Window

#Sender receives a segment with ACK 241

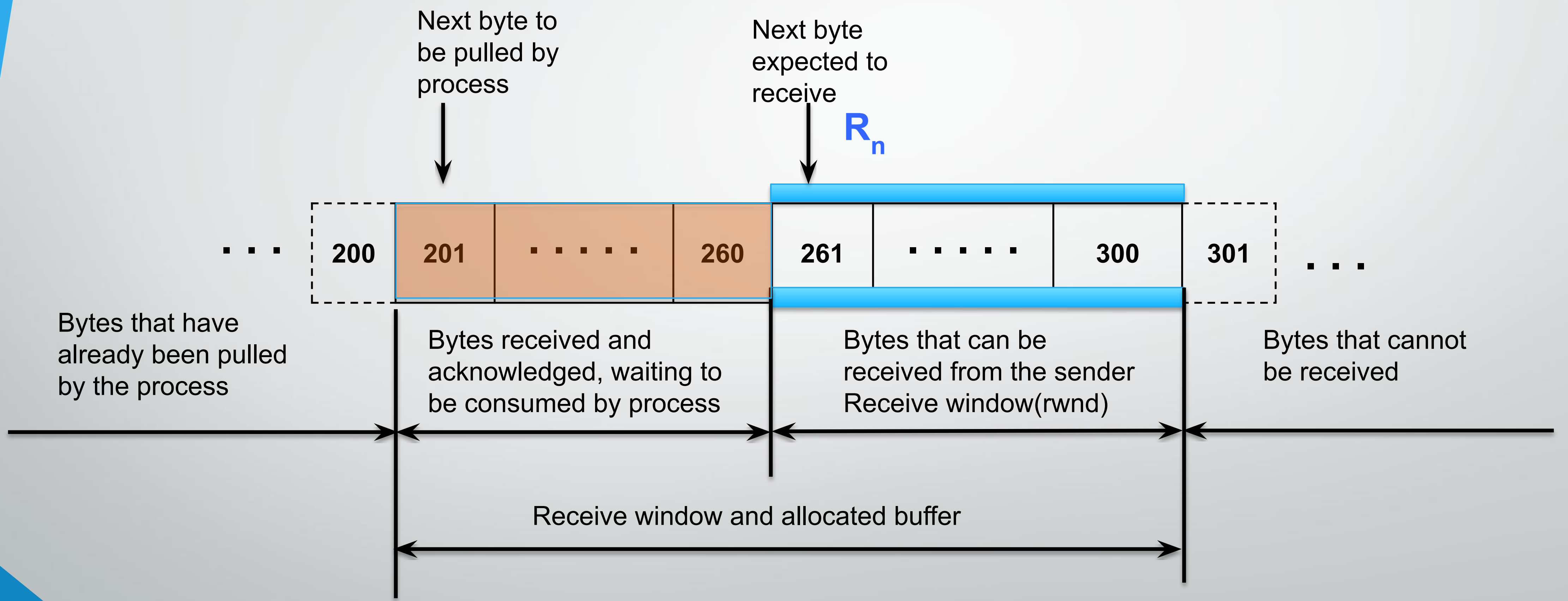
And rwnd = 100

Window Size = 100 bytes



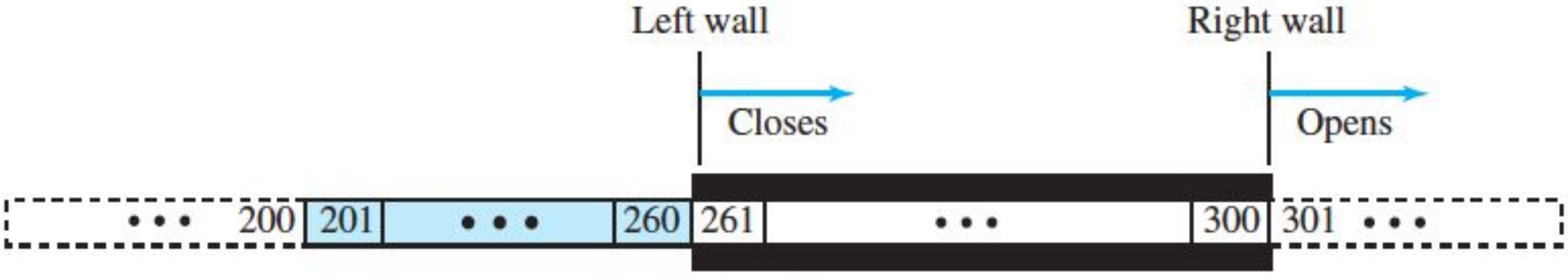
Receiver Sliding Window

Window Size = 100 bytes



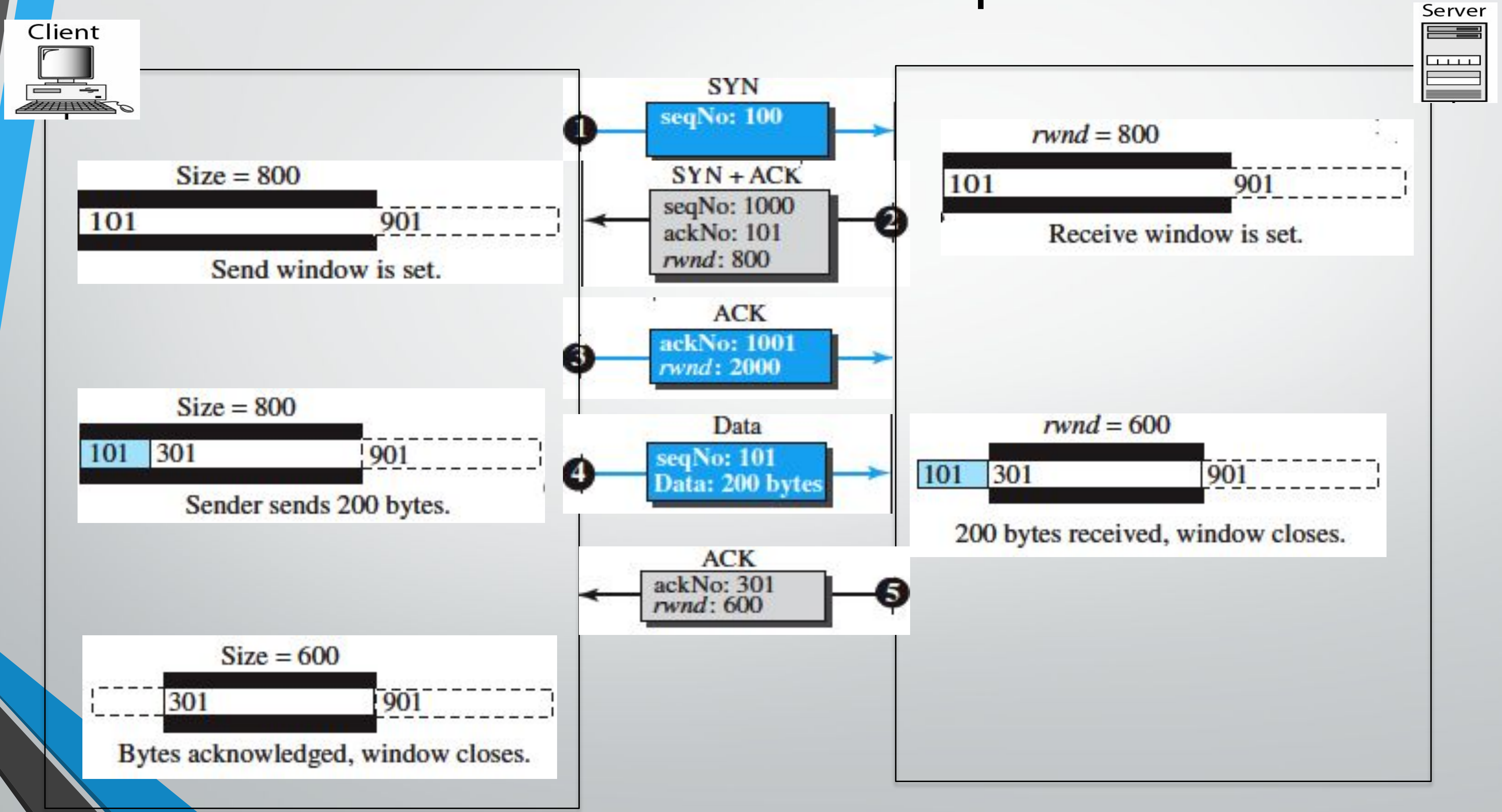
- rwnd= buffer size – number of bytes to be pulled = 40 bytes

Sliding of Receiver Window

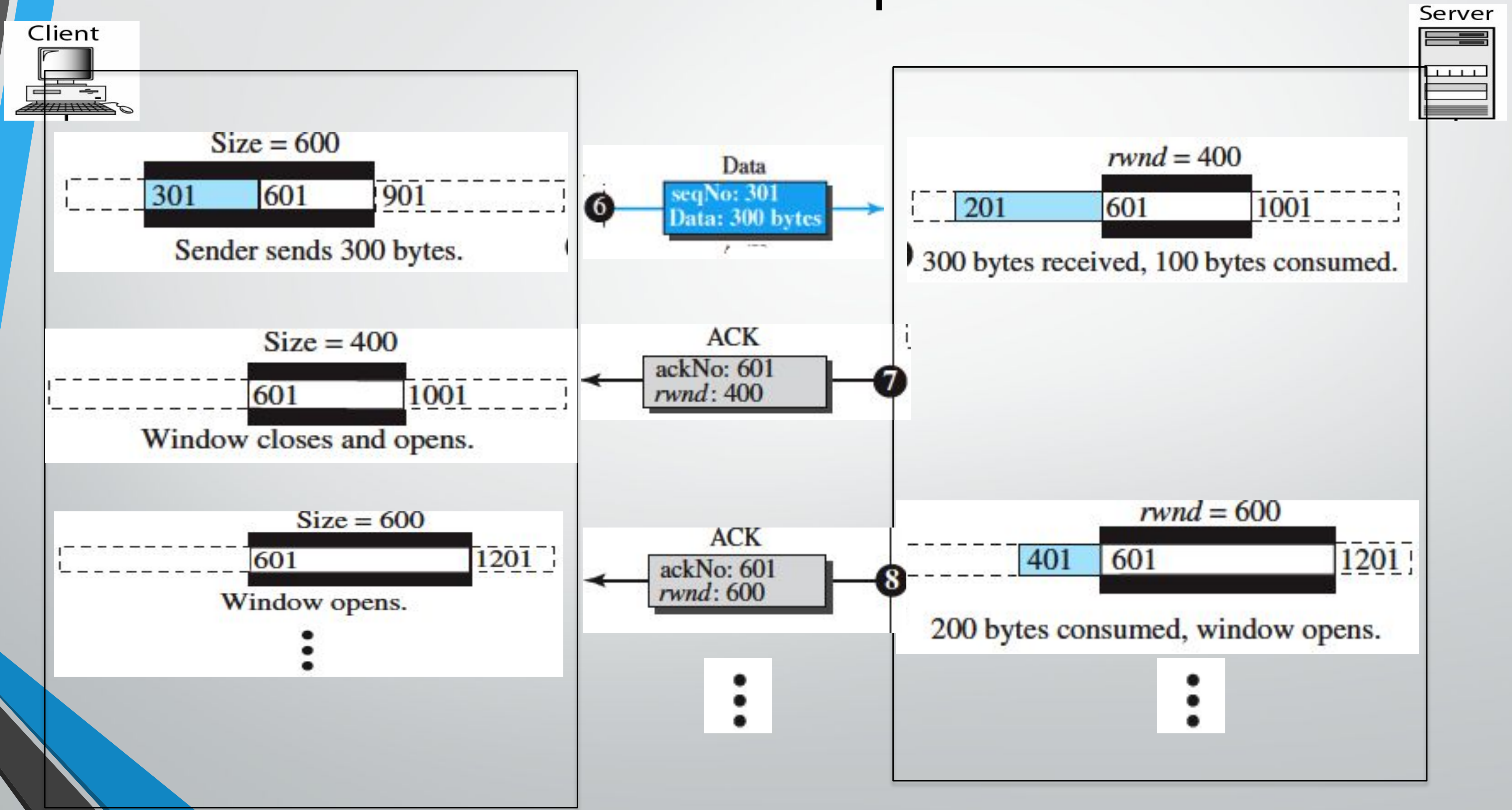


b. Opening and closing of receive window

Flow Control Example



Flow Control Example Contd



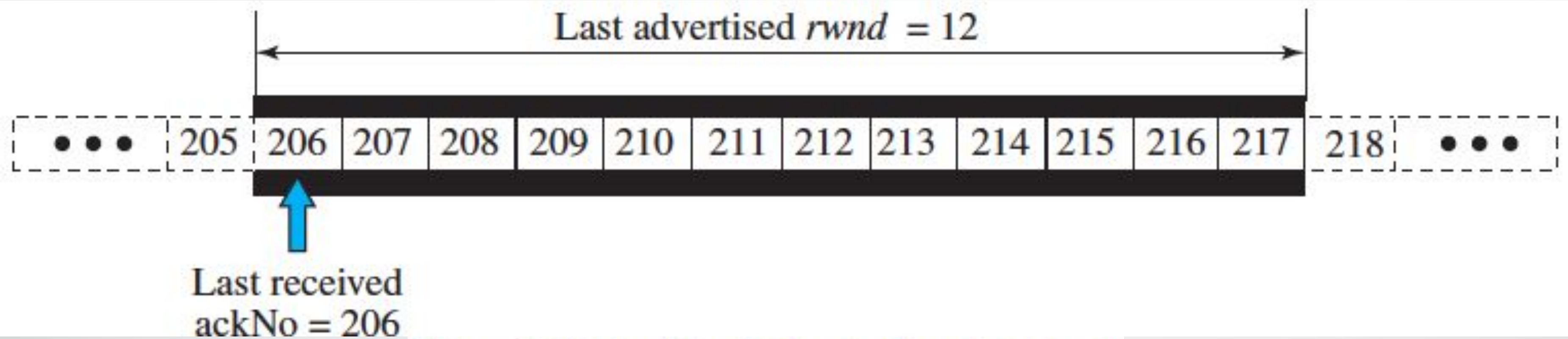
Shrinking of Windows

- The receive window cannot shrink
- The send window can shrink if the receiver defines a such value for rwnd
- However, some implementations do not allow shrinking of the send window
- The receiver needs to keep the following relationship

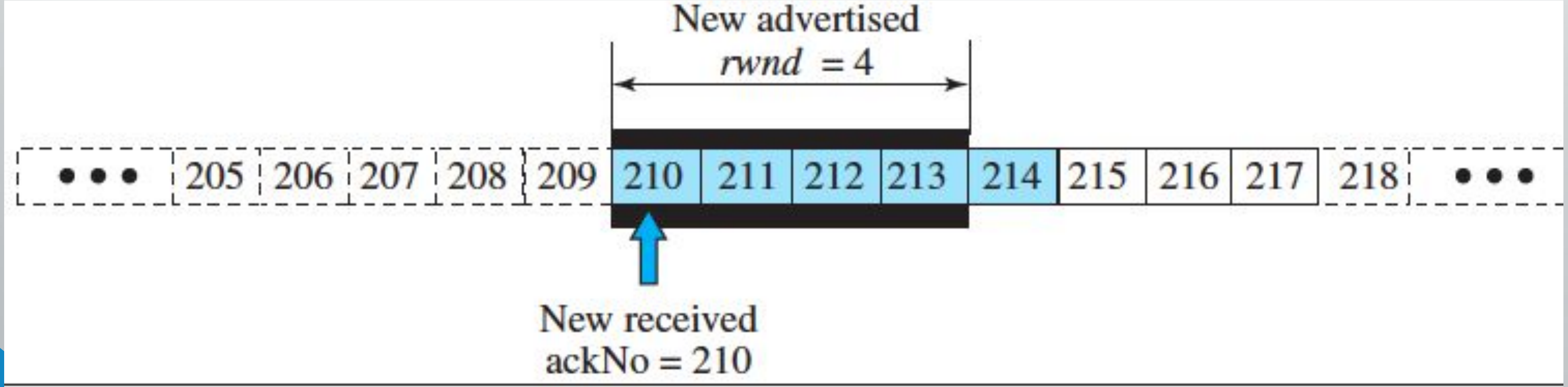
new ackNo + new rwnd ≥ last ackNo + last rwnd

Shrinking of Windows

new ackNo + new rwnd ≥ last ackNo + last rwnd



The window after the last advertisement



The window after the new advertisement; window has shrunk

Different TCP Sliding Window Protocols

- **Go Back N Protocol**
  - If the sent segment are are found corrupted or lost then all the segments are re-transmitted from the lost segment to the last segment transmitted
  - Do not keep track of out of order segments
  - Efficient for less noisy channel

Different TCP Sliding Window Protocols

- **Selective Repeat Protocol**
  - Only those segments are re-transmitted which are found lost or corrupted
  - Keep track of out of order segments at the receiver side
  - More efficient for noisy channels
  - Widely used in TCP

Overall Flow control

- The initial window size is agreed during the **three-way handshake.**
- If this is too much for the receiver and it **loses data** (e.g. buffer overflow) then it can **decrease** the window size.
- If **all is well** then the receiver will **increase** the window size.

The End