Transport Layer (TCP Introduction)

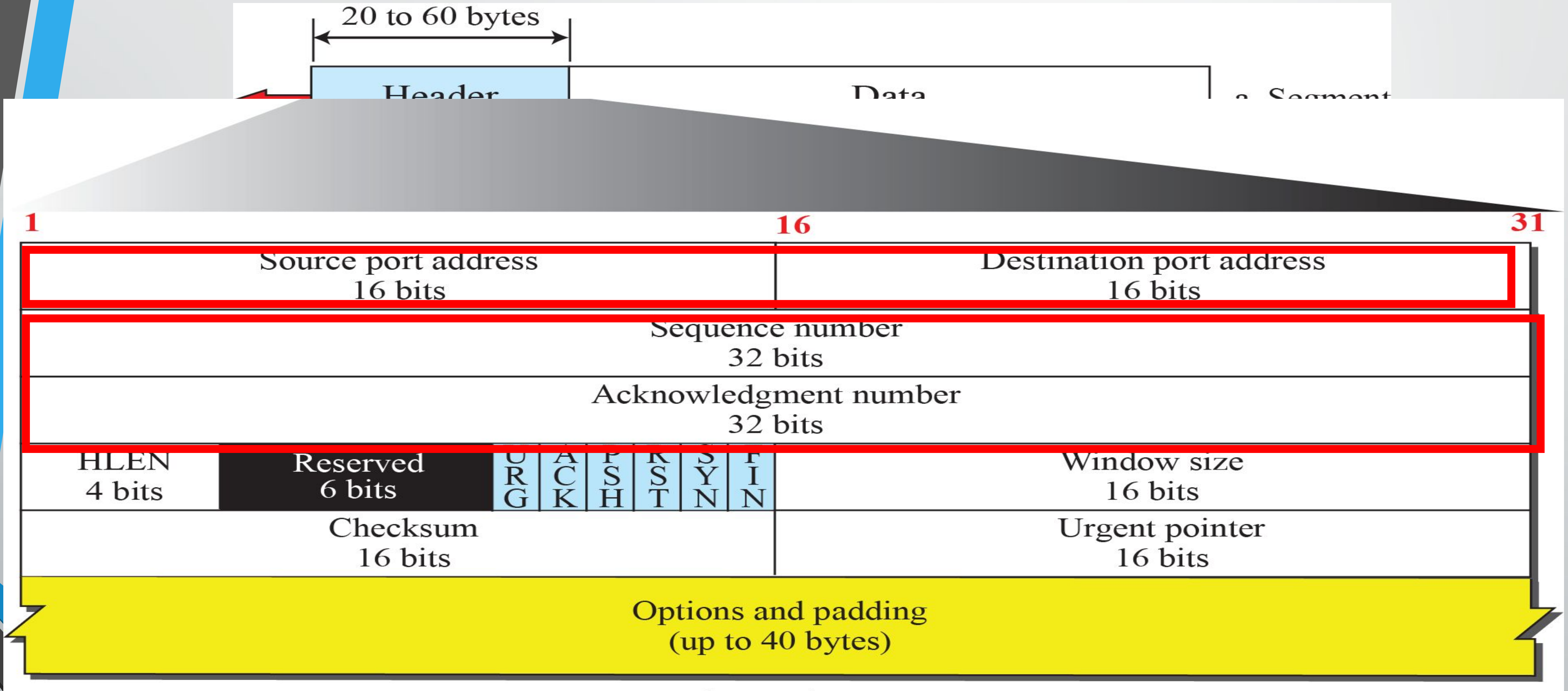
Lecture 5 | CSE421 – Computer Networks

Department of Computer Science and Engineering
School of Data & Science

Objectives

- TCP Header
- TCP Services

TCP Segment Header



b. 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 is acknowledged using the acknowledgement number

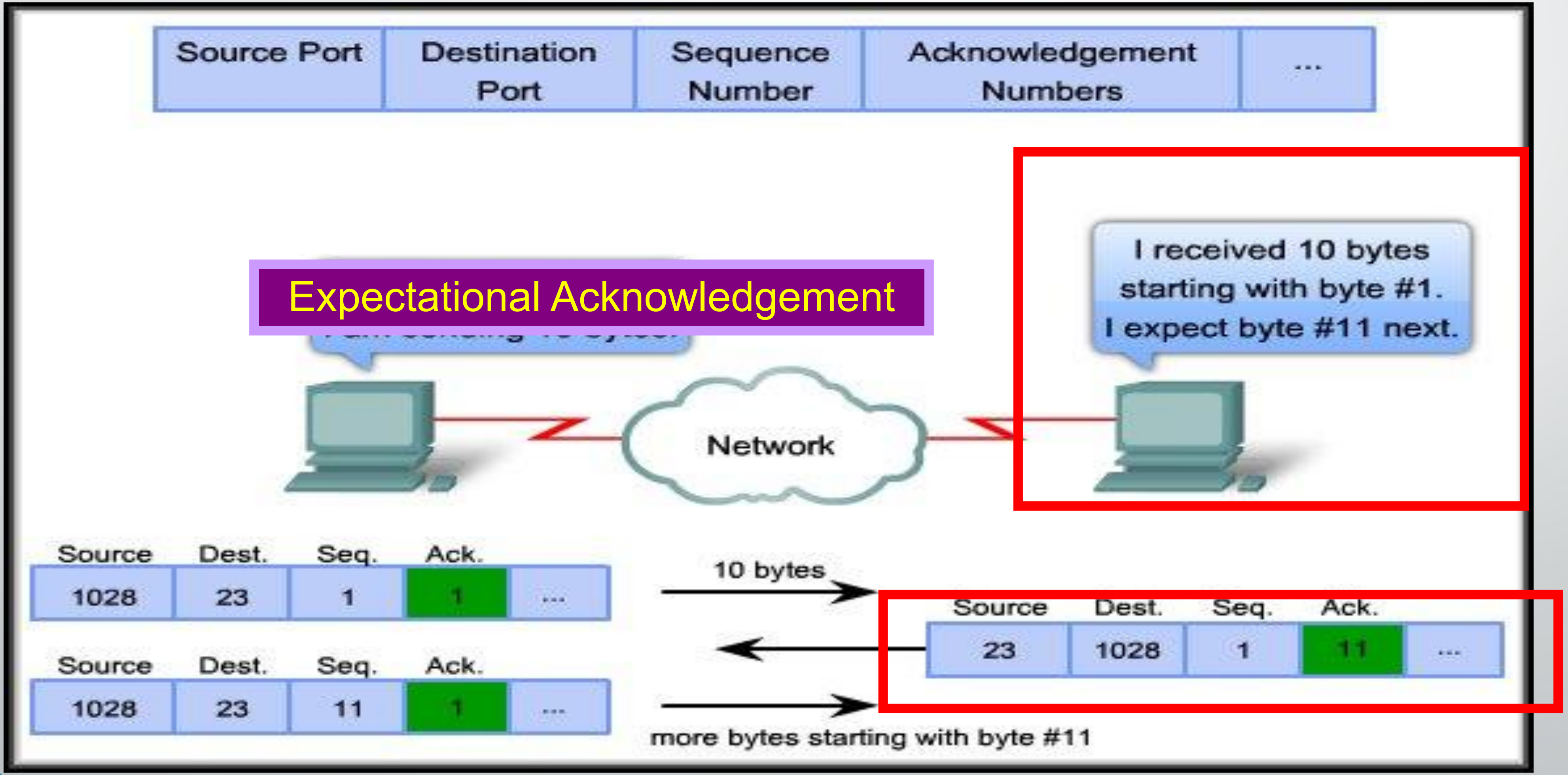
- The value of the acknowledgment field in a segment defines the **number of the next byte** the receiver expects to receive.

- 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.

Acknowledgement Number



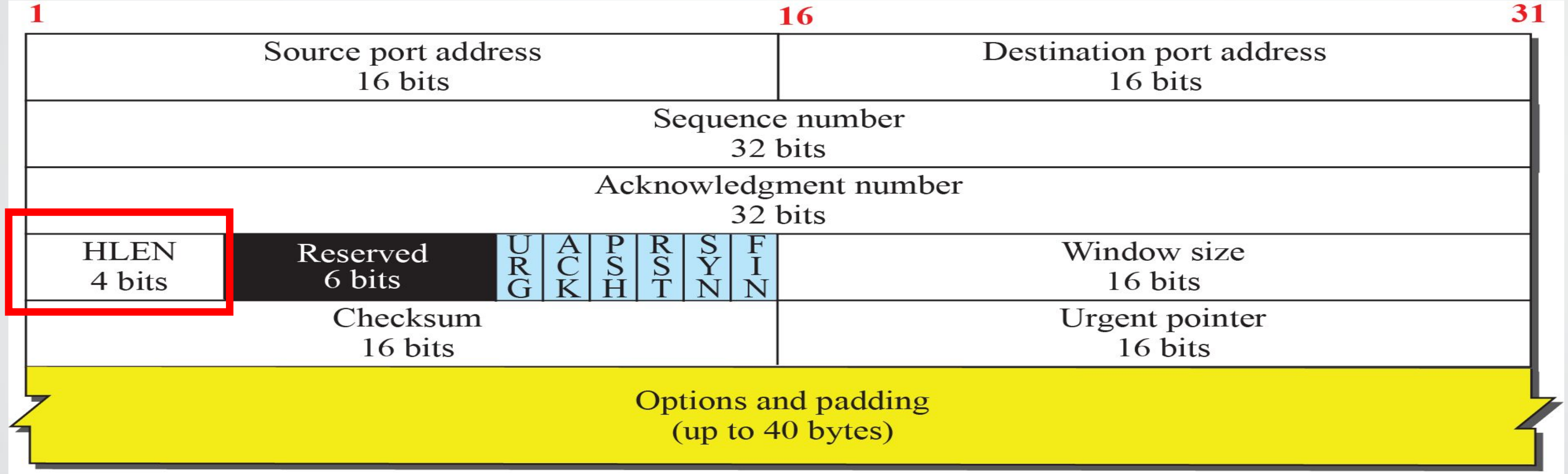
Acknowledgement Number

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

Sender

Receiver

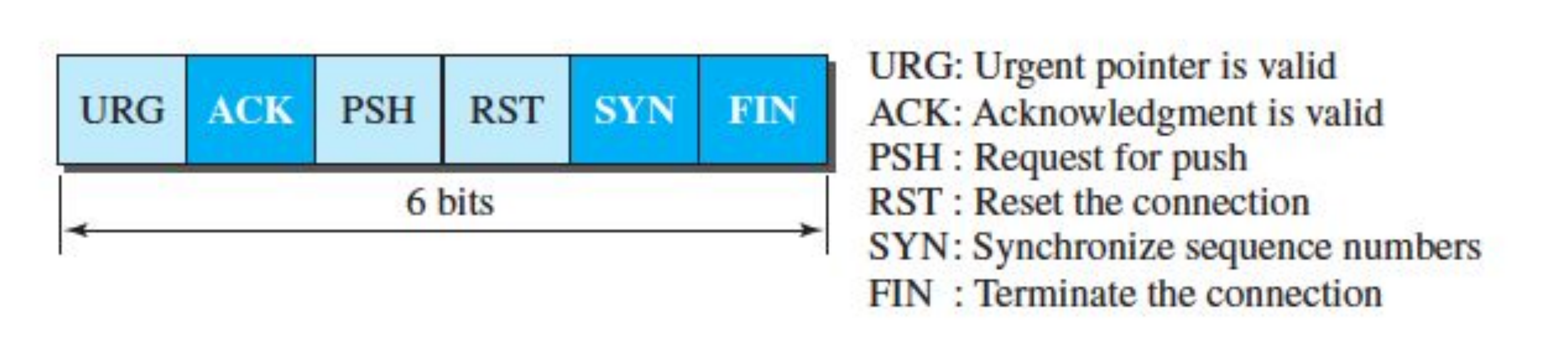
Header Length



b. Header

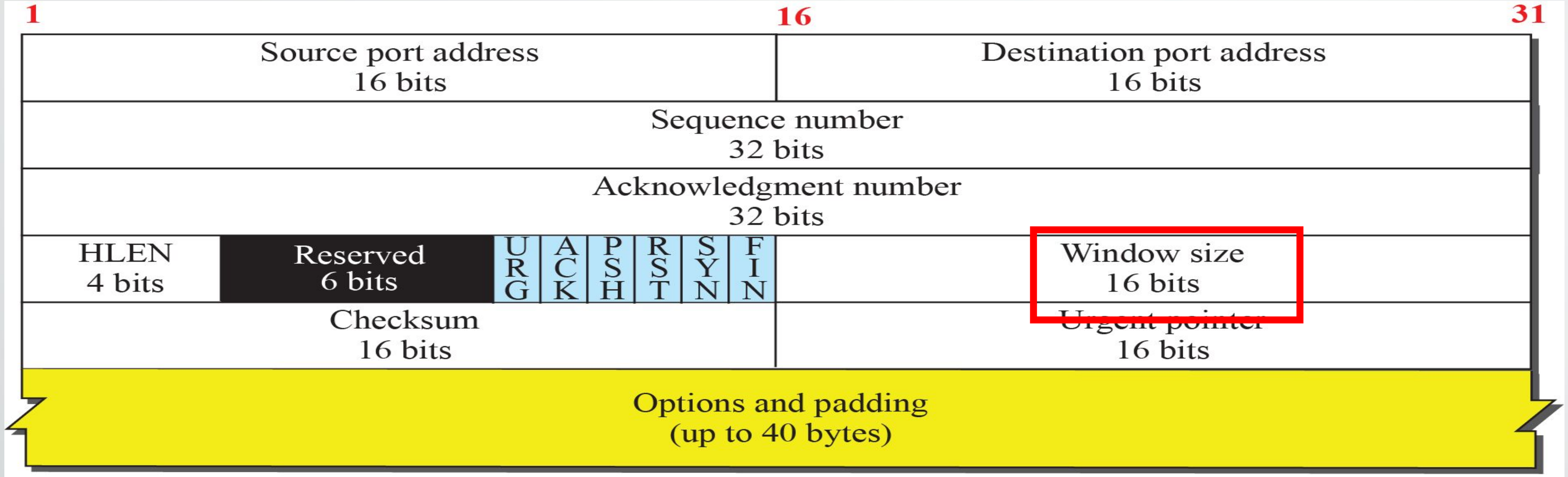
- Header Length :
  - Indicates the number of 4-byte words
  - The length of the header can be between 20 and 60 bytes

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

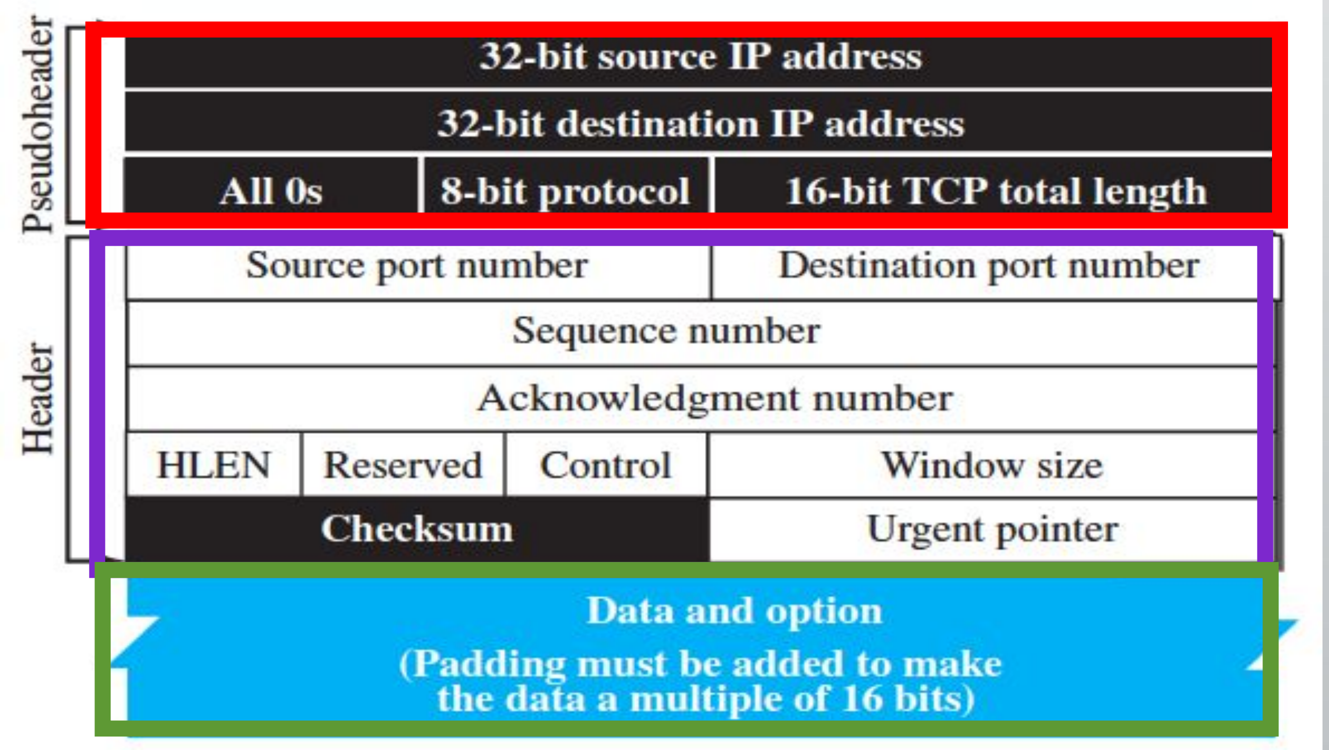
Checksum

- This 16 bits field is used to detect errors (*i.e.,* flipped bits) in the transmitted segment (intentionally or unintentionally) while traveling through the network.
- Also present in UDP header
- Mandatory in TCP but not in UDP
- Process is same for both protocols

UDP HEADER

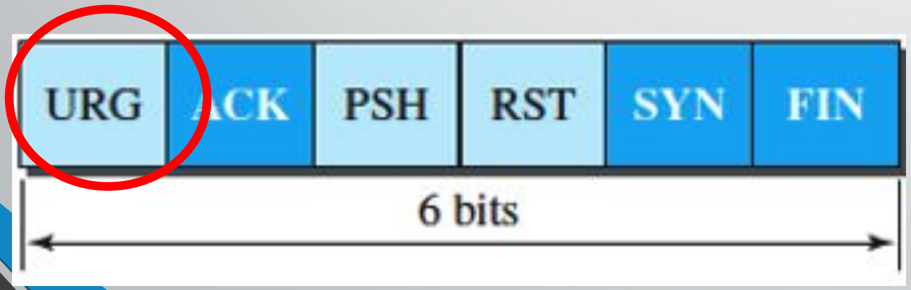
Source port number 16 bits	Destination port number 16 bits
Total length 16 bits	Checksum 16 bits

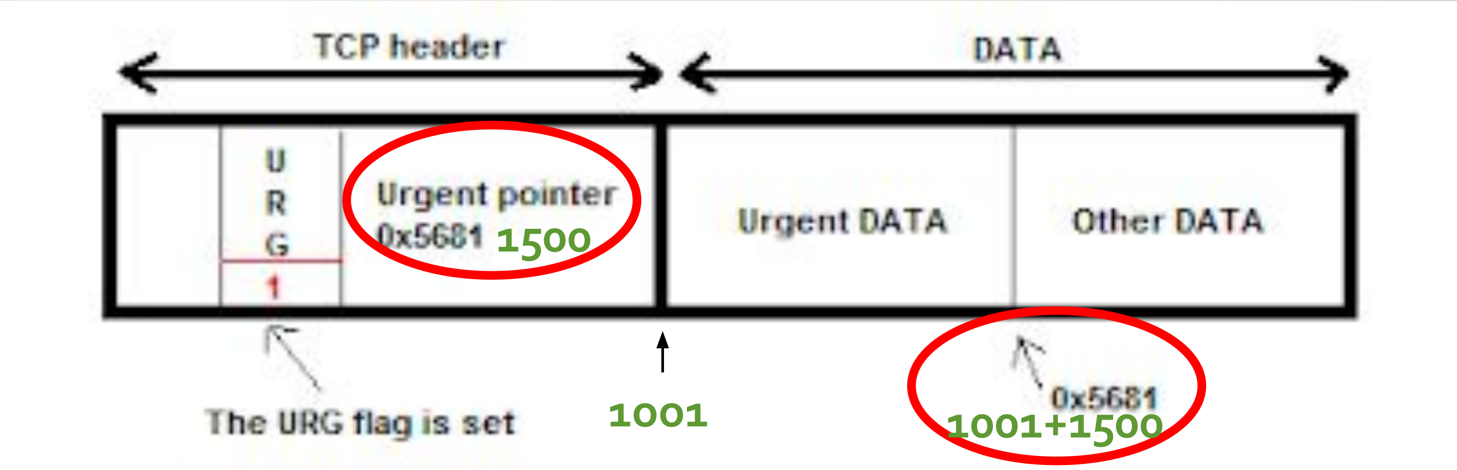
- TCP/UDP Header
- TCP/UDP Body
- Pseudo IP Header



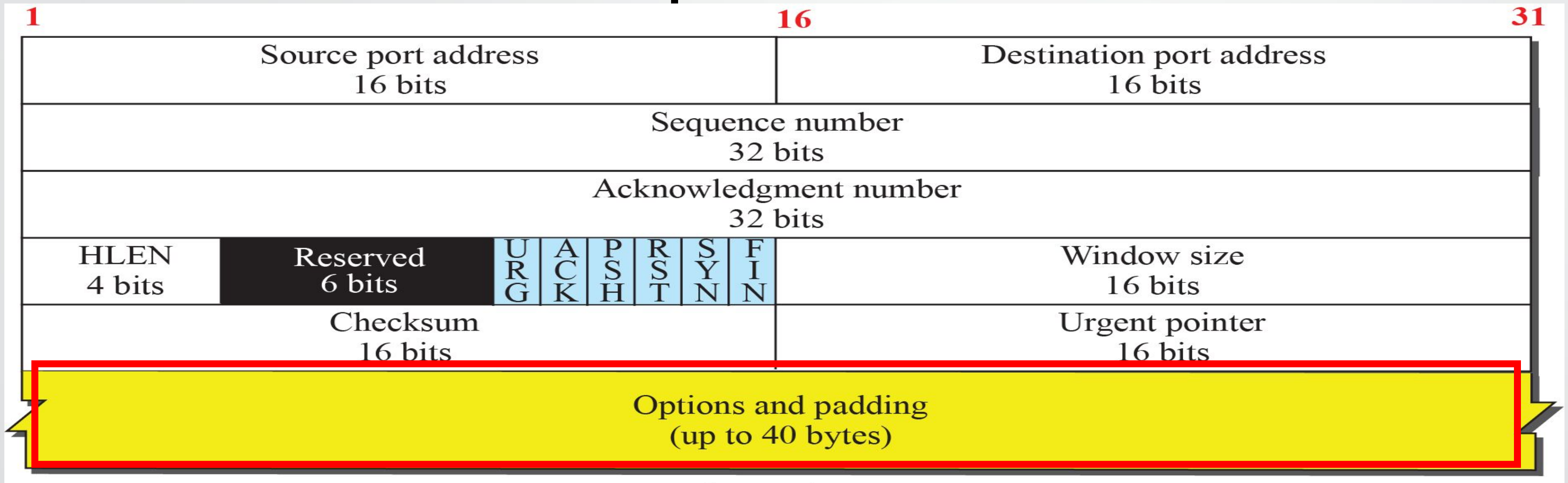
Urgent Pointer

- 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

Functions of the Transport Layer

- Primary responsibilities:

UDP &TCP

1. Segmenting the data and managing each piece.
2. Reassembling the segments into streams of application data.
3. Identifying the different applications.
4. Multiplexing

Only TCP

5. Establishing and terminating a connection.
6. Enabling error control and recovery.
7. Performing flow control between end users.

- Function 6
  Connection Establishment and Termination for Reliability

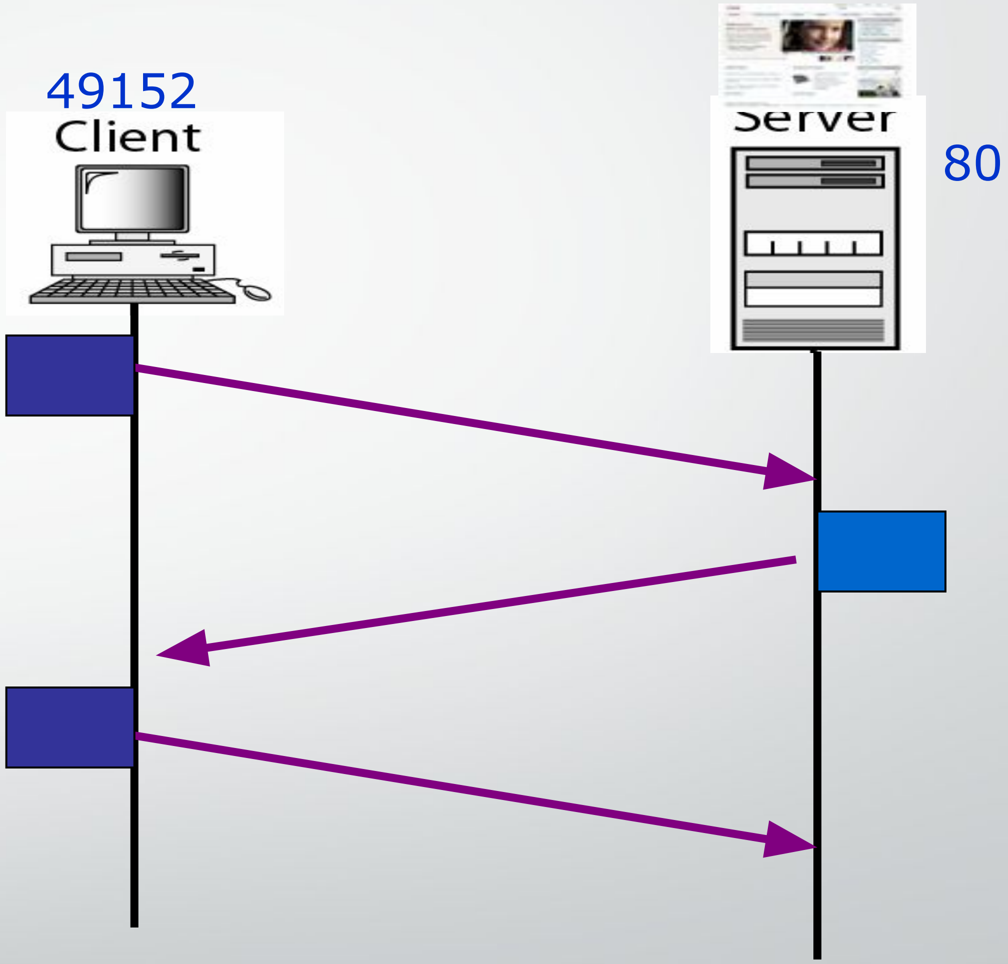
Connection Establishment

- 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

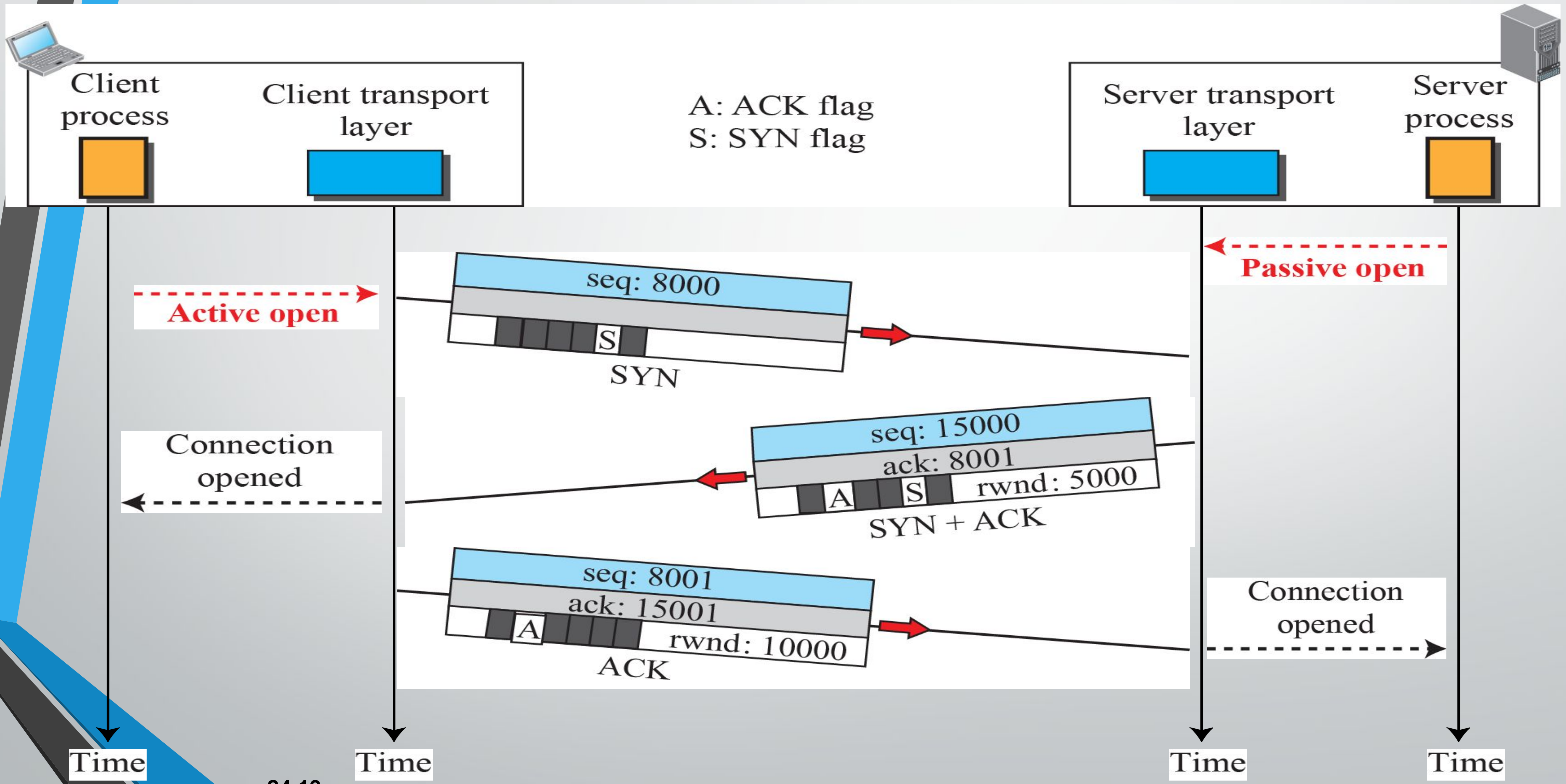
3 Way Handshake : Connection Establishment

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

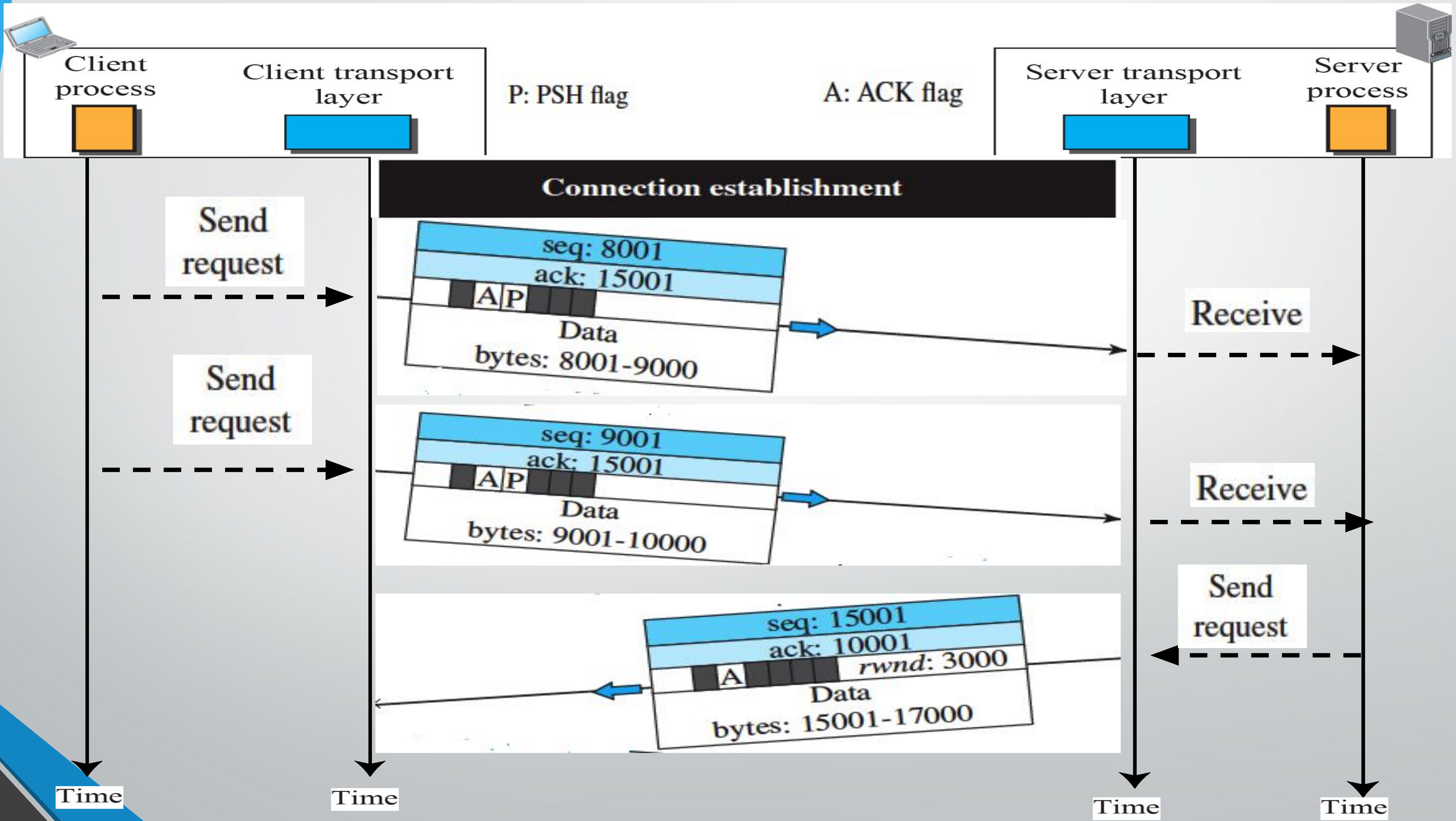
49152 80				80 49152 80			
100 0 101							
0 1011							
		A			S		4000 Bytes 8000 Bytes
Others							



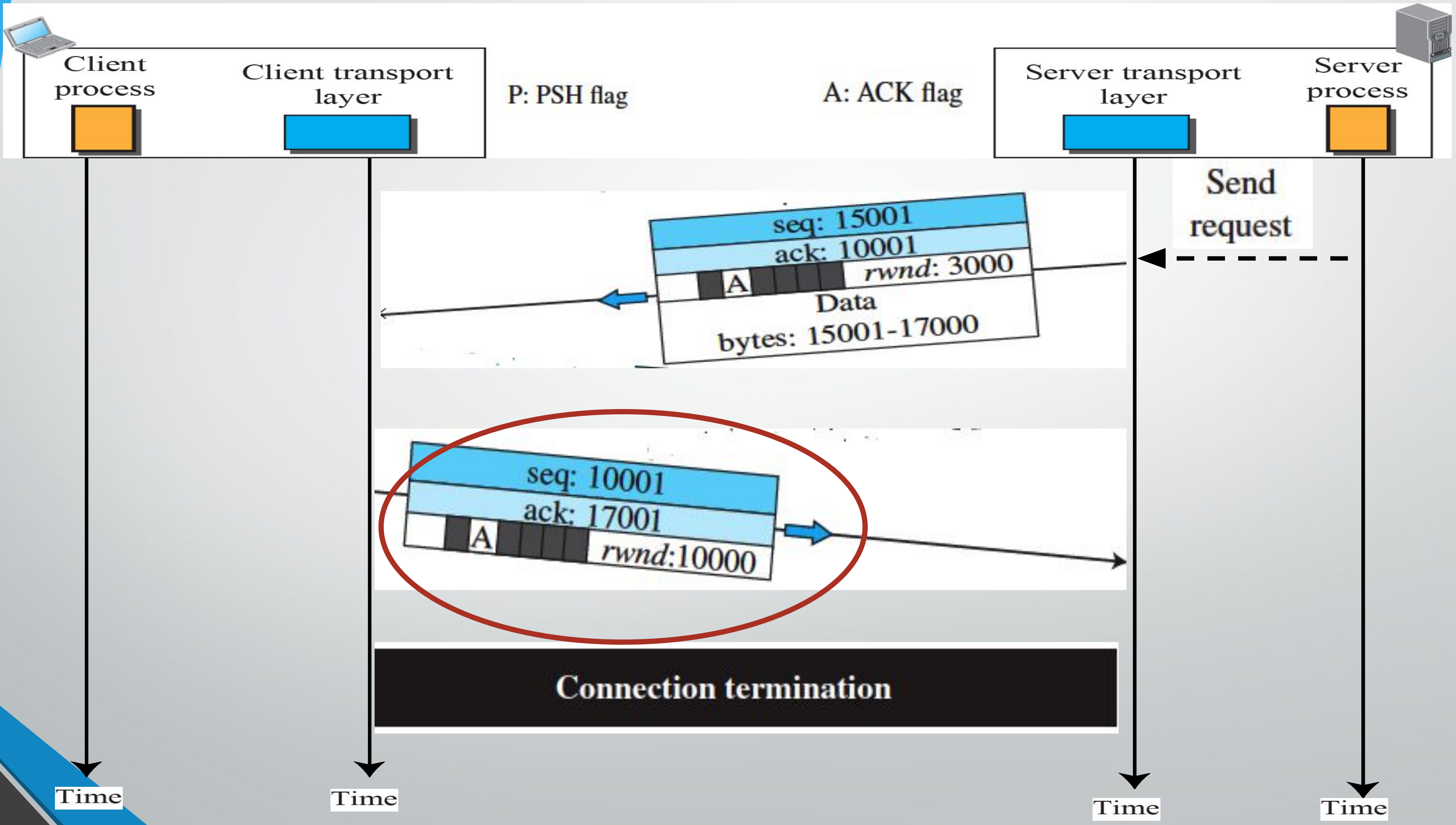
3 Way Handshake : Connection Establishment



Data Transfer



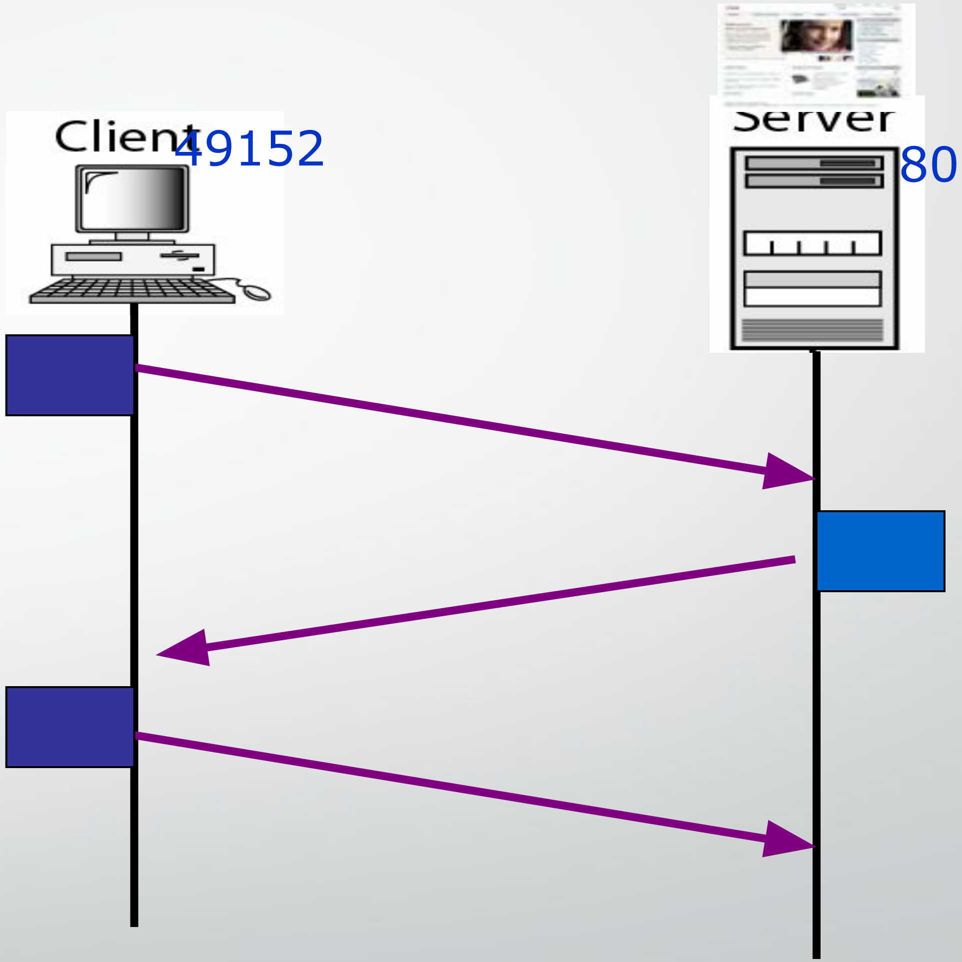
Data Transfer Continued..



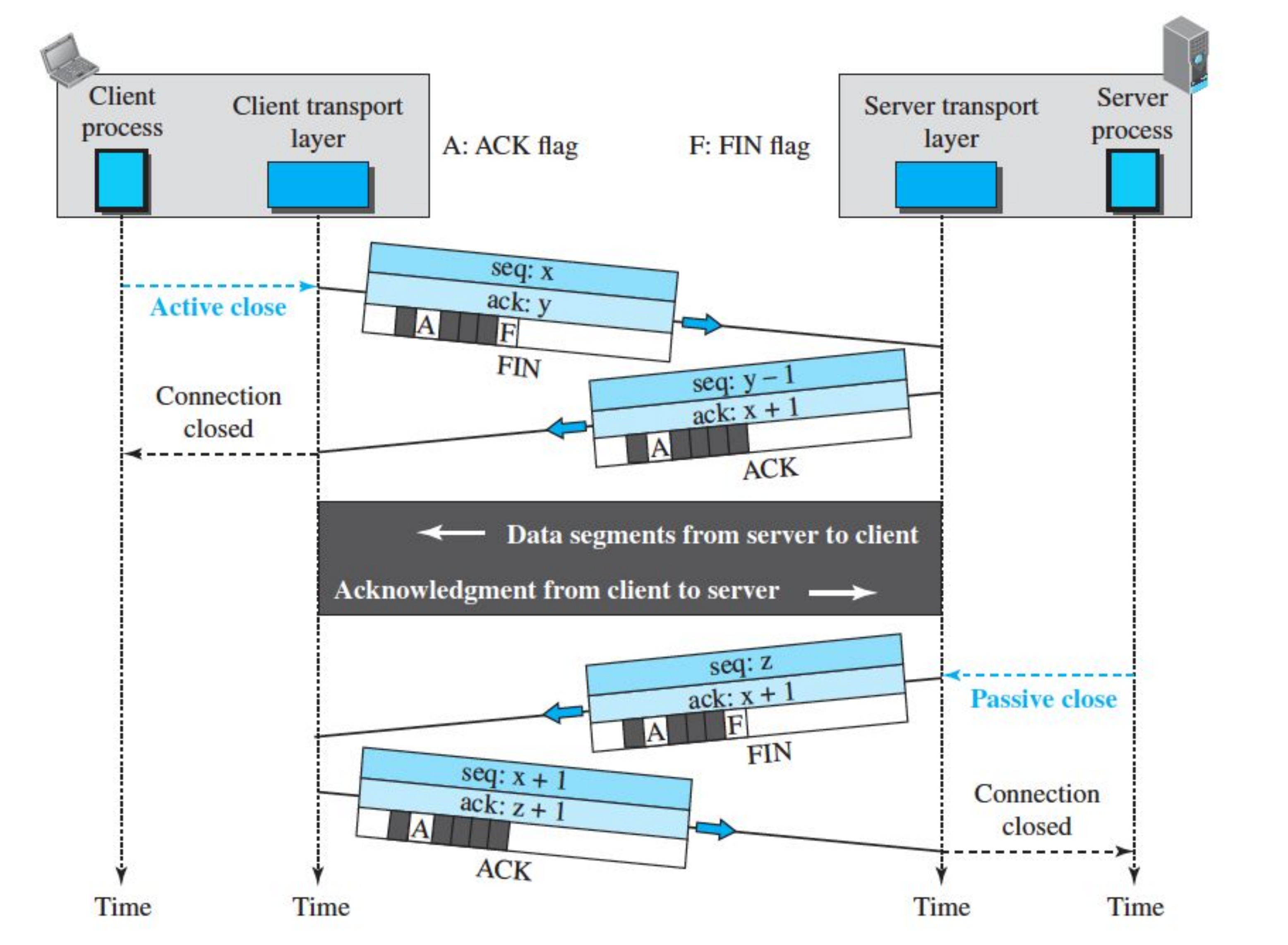
3 Way Handshake : Connection Termination

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

49152 80				80 49152			
10001 17001 10002							
17001 17002 10002							
		A				F	4000 Bytes 8000 Bytes
Others							



Connection Termination :: Half Close



- Function 6
  Error Control and Recovery for Reliability

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

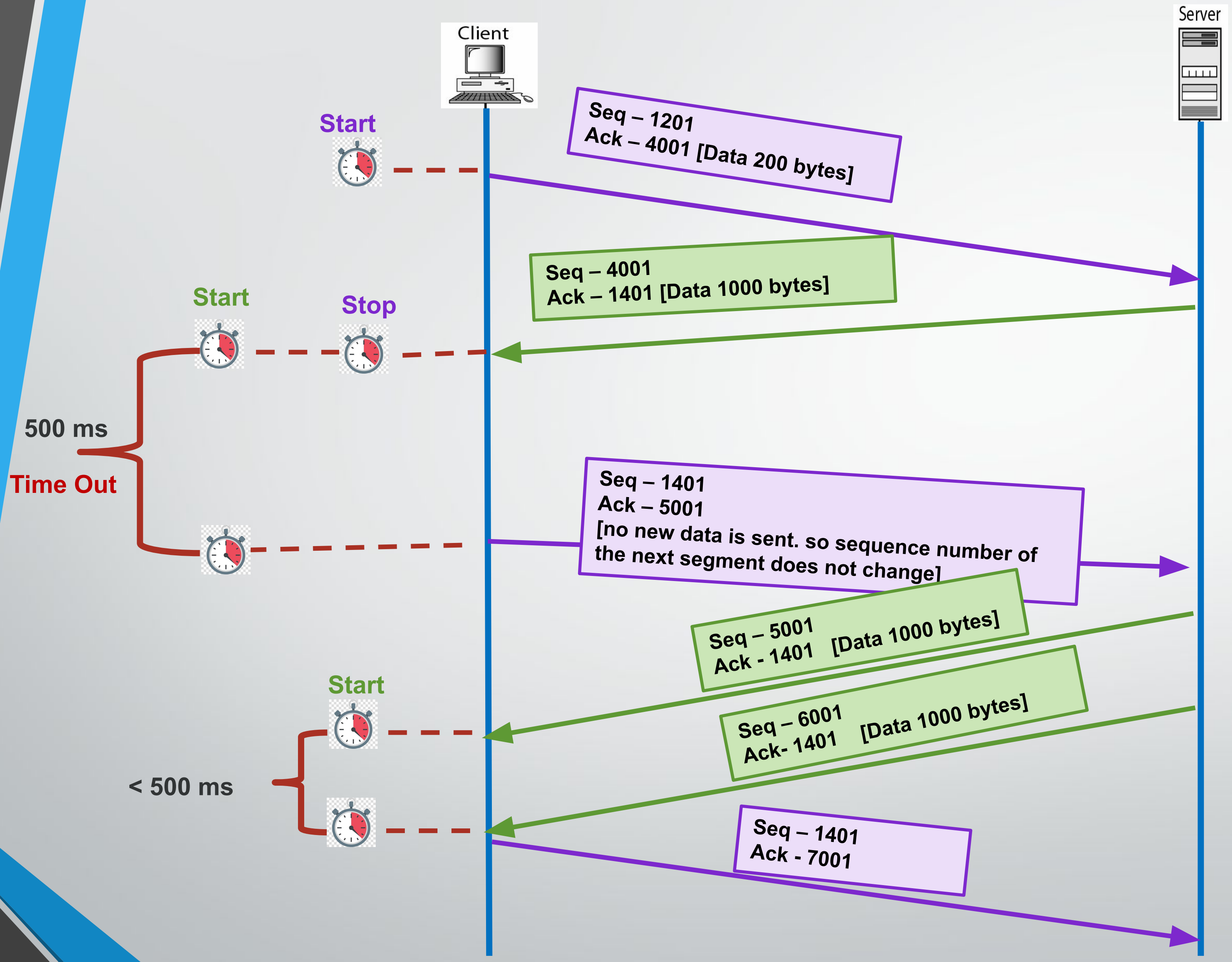
- **Checksum**
  - Each segment includes a checksum field, which is used to check for a corrupted segment
  - If a segment is corrupted, as detected by an invalid checksum, the segment is discarded

- **Acknowledgment**
  - Using Acknowledgement Number 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.

Error Control

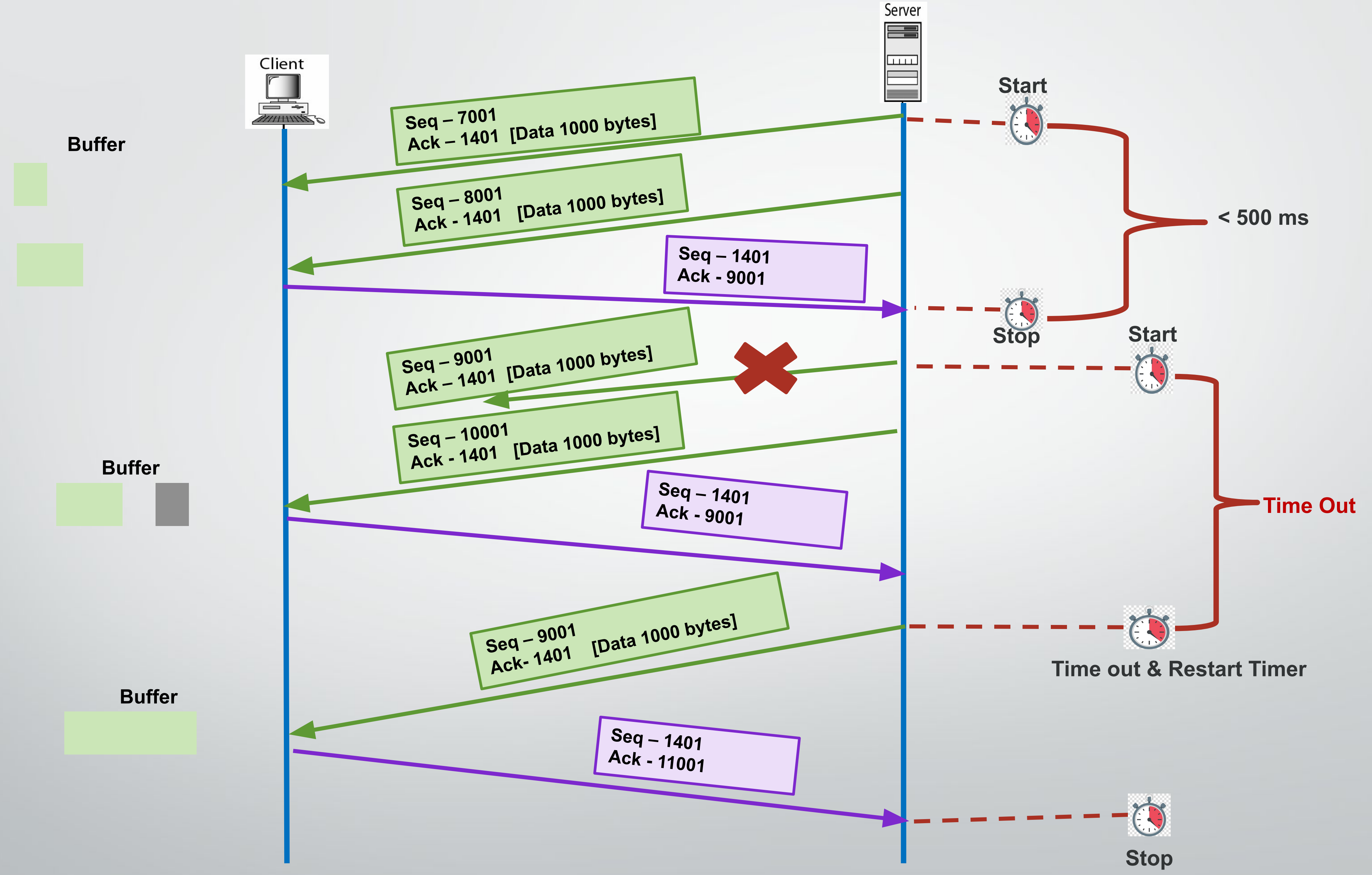
- **Retransmission**
- When a segment is sent, it is stored in a queue until it is acknowledged.
- Retransmission of segment will occur
- **After Retransmission Time Out**
  - The sending TCP maintains one retransmission time-out (RTO) timer for each connection.
  - When the timer matures TCP resends the segment in the front of the queue if the segment is not acknowledged
- **After Three Duplicate ACK Segments**
  - To be explained later

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?
- **RTO - Retransmission** after time out.

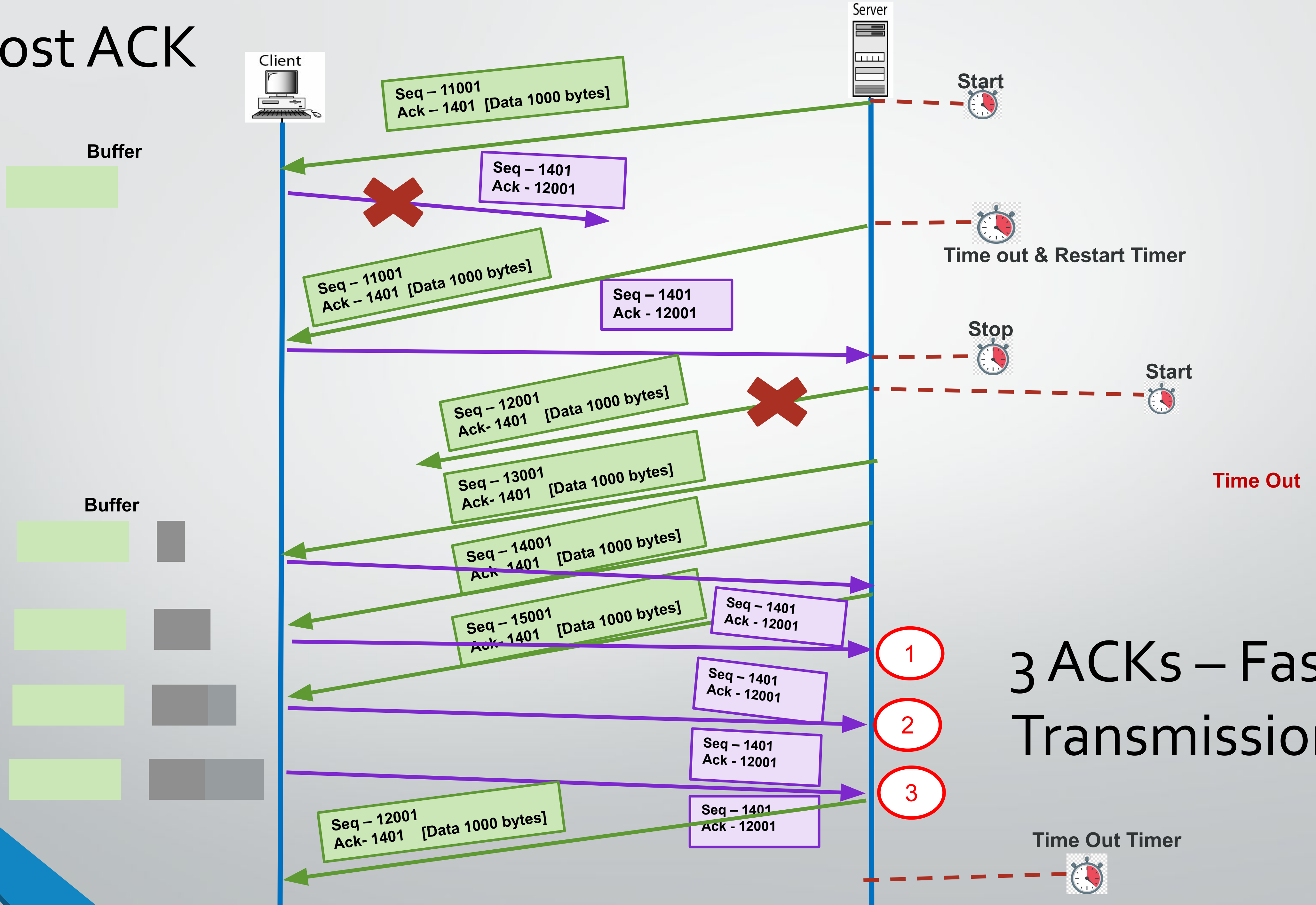


Lost Segment

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.

Lost ACK



3 ACKs – Fast Transmission

- Function 7 :
  Flow Control and Recovery for Reliability

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.

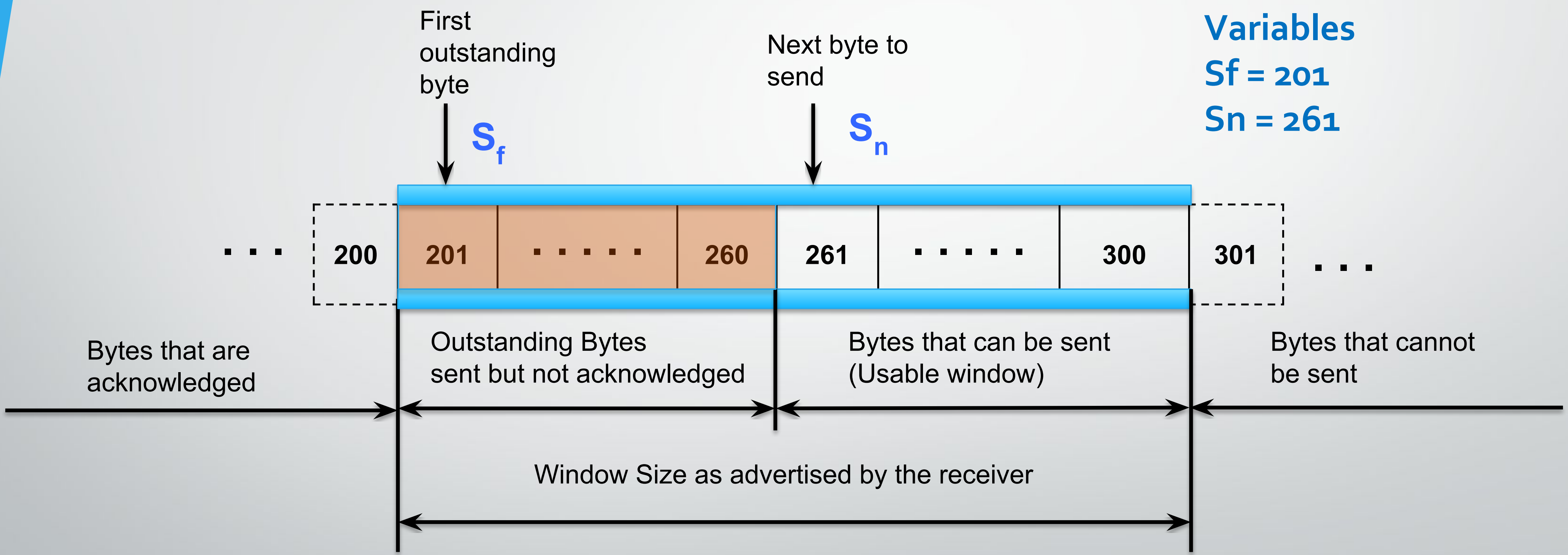
Sender Sliding Window

Window Size = 100 bytes

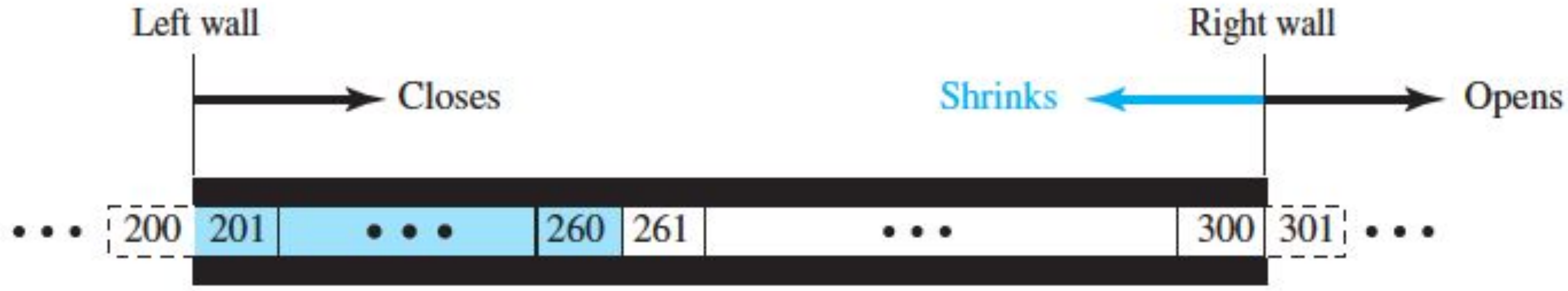
Variables

Sf = 201

Sn = 261



Sliding of Sender Window



b. Opening, closing, and shrinking send window

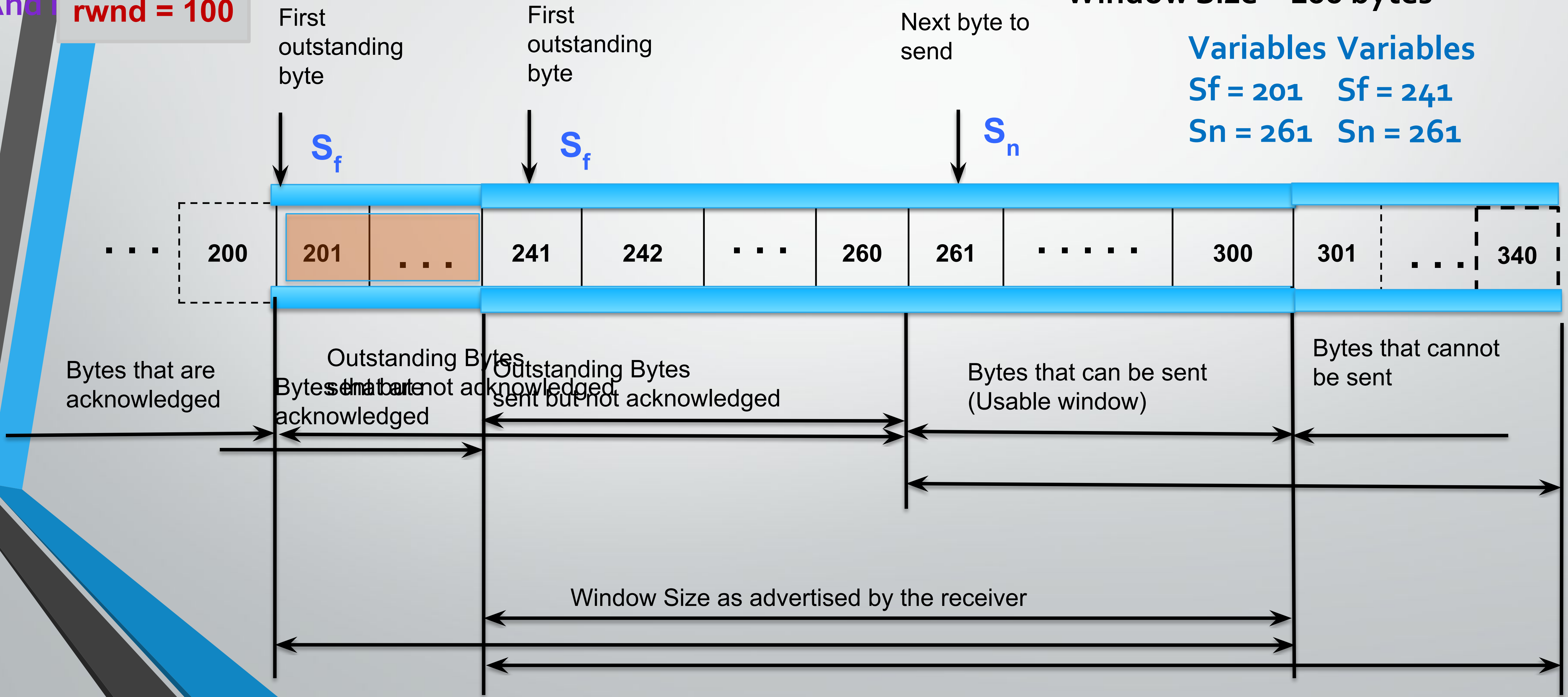
Sliding of Sender Window

#Sender receives a segment with ACK 241

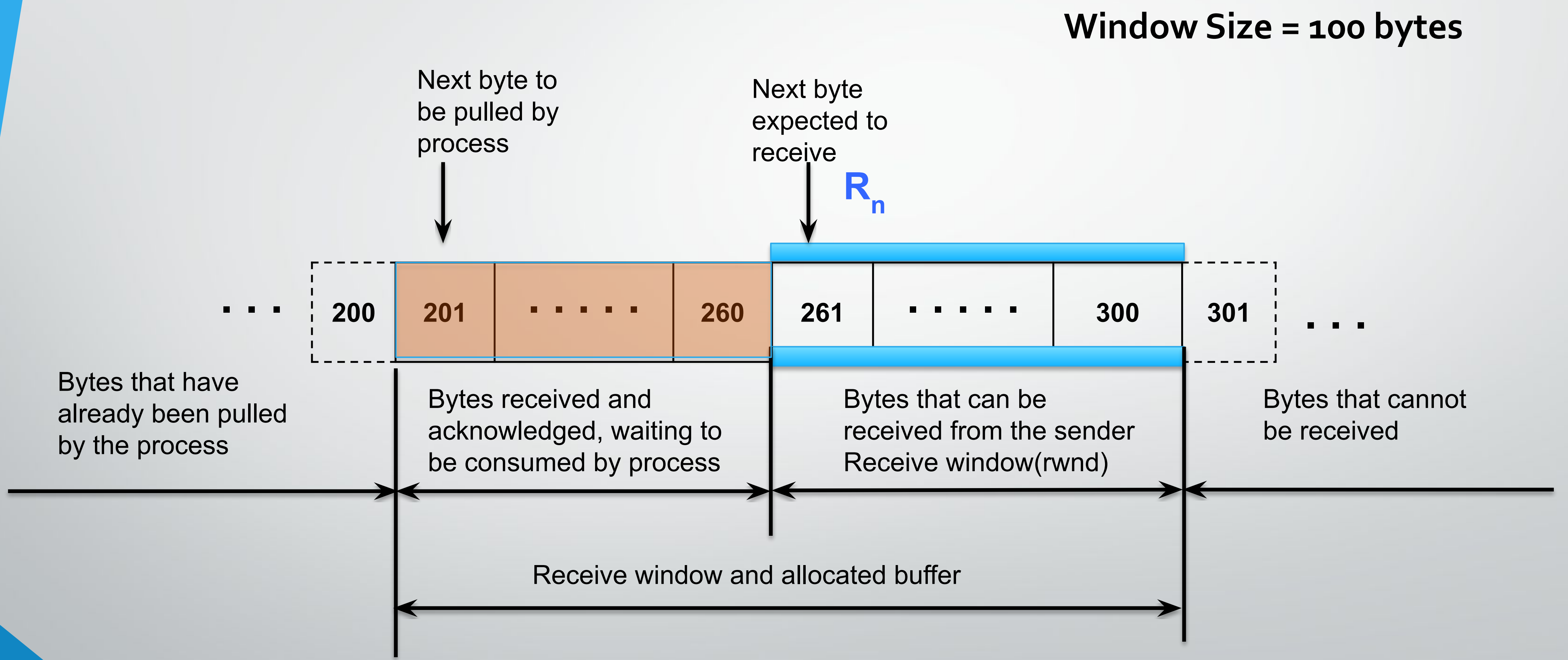
And rwnd = 100

Window Size = 100 bytes

Variables	Variables
Sf = 201	Sf = 241
Sn = 261	Sn = 261

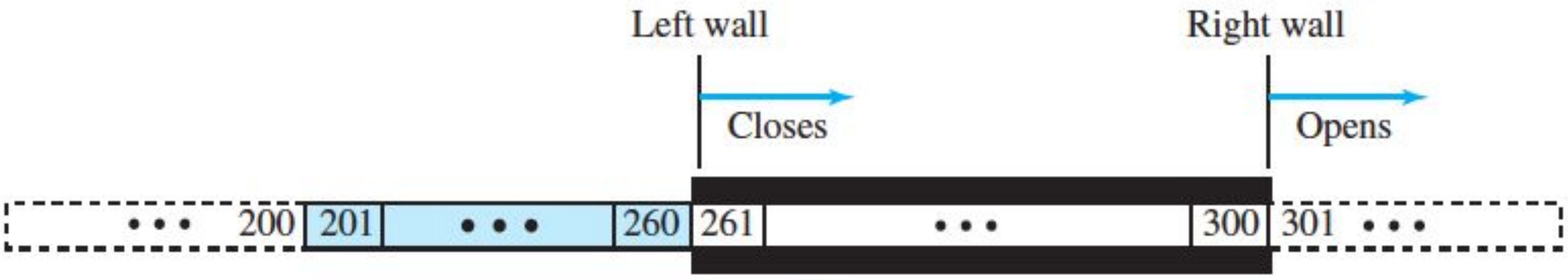


Receiver Sliding Window



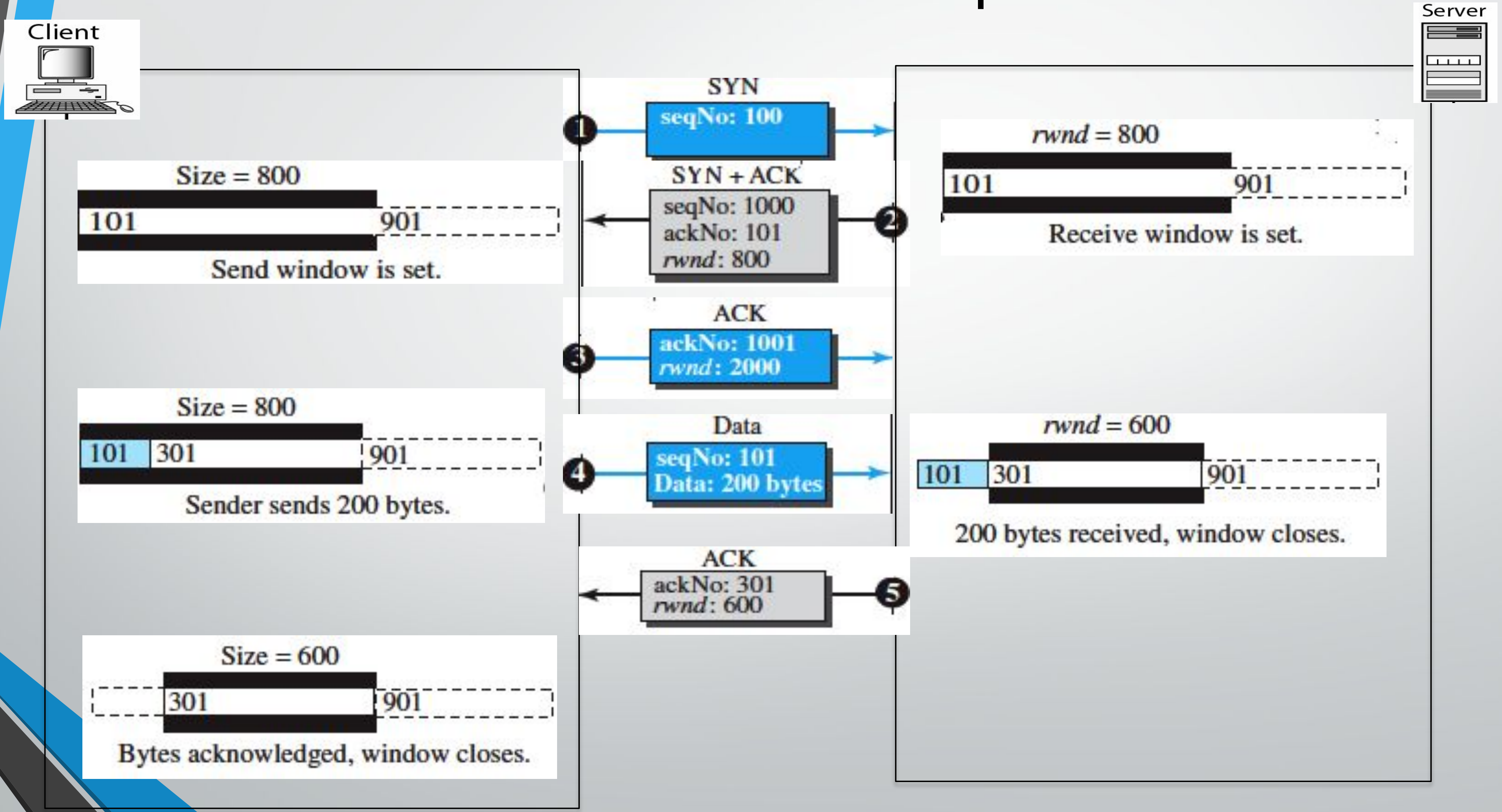
- rwnd= buffer size – number of bytes to be pulled = 40 bytes

Sliding of Receiver Window

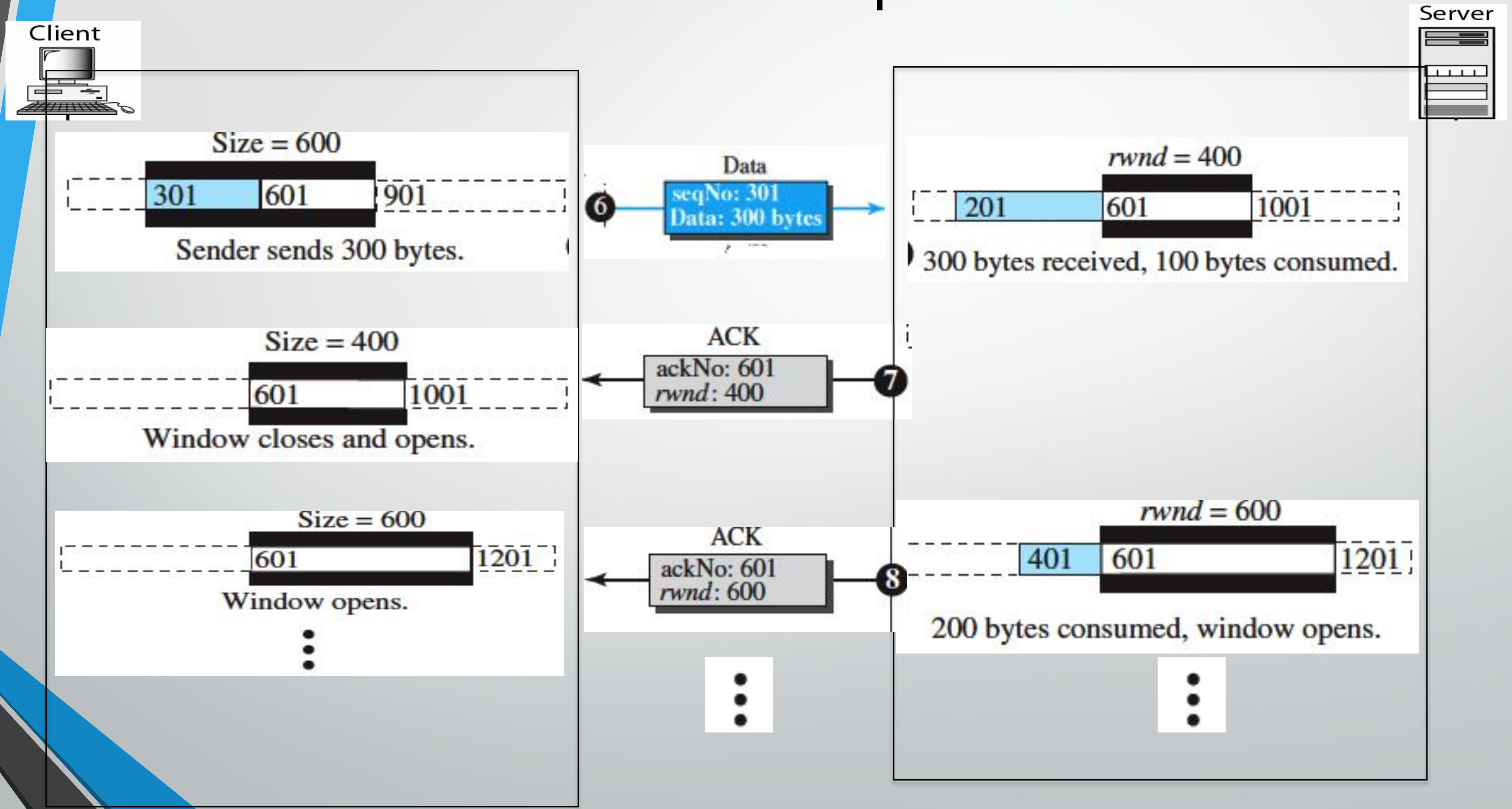


b. Opening and closing of receive window

Flow Control Example

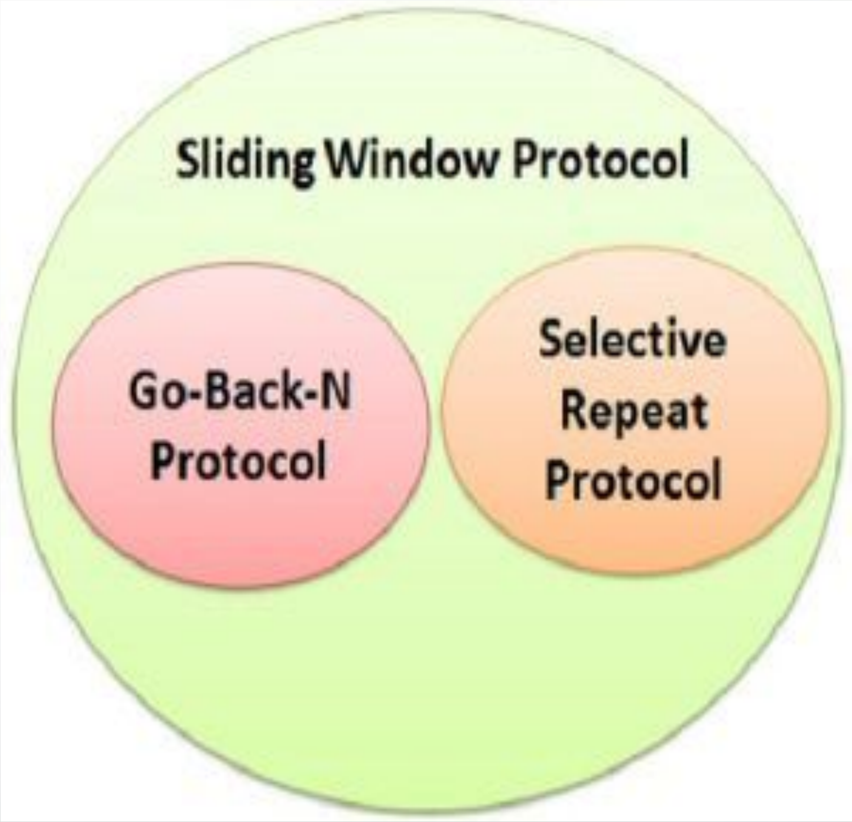


Flow Control Example Contd

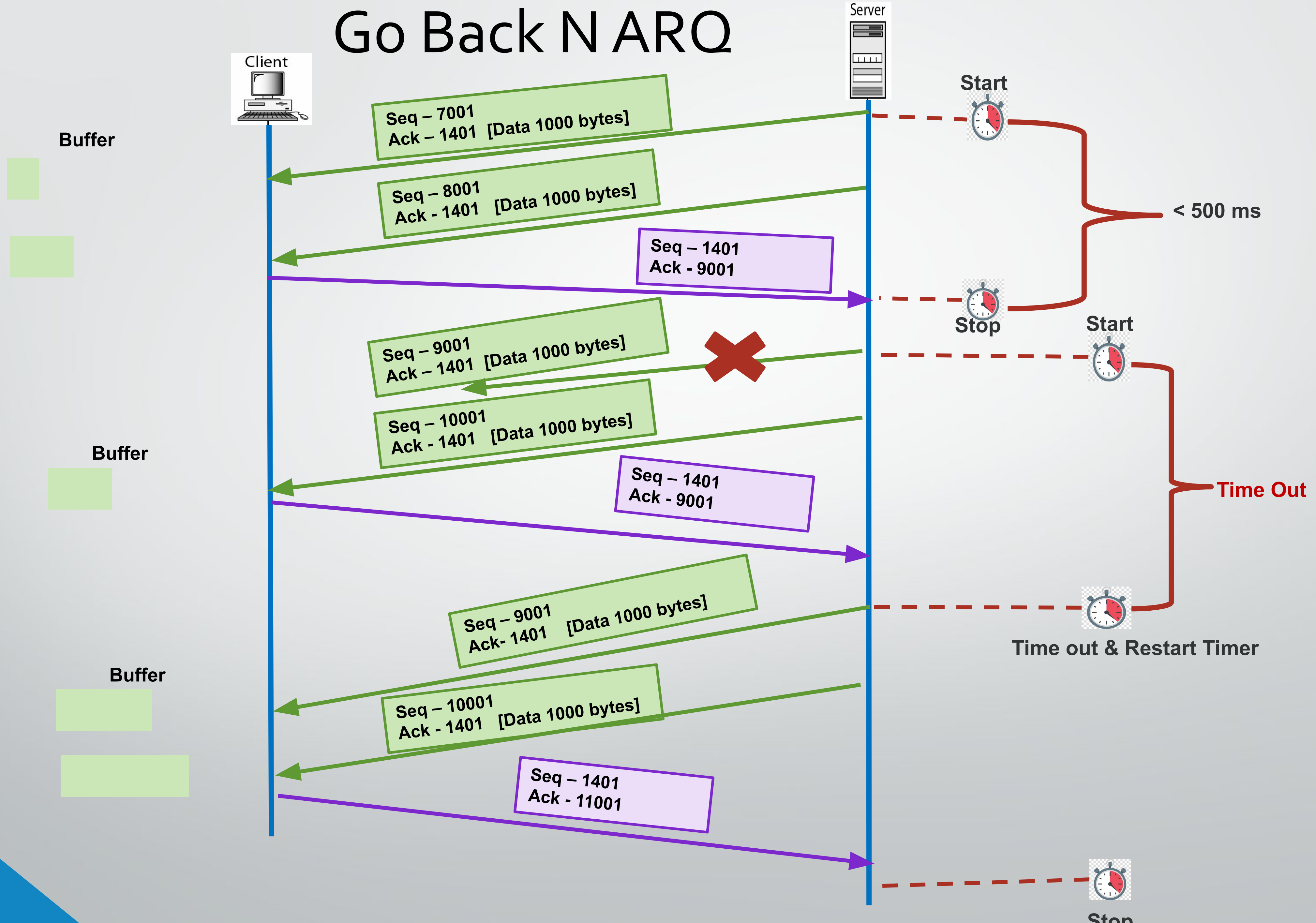


Different TCP Sliding Window Protocols

- **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



Go Back N ARQ



Client
Server
Buffer
Buffer
Buffer
Seq – 7001
Ack – 1401 [Data 1000 bytes]
Seq – 8001
Ack - 1401 [Data 1000 bytes]
Seq – 1401
Ack - 9001
Seq – 9001
Ack – 1401 [Data 1000 bytes]
Seq – 10001
Ack - 1401 [Data 1000 bytes]
Seq – 1401
Ack - 9001
Seq – 9001
Ack- 1401 [Data 1000 bytes]
Seq – 10001
Ack - 1401 [Data 1000 bytes]
Seq – 1401
Ack - 11001
Start
Stop
< 500 ms
Start
Time Out
Time out & Restart Timer

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

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