Computer Network

Lecture 8

Transport Layer Part.1

2019. 03. 01

Sungwon Lee

Department of Software Convergence

경희대학교
KYUNG HEE UNIVERSITY

Contents

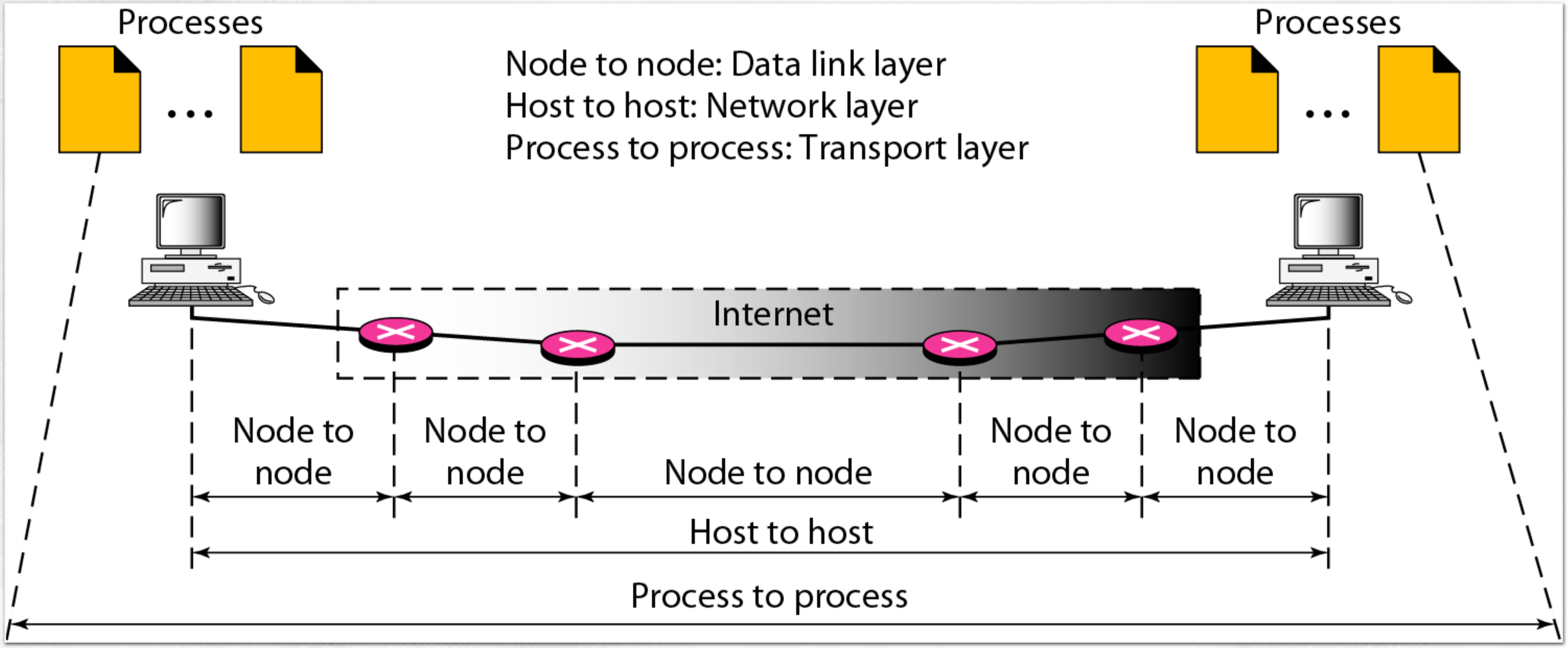
- **Transport Layer Basic**
- User Datagram Protocol (UDP)
- Transmission Control Protocol (TCP)

Process-to-Process Delivery

- The data link layer is responsible for delivery of frames between two neighboring nodes over a link. This is called node-to-node delivery.
- The network layer is responsible for delivery of datagram between two hosts. This is called host-to-host delivery.

Process-to-Process Delivery

- The transport layer is responsible for process- to-process delivery—the delivery of a packet, part of a message, from one process to another. Two processes communicate in a client/server relationship.



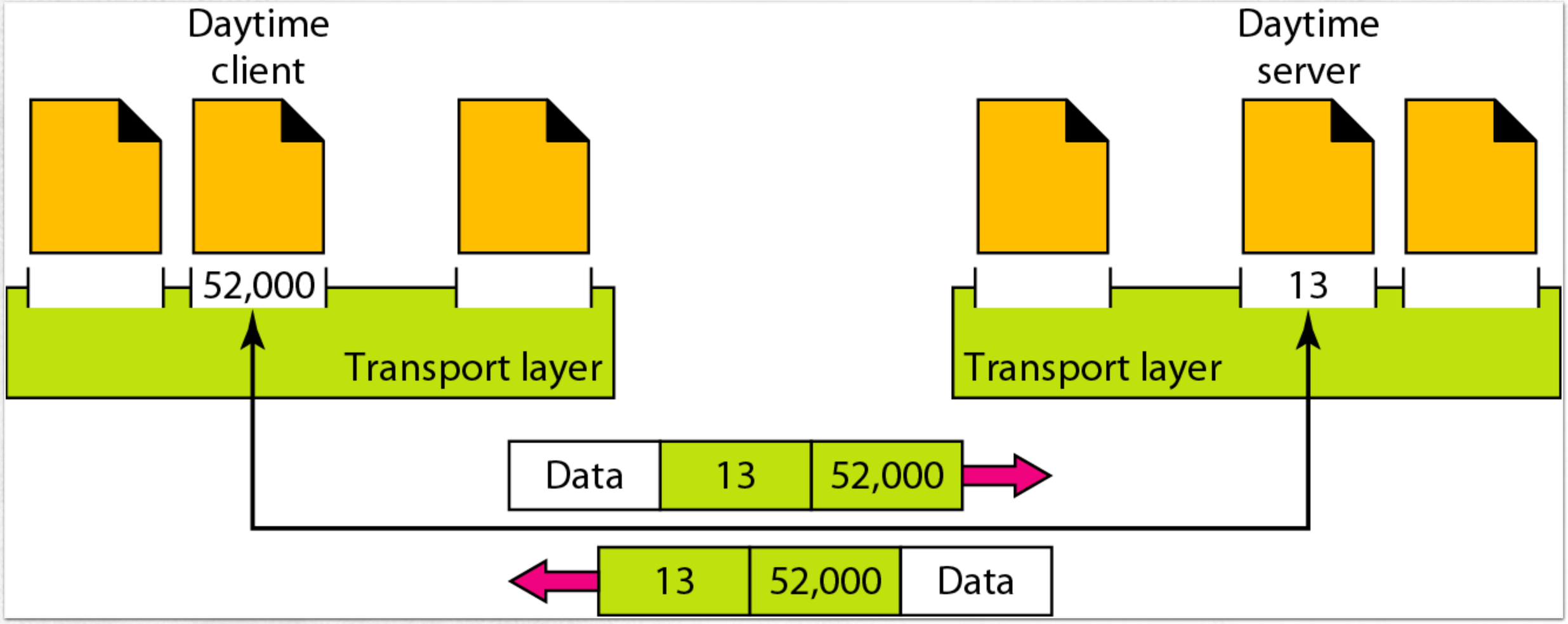
Client/Server Paradigm

- The most common process-to-process communication is though the Client/Server Paradigm.
  - Client : A process on the local host
  - Server : A process on the remote host to provide services.
- Both processes (Client and Server) have the same name.
  - Ex.) day time Client process and day time Server process.
- For communication, we must define the following;
  - Local host
  - Local process
  - Remote host
  - Remote process

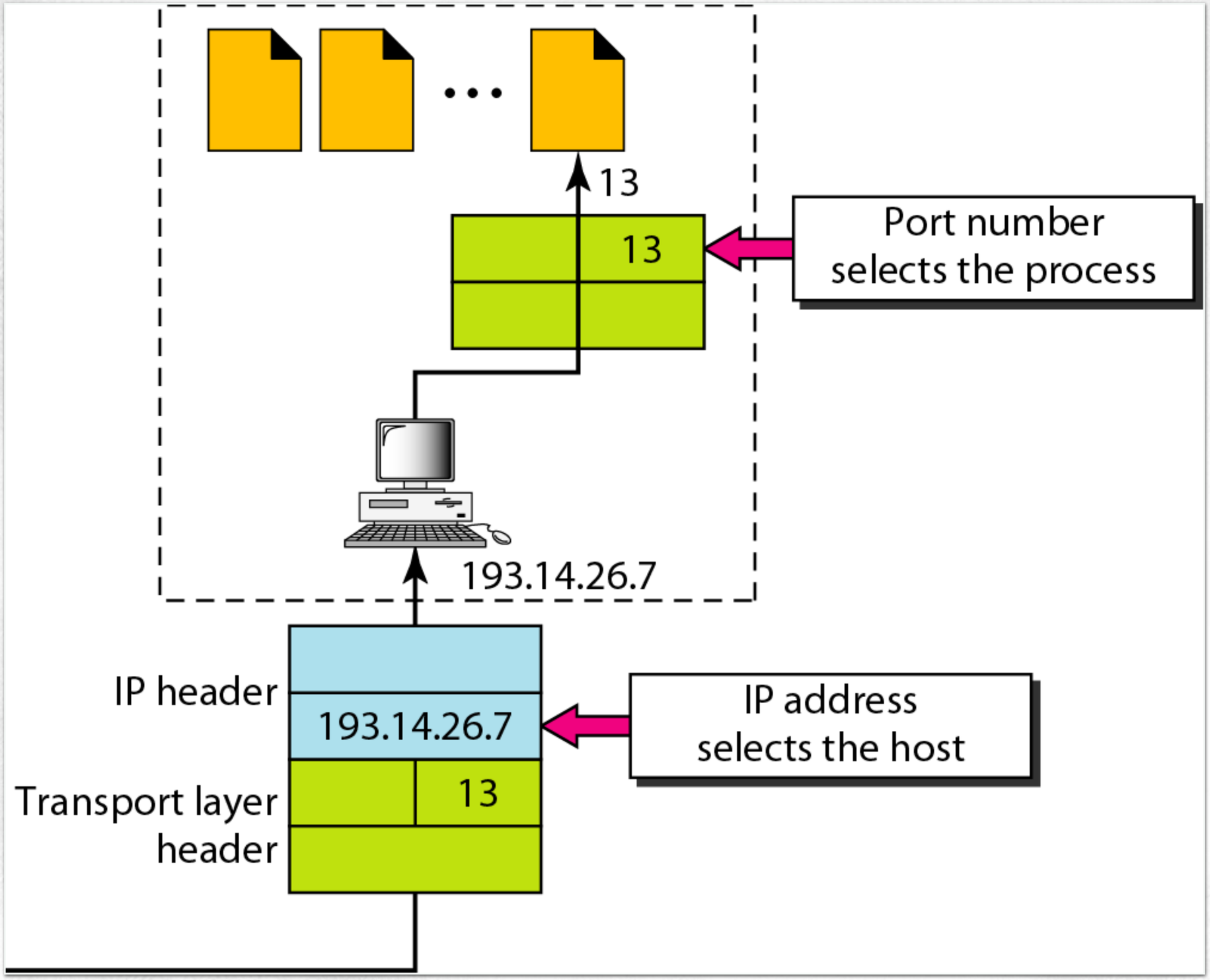
Addressing

- A frame in the data link layer needs a destination MAC address.
- At the network layer, we need an IP address.
- At the transport layer, we need a transport layer address, called a Port number to choose among multiple processes running on the host.
- Port number range : 0~65,535 (16-bit integer)
- Well-known port number : Universal port No. for server (fixed value, ranging : 0 ~1,023)
- Ephemeral port number : A port No. chosen randomly by the transport layer SW running on the client host.

Addressing

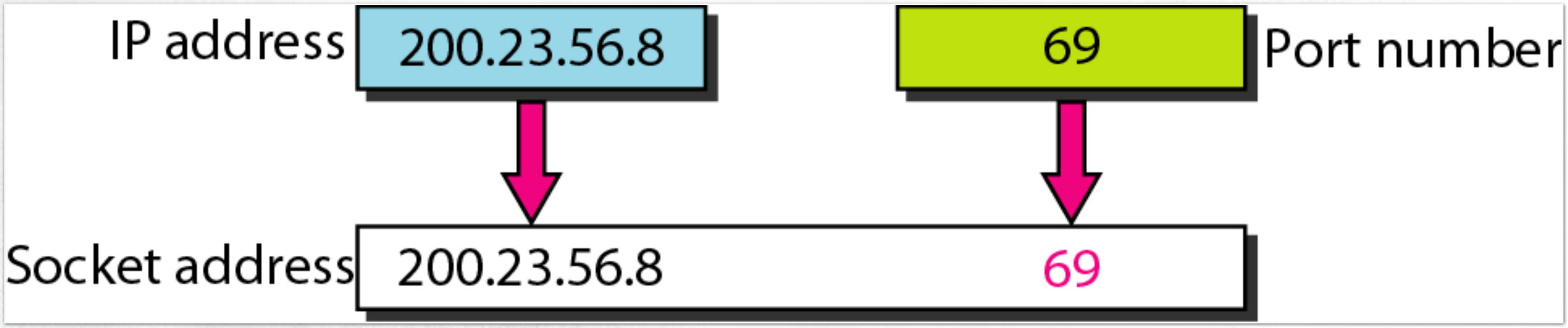

Daytime client
Daytime server
52,000
13
Transport layer
Transport layer
Data
13
52,000
13
52,000
Data

IP addresses Versus Port numbers

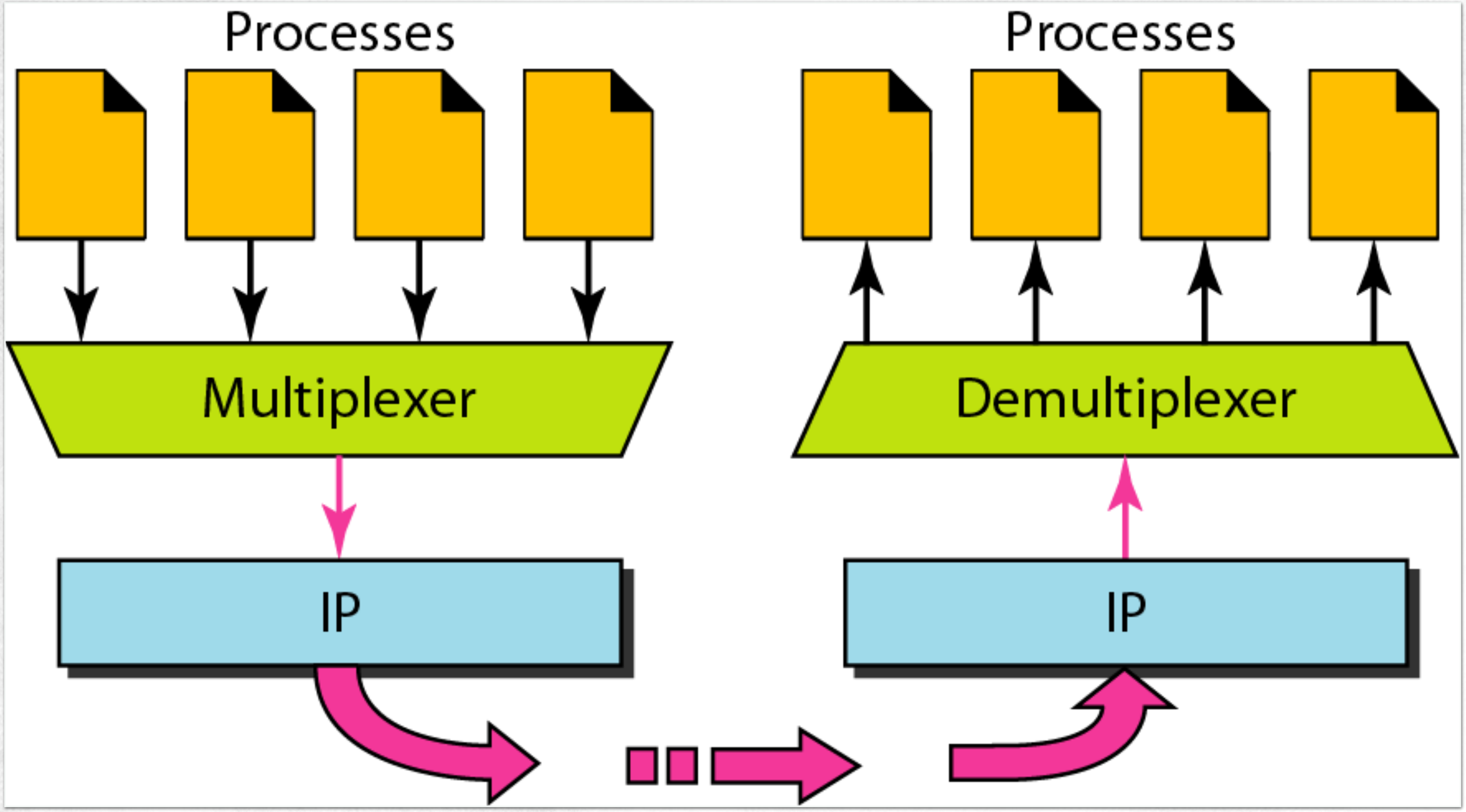


Socket Addresses

- The combination of an IP address and a port number.
- A transport layer protocol needs a pair of socket addresses: the client socket address and the server socket address.



Multiplexing and Demultiplexing



Connectionless Vs Connection-Oriented Service

- Connectionless Service
  - The packets are not numbered; they may be delayed or connection release.
  - There is no acknowledgment either.
  - UDP protocol is connectionless in the Internet model.
- Connection-Oriented Service
  - A connection is first established between the sender and the receiver.
  - Data are transferred.
  - At the end, the connection is released.
  - TCP and SCTP are connection-Oriented protocols.

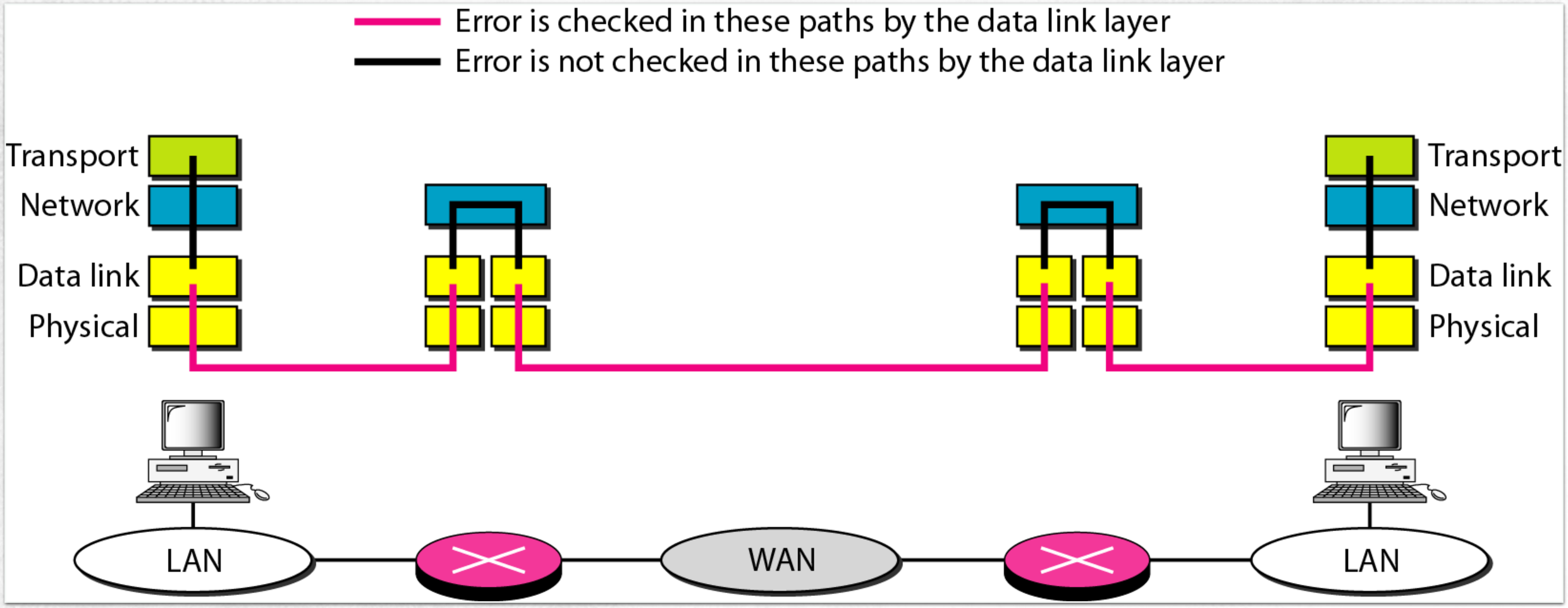
Reliable Versus Unreliable

- Reliable Service
  - If the application layer program needs reliability, we use a reliable transport protocol such as TCP and SCTP.
  - This means a slower and more complex service.
- Unreliable Service
  - If the application layer program does not need reliability because it uses its own flow and error control mechanism
  - or it needs fast service or the nature of the service does not demand flow and error control (real-time application), then unreliable protocol such as UDP can be used.

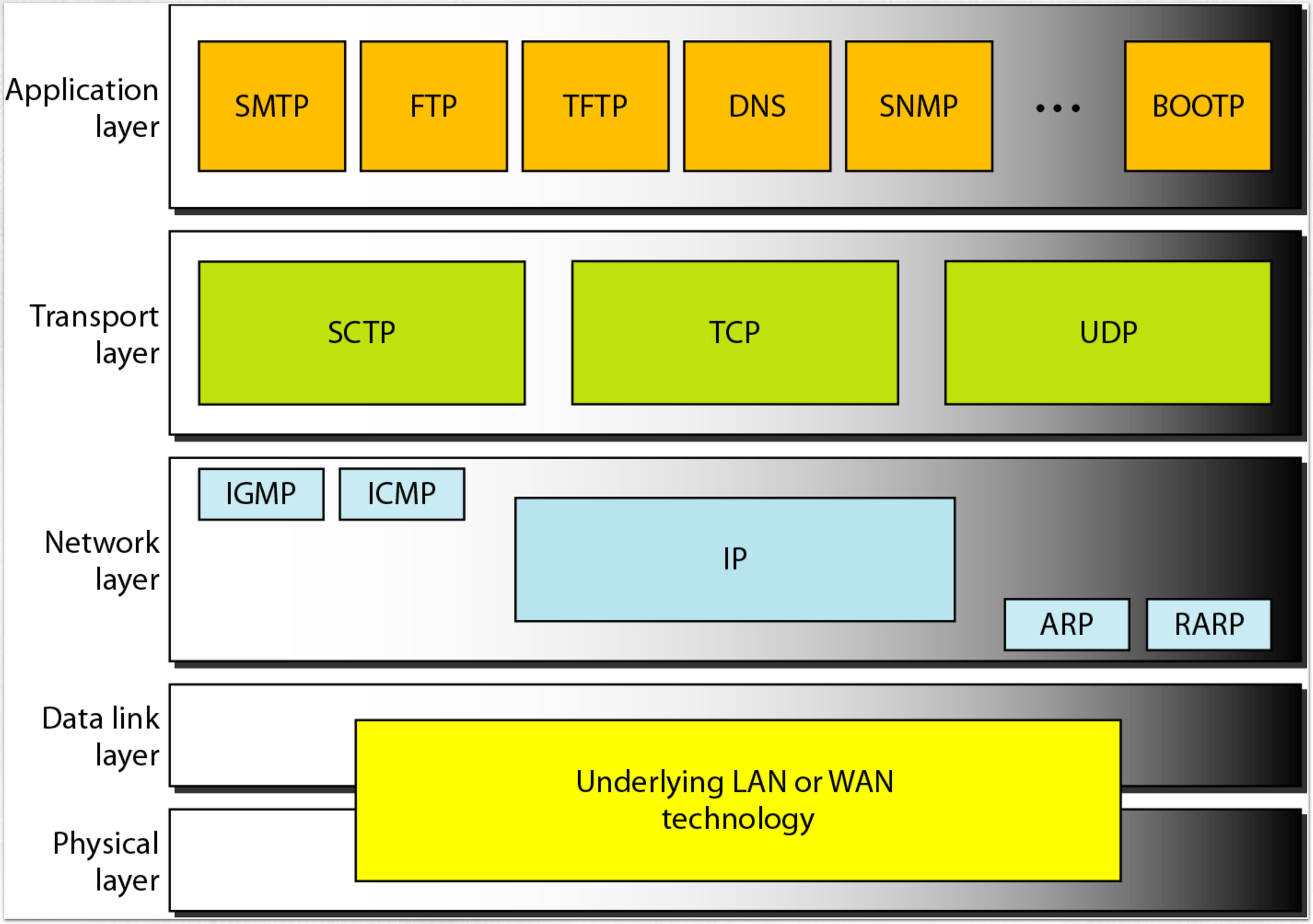
Reliable Versus Unreliable

- Do we need reliability control at the transport layer, even the data link layer is reliable and has flow and error control?
  - The answer is yes.
  - The network layer in the Internet is unreliable (best-effort delivery), we need to implement reliability at the transport layer.

Error control



Major Transport Layer Protocols



Contents

- Transport Layer Basic
- **User Datagram Protocol (UDP)**
- Transmission Control Protocol (TCP)

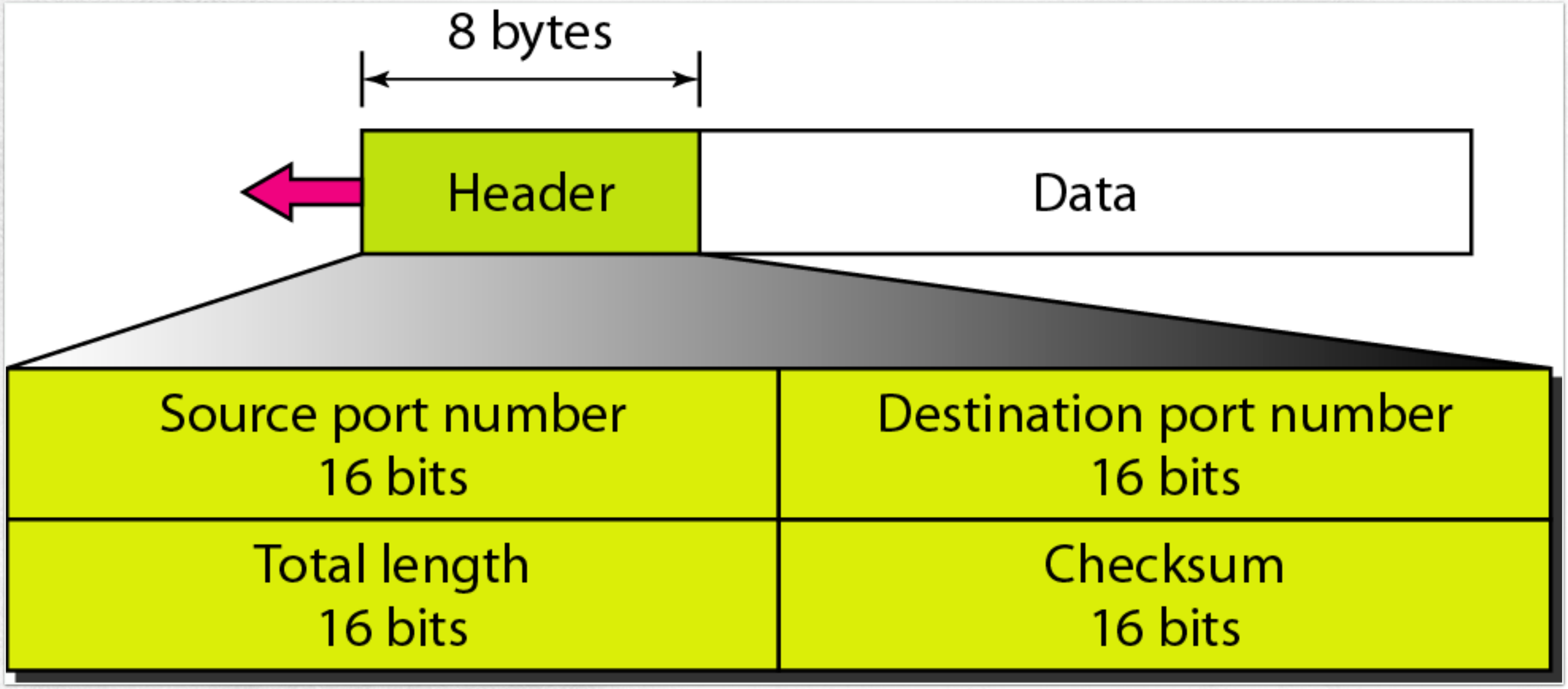
Concept

- 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.
- If UDP is so powerless, why would a process want to use it? Very simple protocol using a minimum of overhead
  - if a process wants to send a small message and does not care much about reliability, it can use UDP
  - if it sends a small message, taking much less interaction between the sender and receiver than it does using TCP

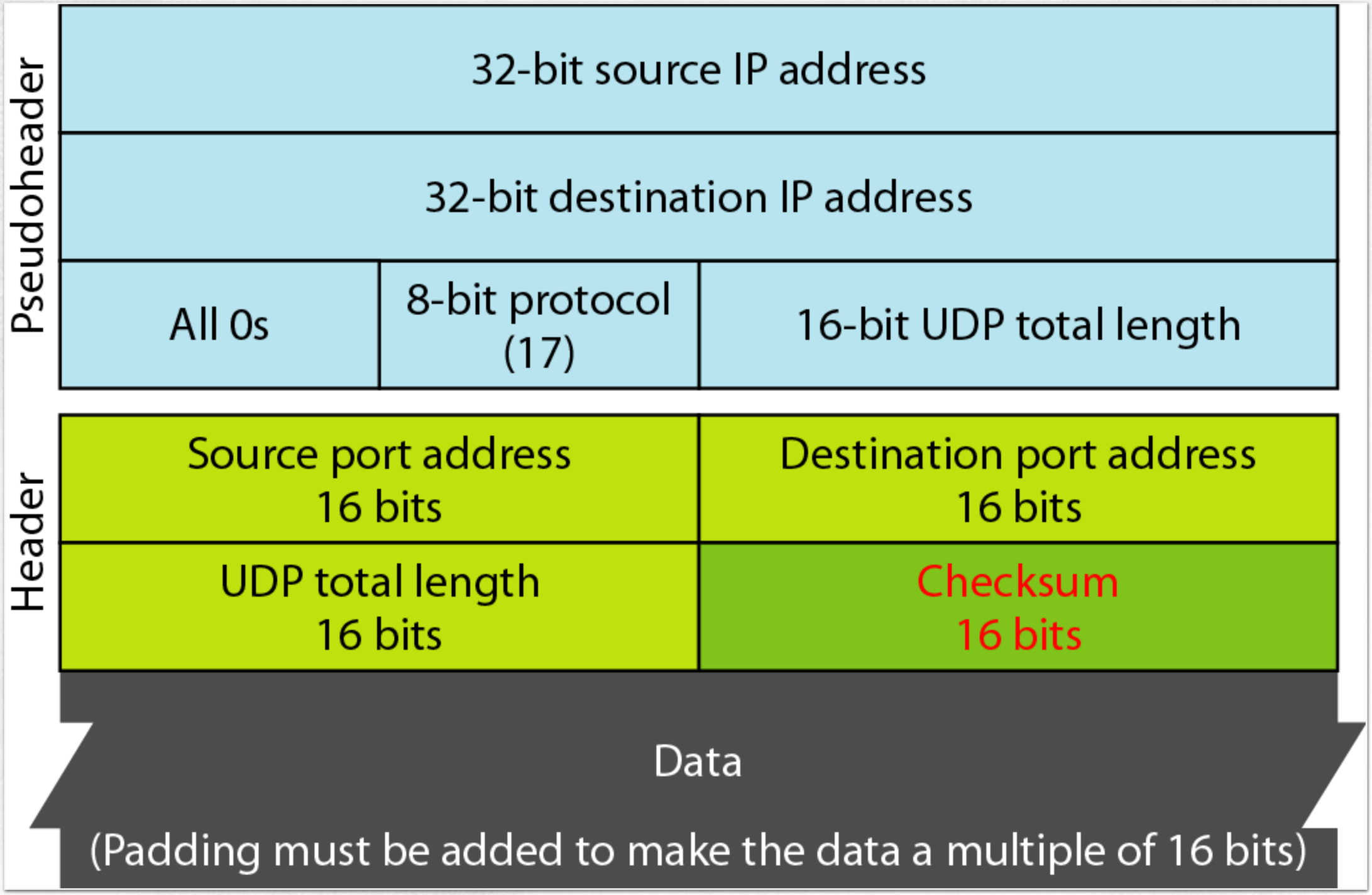
Well-known ports used with UDP

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)

Frame format



Frame format



Operation

- Connectionless Services :
  - UDP provides a connectionless services that each datagram sent by UDP is an independent datagram.
- Flow and Error Control :
  - There is no flow control and hence no window mechanism.
  - There is no error control mechanism in UDP except for the checksum.
- Encapsulation and Decapsulation :
  - The UDP protocol encapsulates and decapsulates messages in an IP datagram.

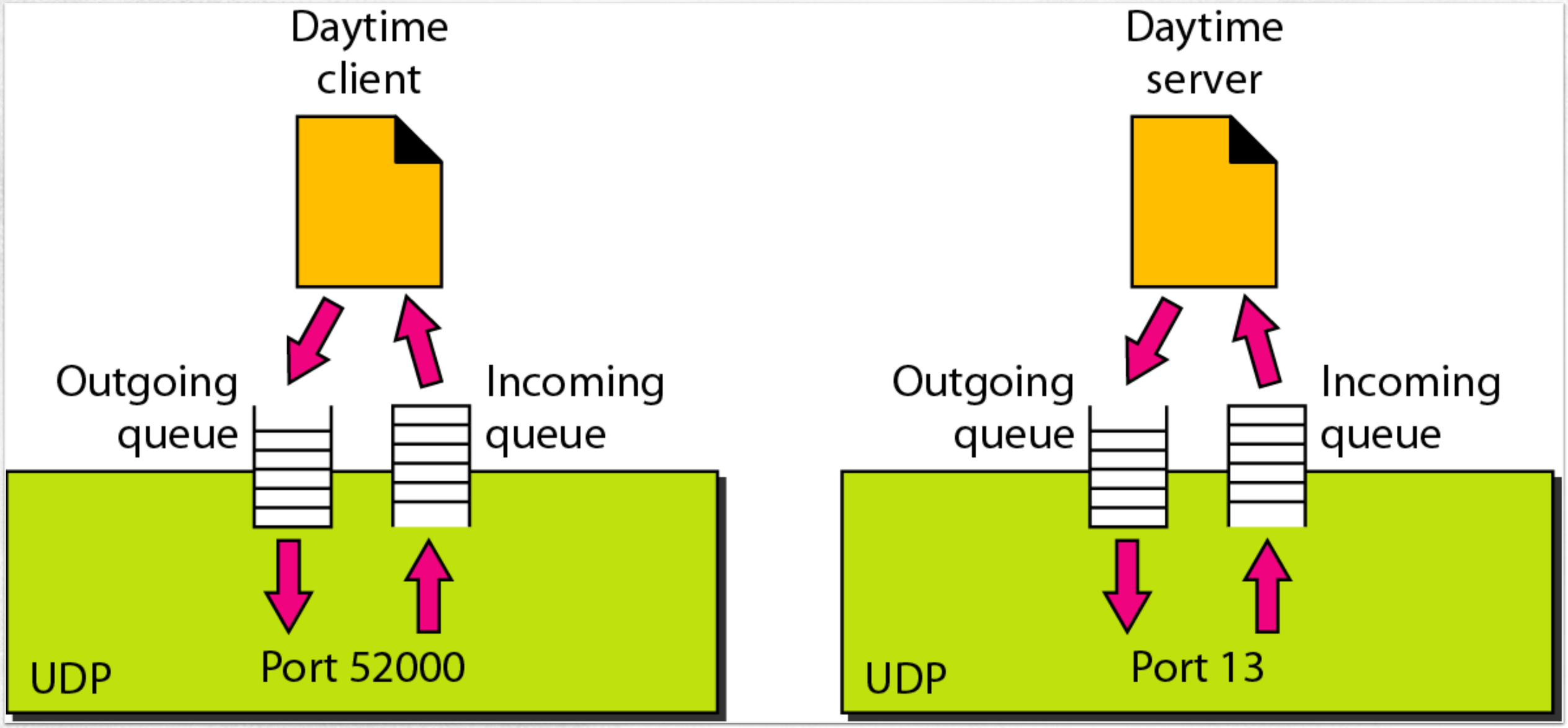
Queuing at client site

- When a process starts, it requests a port No. from the operating system.
- The client process sends messages to the outgoing queue by using the source port No. specified in the request.
- UDP removes the messages one by one and, after adding the UDP header, delivers them to IP.
- An outgoing queue can overflow. If this happens, the OS can ask the client process to wait before sending any more messages.
- When a message arrives for a client, UDP checks to see if an incoming queue has been created for the port No. specified in the destination port No. field of the user datagram. If there is a queue, UDP sends the received user datagram to the end of the queue.

Queuing at server site

- When it starts running, a server asks for incoming and outgoing queues, using its well-known port.
- When a message arrives for a server,
- UDP checks to see if an incoming queue has been created for the port number specified in the destination port number field of the user datagram.
- If there is a queue, UDP sends the received user datagram to the end of the queue.
- When a server wants to respond to a client, it sends messages to the outgoing queue, using the source port No. specified in the request.
- UDP removes the messages one by one and, after adding the UDP header, delivers them to IP.

Queuing in UDP



Applications of UDP

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.
  - Not used for sending bulk data, such as FTP
  - TFTP including internal flow and error control
  - Suitable transport protocol for multicasting
  - Used for some route updating protocols such as Routing Information Protocol (RIP)
  - Used for management processes such as SNMP.

Contents

- Transport Layer Basic
- User Datagram Protocol (UDP)
- Transmission Control Protocol (TCP)

Concept

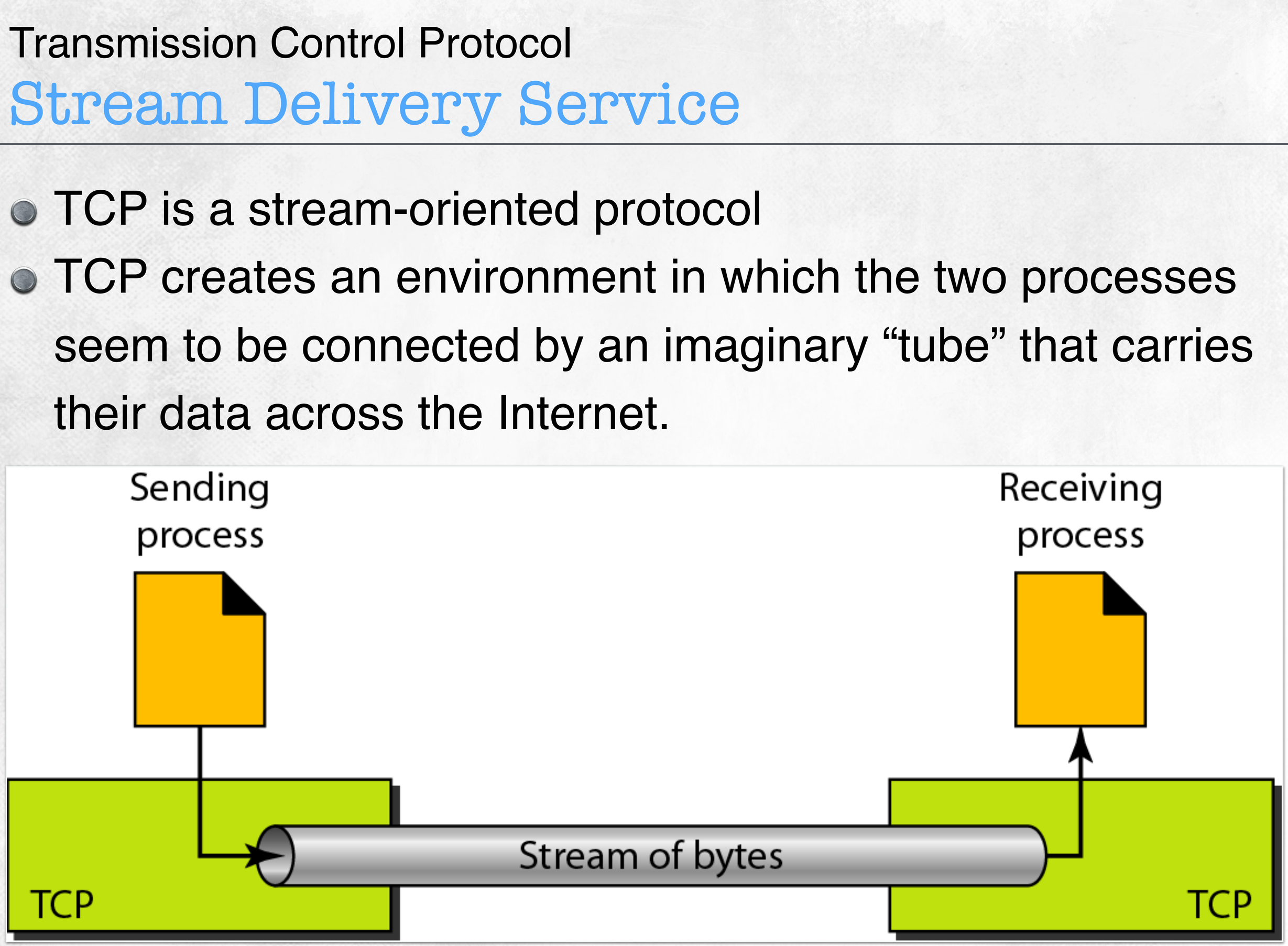
- TCP is a connection-oriented protocol; it creates a virtual connection between two TCPs to send data. In addition, TCP uses flow and error control mechanisms at the transport level.

TCP Services

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

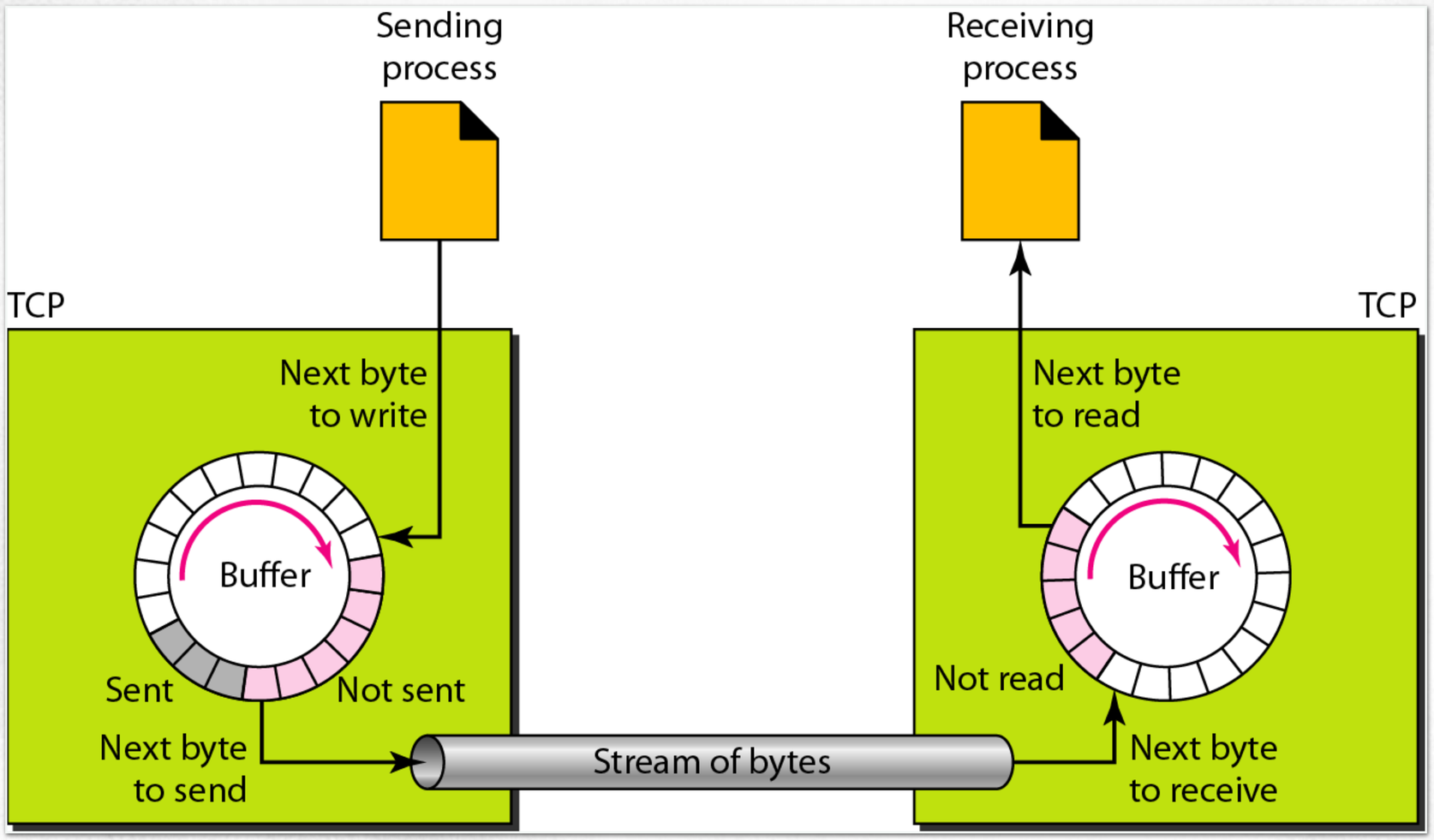
Stream Delivery Service

- TCP is a stream-oriented protocol
- TCP creates an environment in which the two processes seem to be connected by an imaginary “tube” that carries their data across the Internet.



Sending and Receiving Buffers

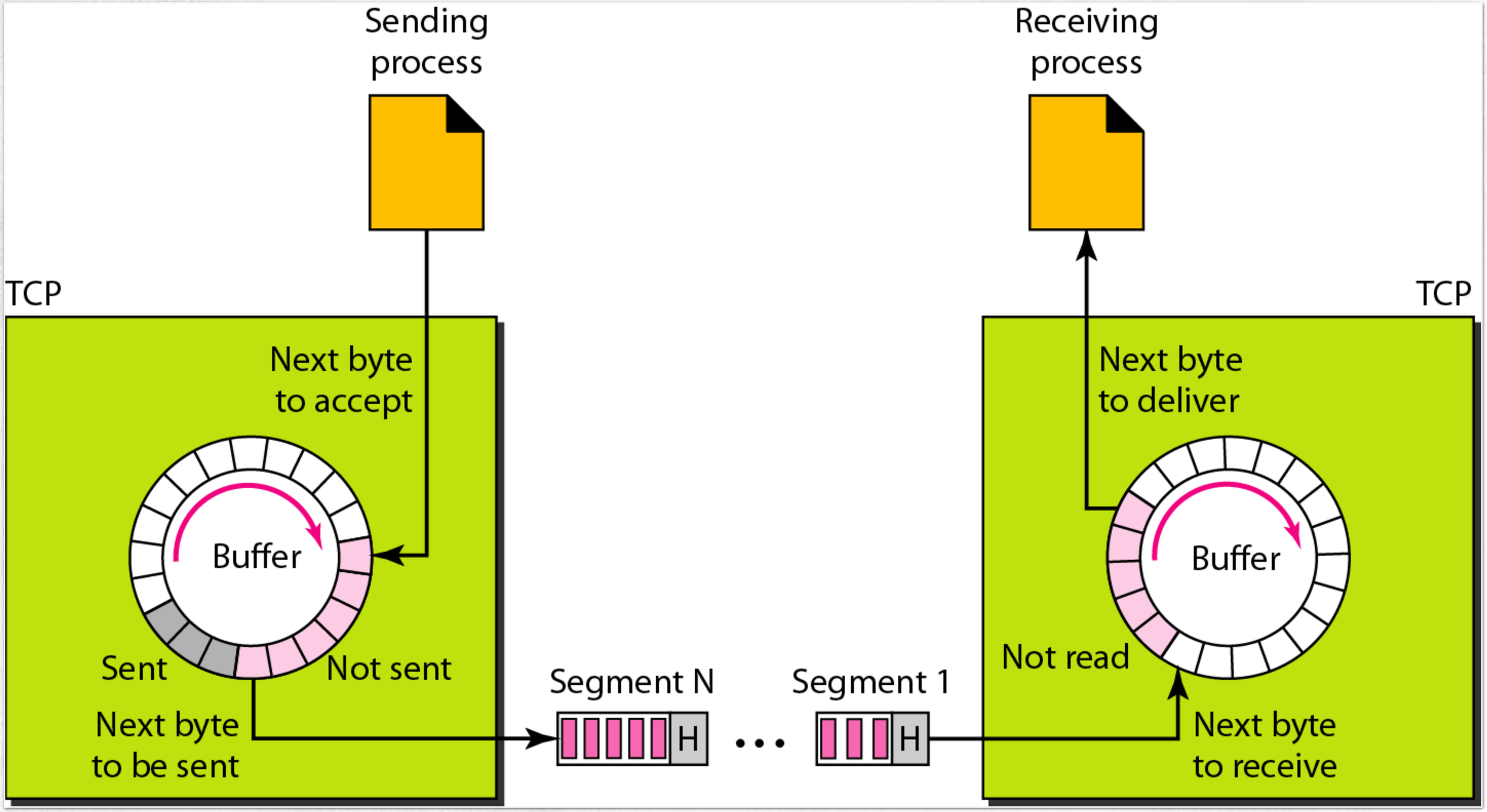
- Because the sending and receiving processes may not produce and consume data at the same speed, TCP needs buffers for storage.
- One way to implement is to use a circular array



Segments

- At the transport layer, TCP groups a number of bytes together into a packet called a segment.
- TCP adds a header to each segment (for control purposes) and delivers the segment to the IP layer for transmission.
- The segments are encapsulated in IP datagrams and transmitted.
- This entire operation is transparent to the receiving process.
- Segments received out of order, lost, or corrupted may be resent.

Segments


Sending process
Receiving process
TCP
TCP
Next byte to accept
Next byte to deliver
Buffer
Buffer
Sent
Not sent
Not read
Segment N
Segment 1
H
H
Next byte to be sent
Next byte to receive

Full-Duplex Service

- TCP offers full-duplex service : After two application programs are connected to each other, they can both send and receive data.
- Piggybacking : When a packet is going from A to B, it can also carry an acknowledgment of the packets received from B

Full-Duplex Service

- Connection-Oriented Services
  - A's TCP informs B's TCP and gets approval from B's TCP
  - A's TCP and B's TCP exchange data in both directions
  - After both processes have no data left to send and the buffers are empty, two TCPs destroy their buffers
- Reliable Service
  - TCP uses the acknowledgment mechanism to check the safe and sound arrival of data

Byte numbers

- There is no field for a segment number value. Instead, there are two fields called the sequence No. and the acknowledgment No. These two fields refer to the byte No.
- All data bytes being transferred in each connection are numbered by TCP.
- The numbering starts with a randomly generated number.
- Number range for first byte : 0 ~ 2 32 -1 (If random number is 1,057 and total number 6,000bytes, the bytes are numbered from 1,057 to 7,056)
- Byte numbering is used for flow and error control.

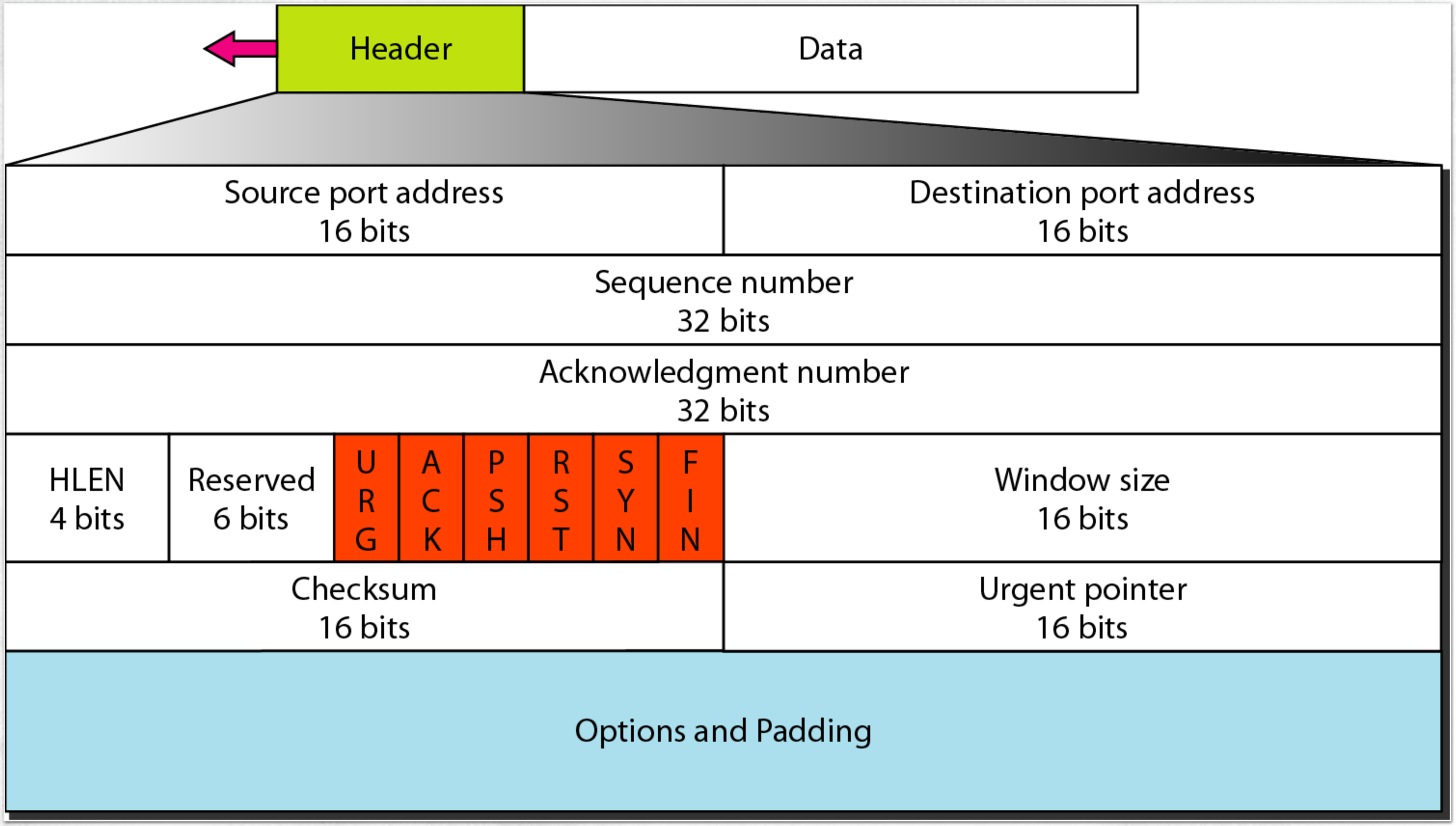
Sequence number

- After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent.
- Segment number for each segment is number of the first byte carried in that segment.

Acknowledgment number

- 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.

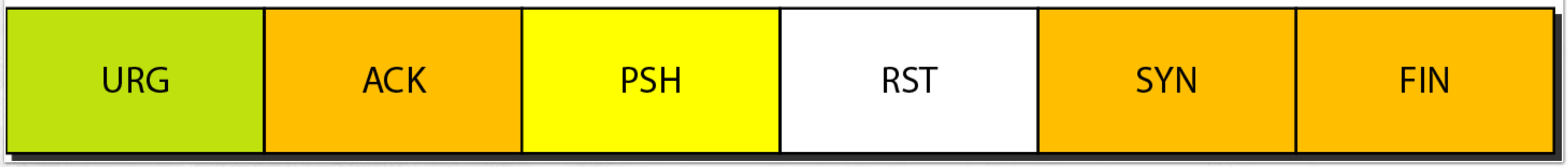
TCP Segment format



Control field

- Enabling flow control, connection establishment and termination, and mode of data transfer in TCP

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



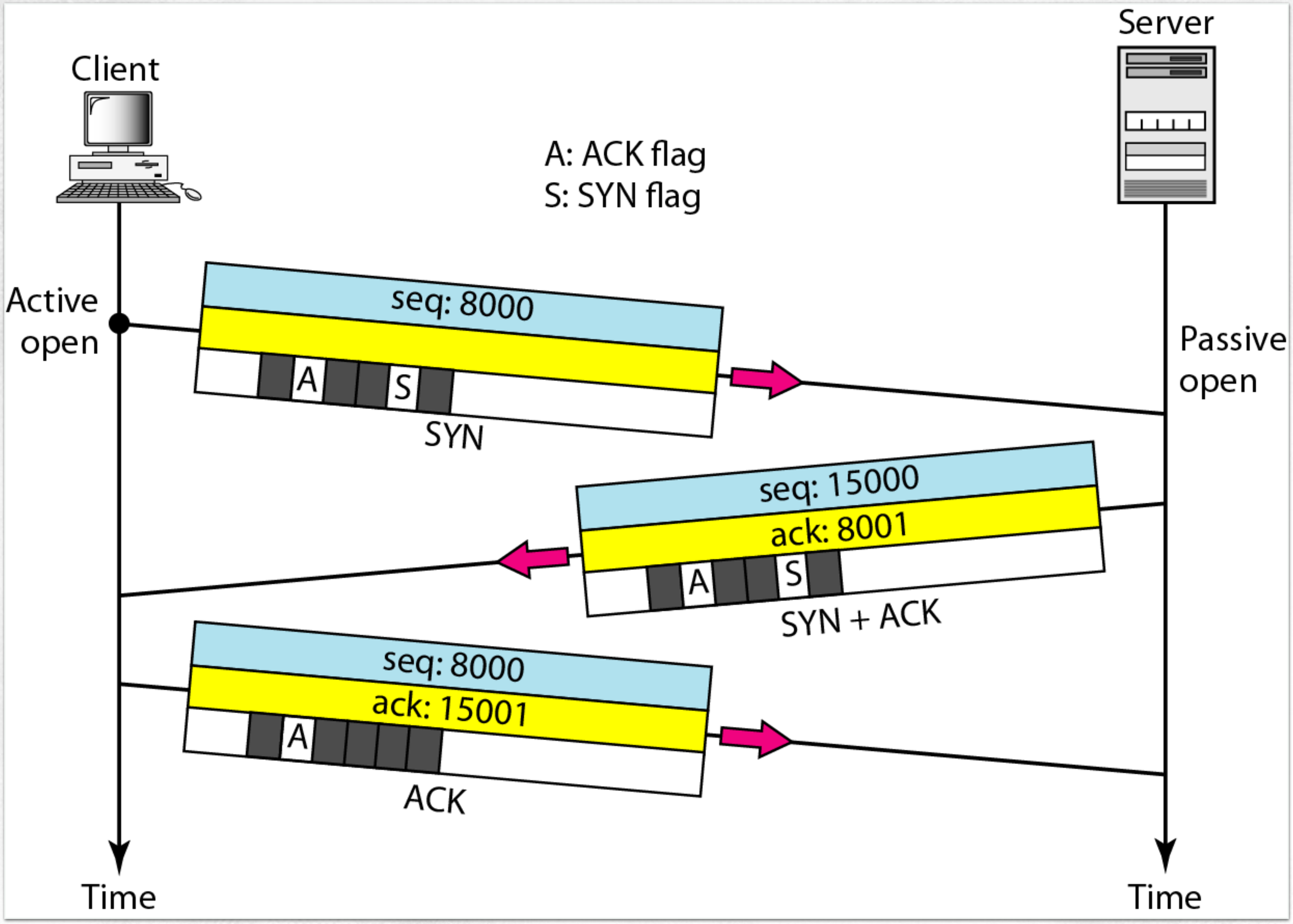
TCP is connection-oriented

- A connection-oriented transport protocol establishes a virtual path between the source and destination.
- All the segments belonging to a message are then sent over this virtual path.
- Using a single virtual pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames.
- You may wonder how TCP, which uses the services of IP, a connectionless protocol, can be connection-oriented.
- The point is that a TCP connection is virtual, not physical.
- TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself.

Connection Establishment

- TCP transmits data in full-duplex mode.
- When two TCPs in two machines are connected, they are able to send segments to each other simultaneously.
- This implies that each party must initialize communication and get approval from the other party before any data are transferred.

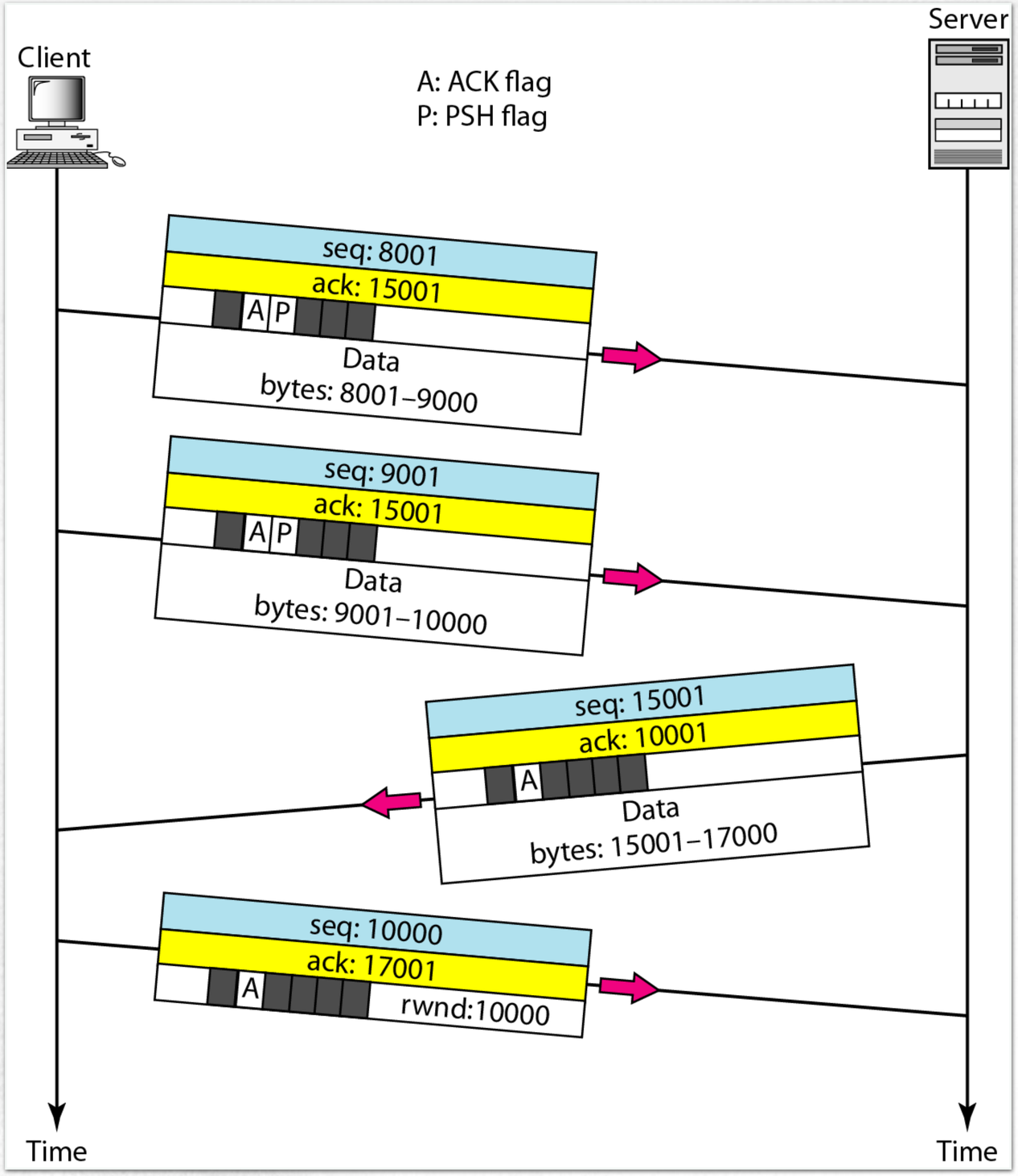
Connection Establishment



Connection Establishment

- Simultaneous Open
  - Simultaneous Open may occur when both processes issue an active open.
  - In this case, both TCPs transmit a SYN+ACK segment to each other, and one single-connection is established between them.
- SYN Flooding Attack
  - SYN Flooding Attack happens when a malicious attacker sends a large number of SYN segments to a server, pretending that each of them is coming from a different client by faking the source IP addresses in the data-grams.
  - The SYN flooding attack belongs to a type of security attack known as a denial-of-service attack, in which an attacker monopolizes a system with so many service requests that the system collapses and denies service to every request.

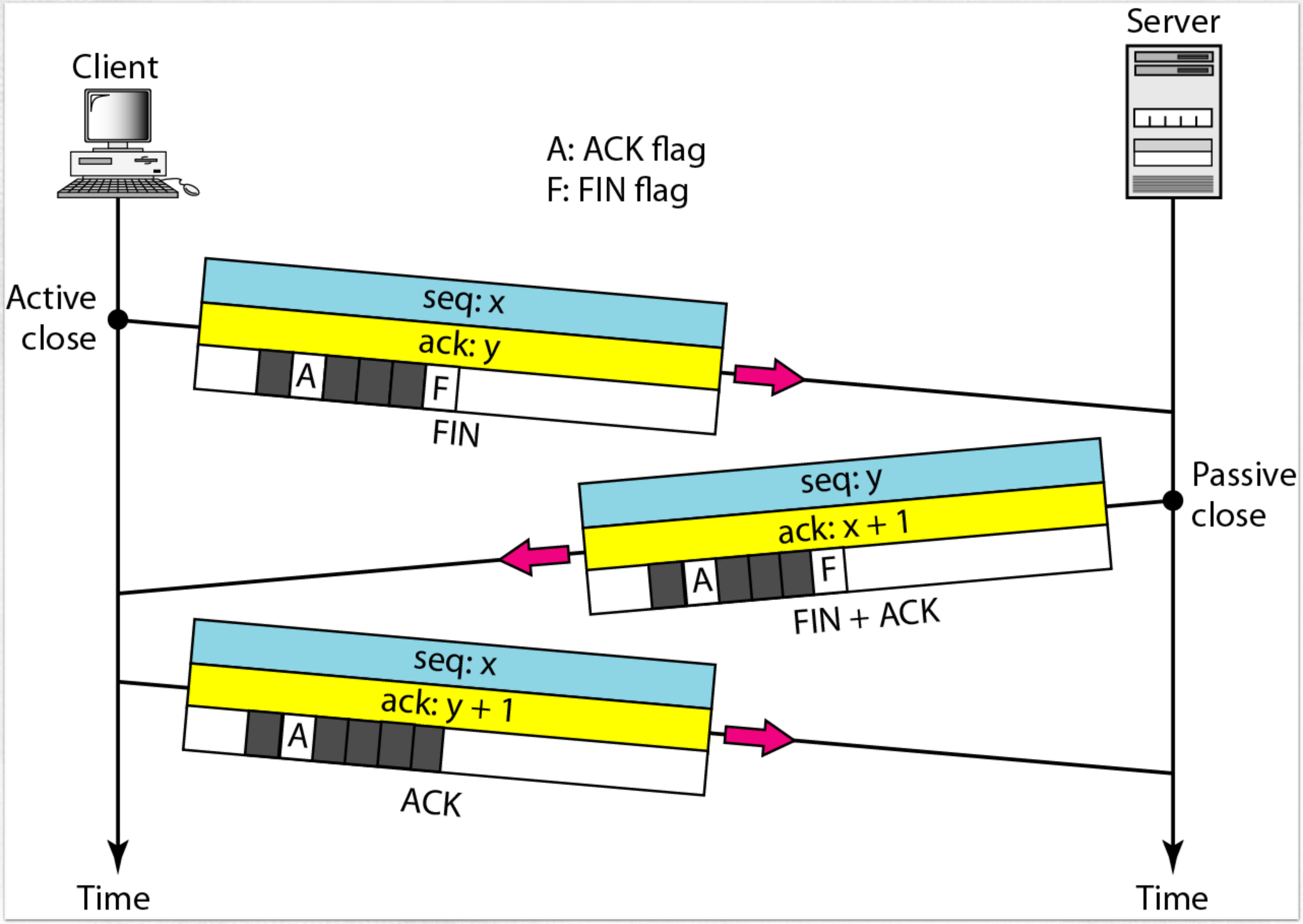
Data transfer



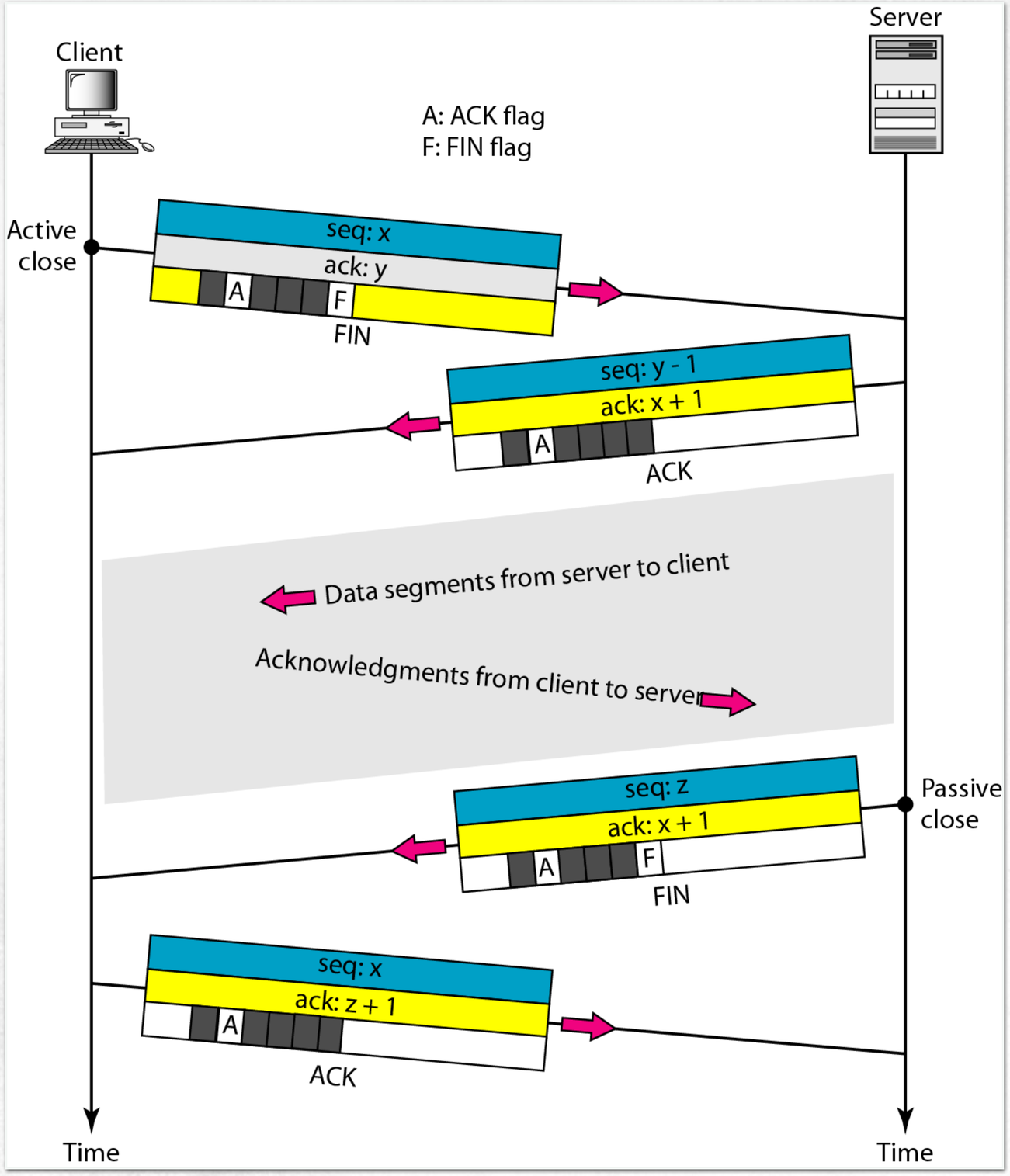
Data transfer

- Pushing Data
  - The application program at the sending site can request a push operation that the sending TCP must not wait for the window to be filled.
  - It must create a segment and send it immediately.
- Urgent Data
  - When the sending application program wants a piece of data to be read out of order by the receiving application program.
  - Sender can send a segment with the URG bit set.
  - When the receiving TCP receives a segment with the URG bit set, it extracts the urgent data from the segment, using the value of the pointer, and delivers them, out of order, to the receiving application program.

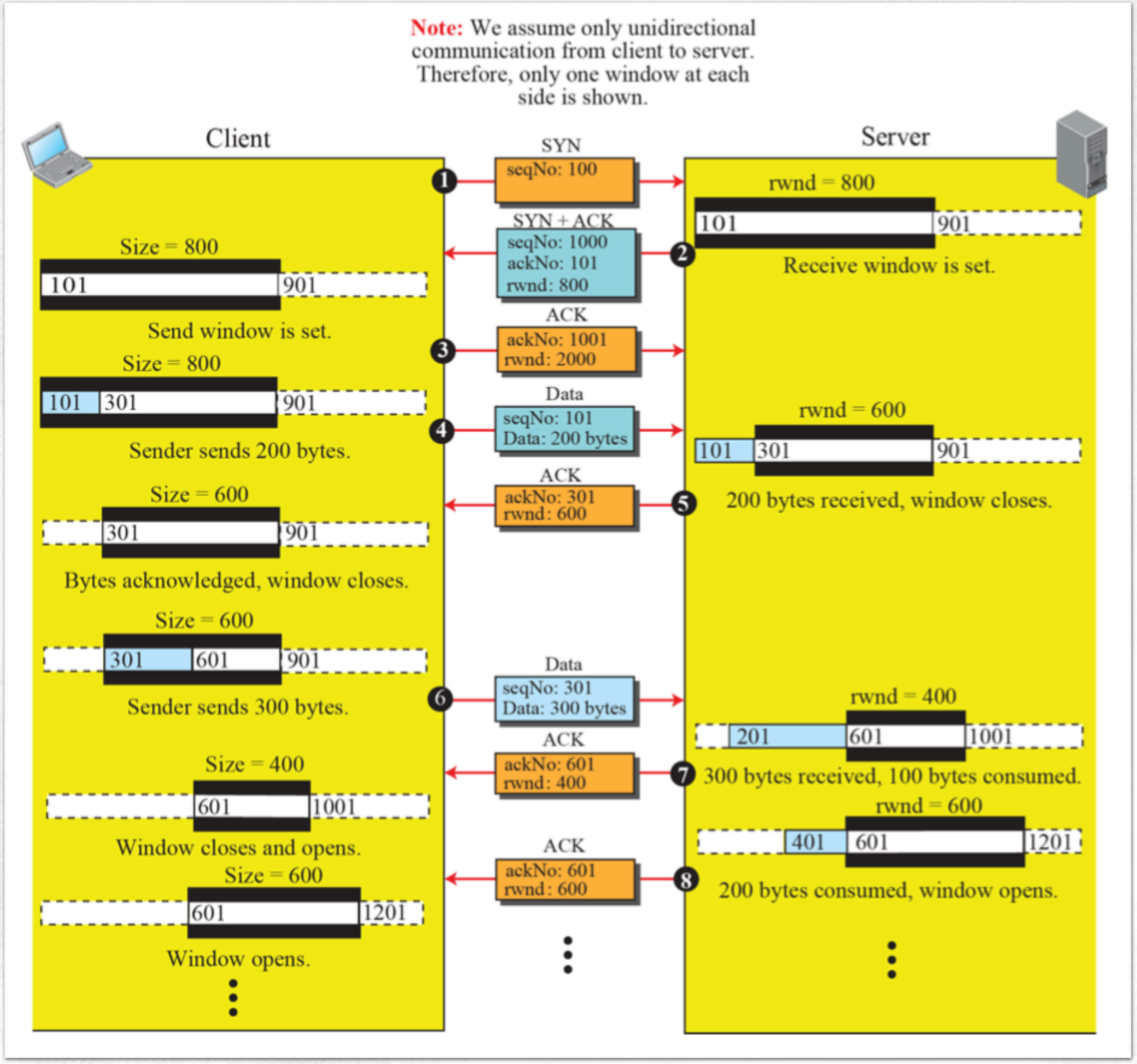
Connection termination



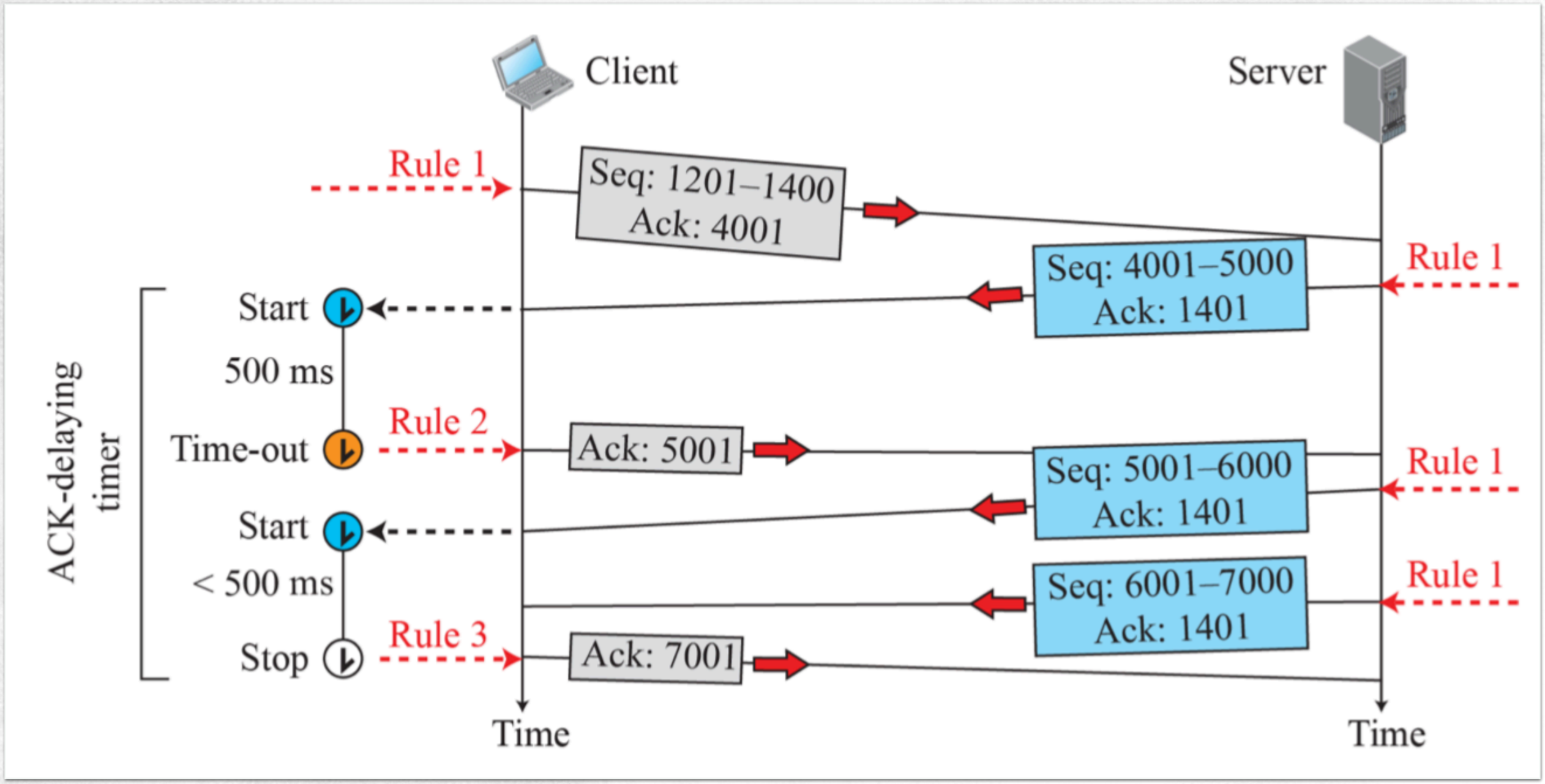
Half close


Client
Server
A: ACK flag
F: FIN flag
Active close
seq: x
ack: y
A
F
FIN
seq: y - 1
ack: x + 1
A
ACK
Data segments from server to client
Acknowledgments from client to server
seq: z
ack: x + 1
A
F
FIN
Passive close
seq: x
ack: z + 1
A
ACK
Time
Time

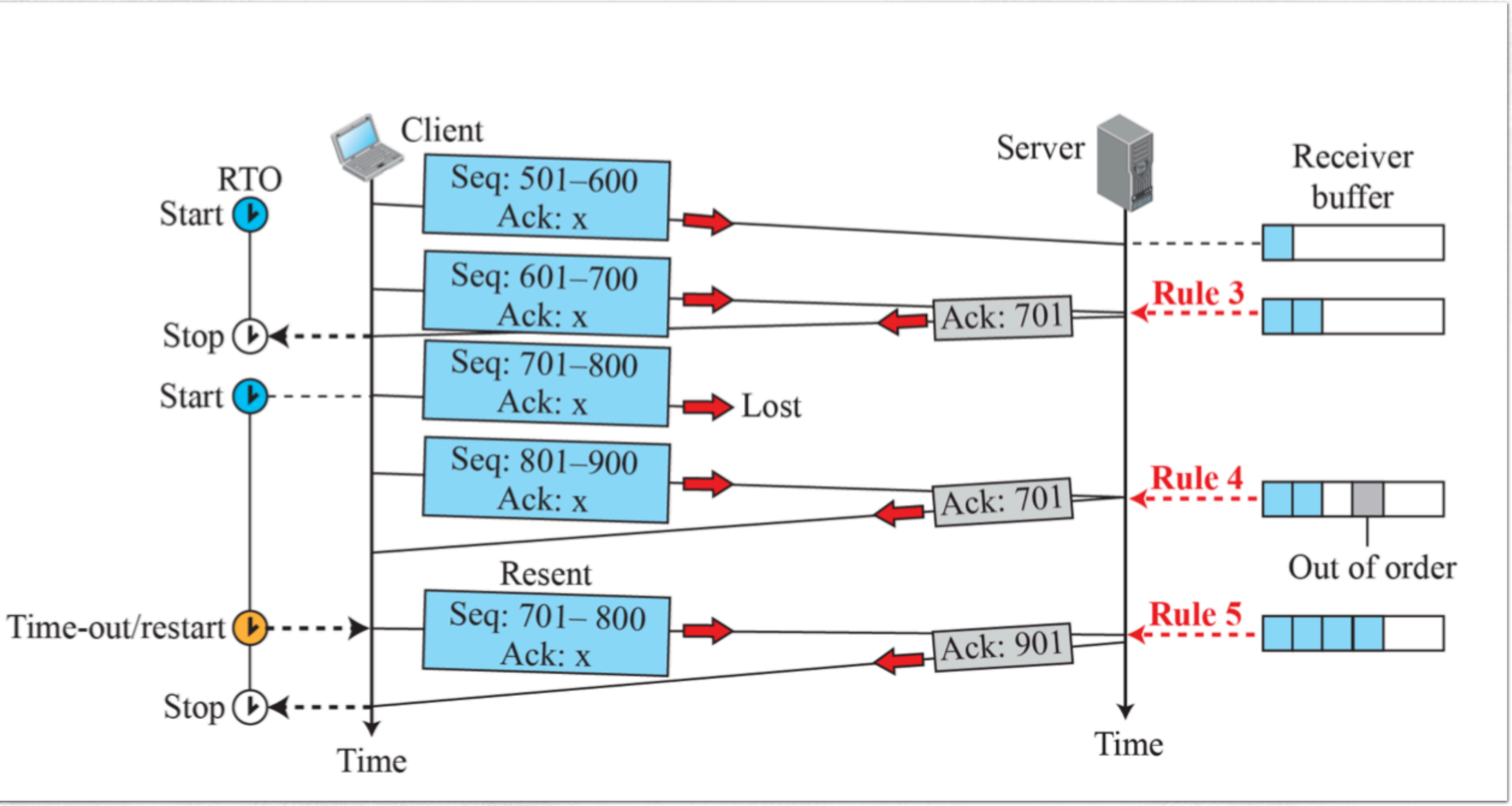
An example of flow control



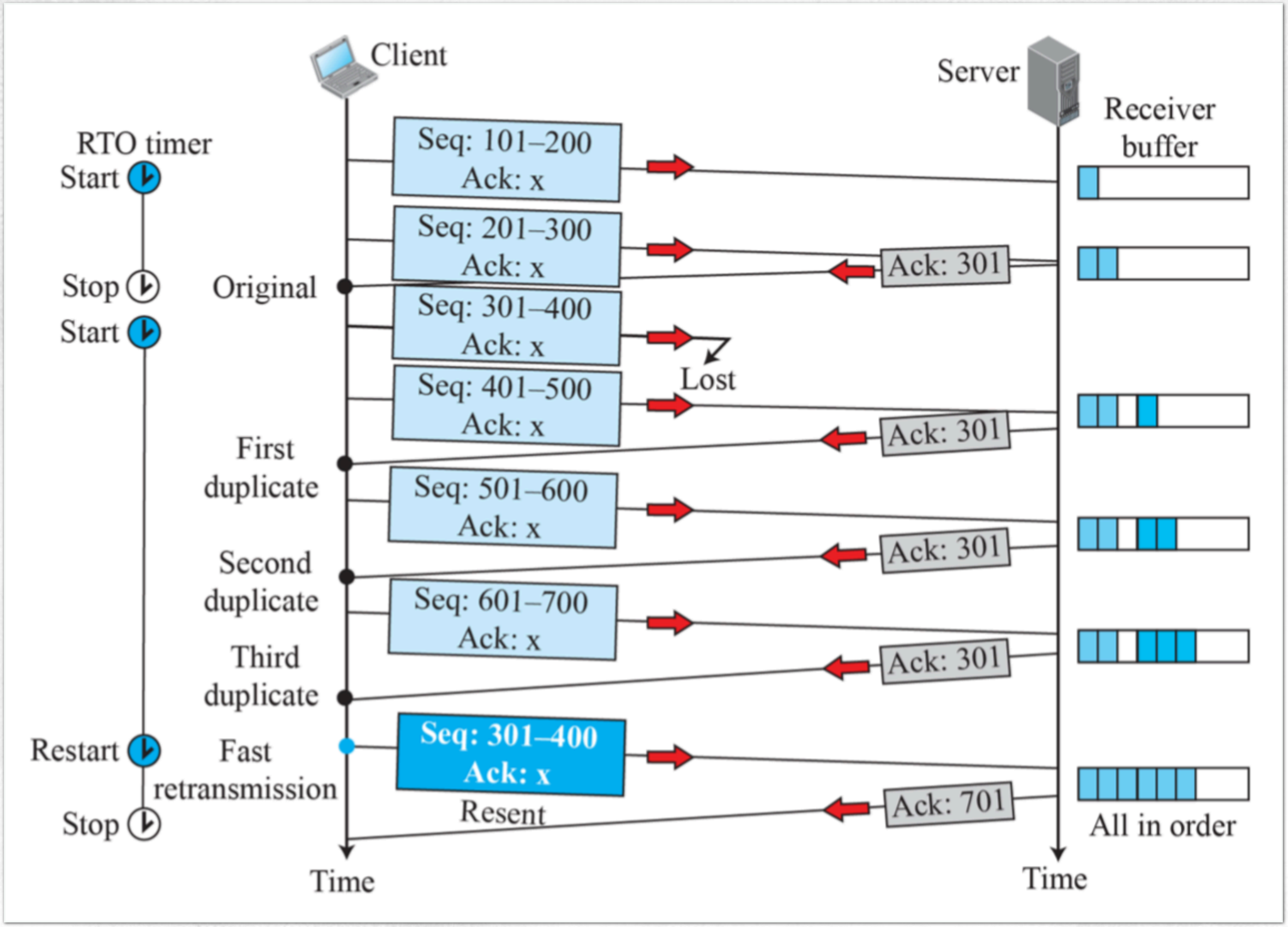
Normal operation



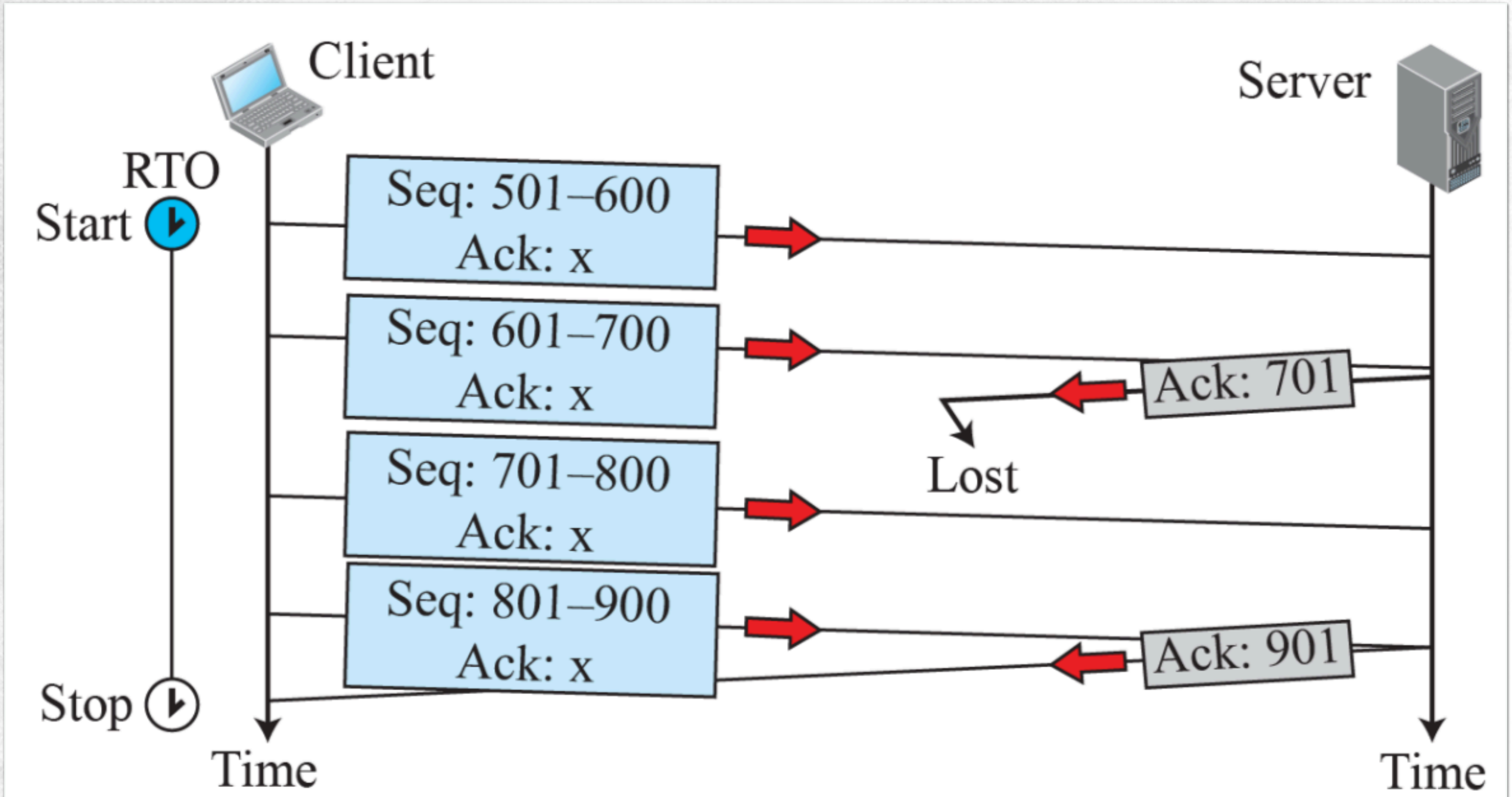
Lost segment



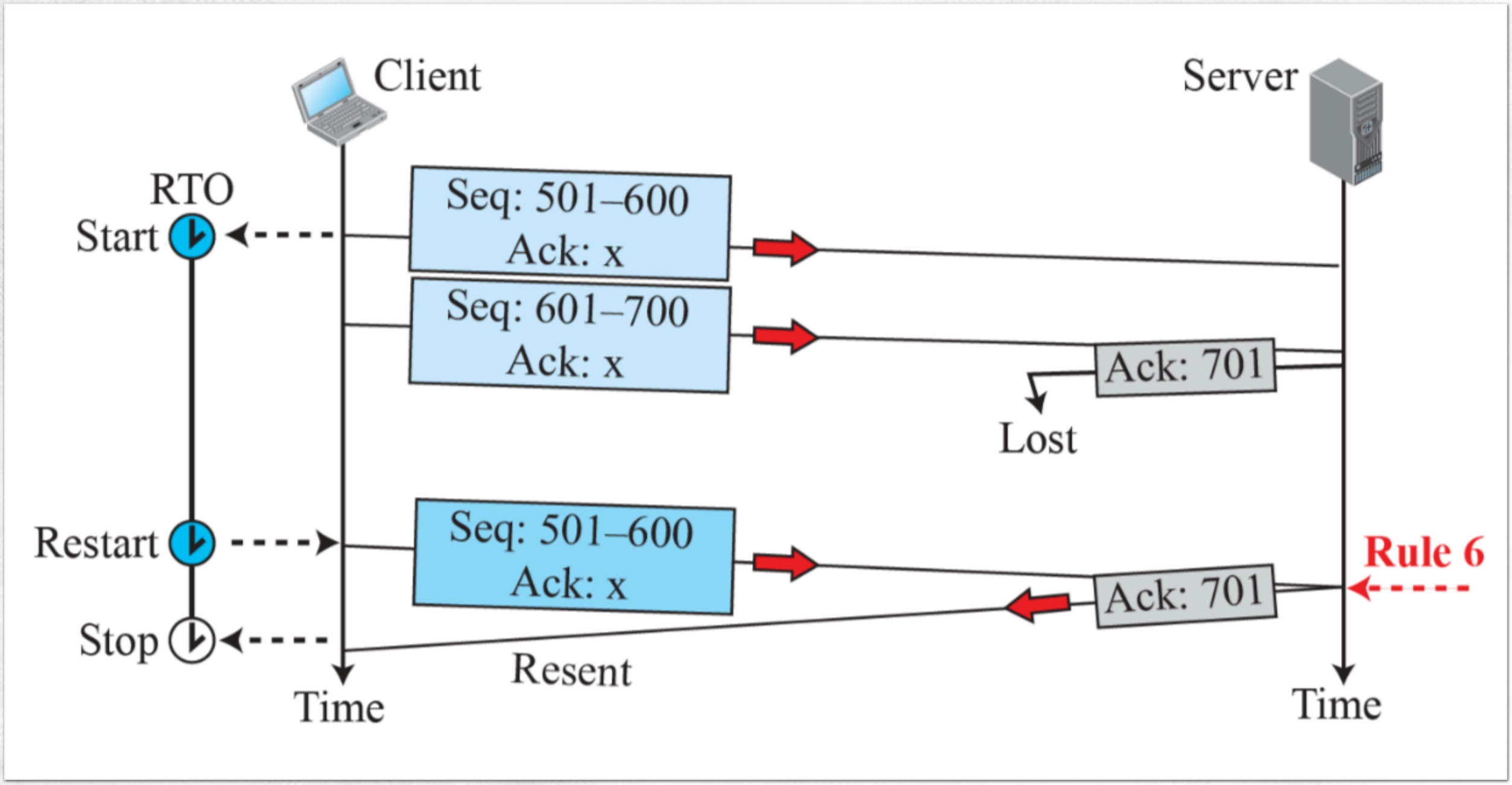
Fast retransmission



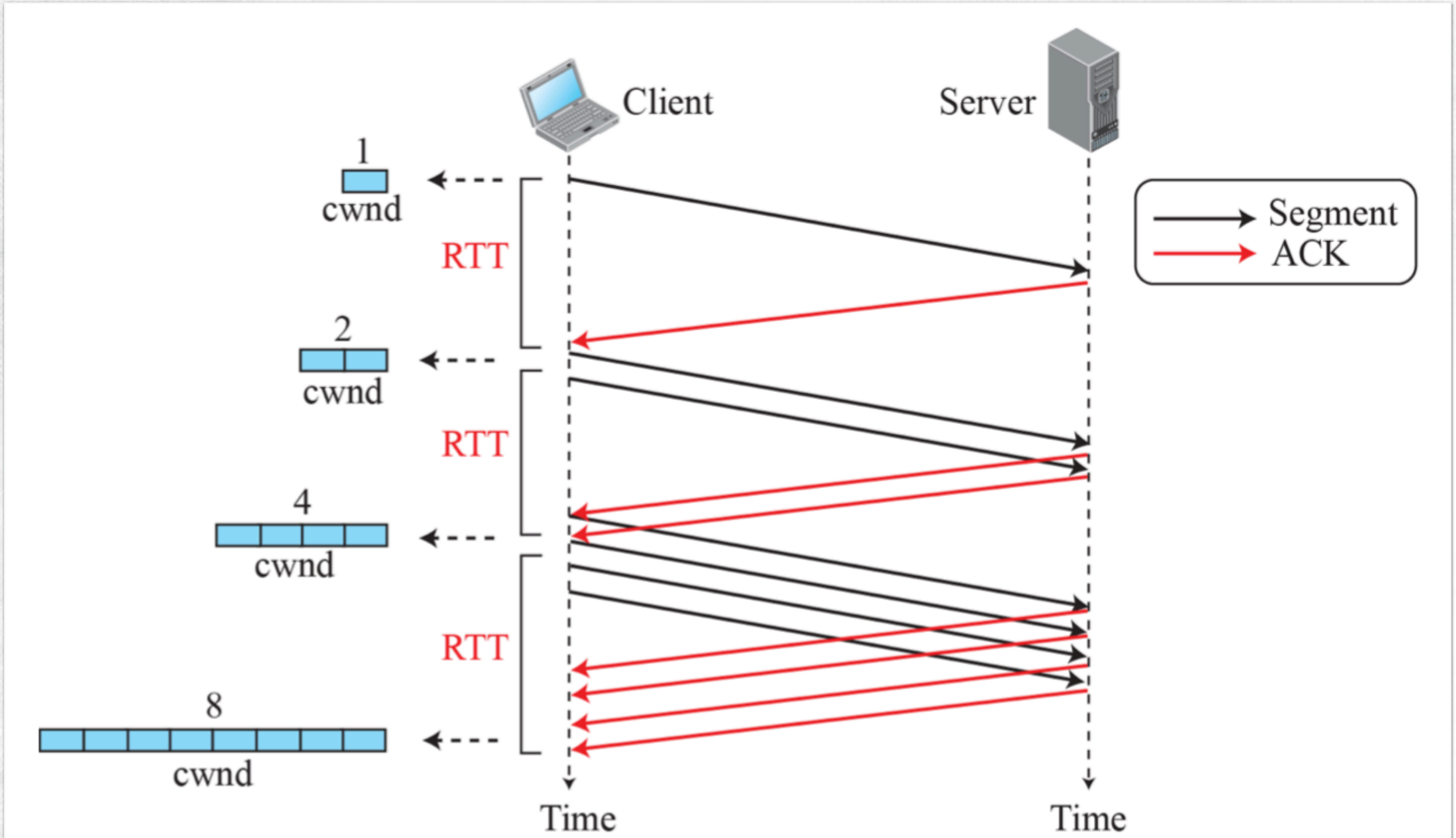
Lost acknowledgment



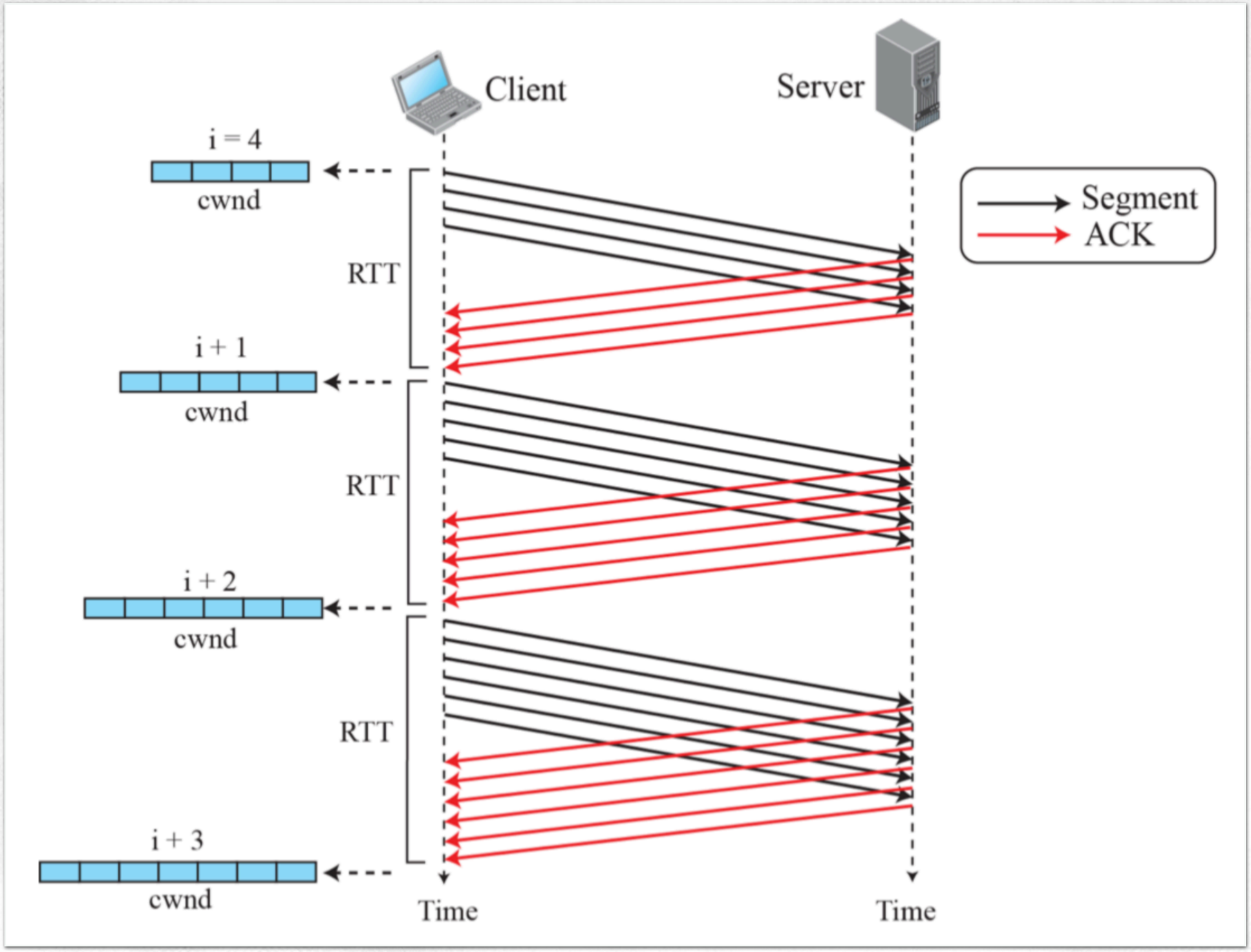
Lost acknowledgment corrected by resending a segment



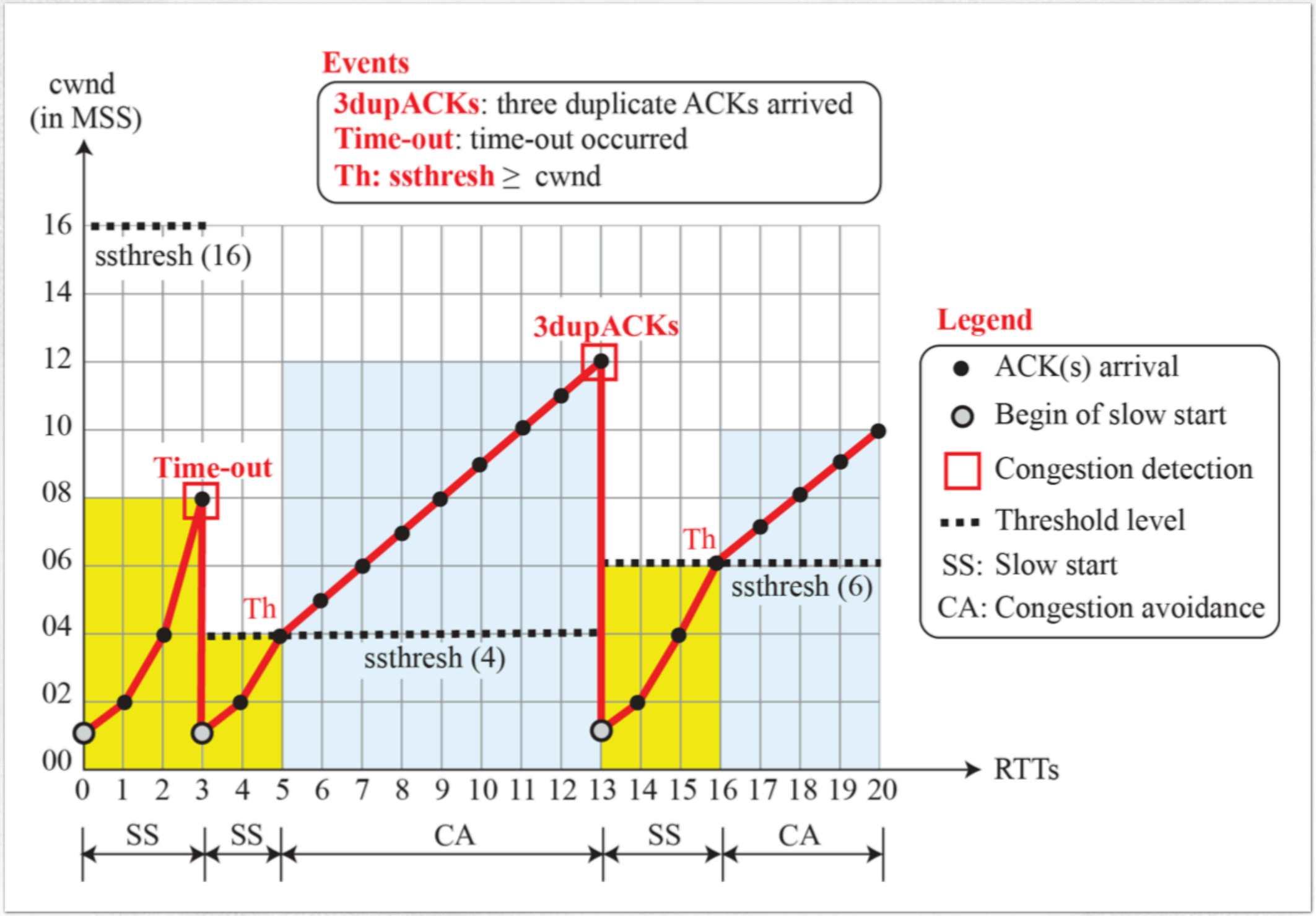
Slow start, exponential increase



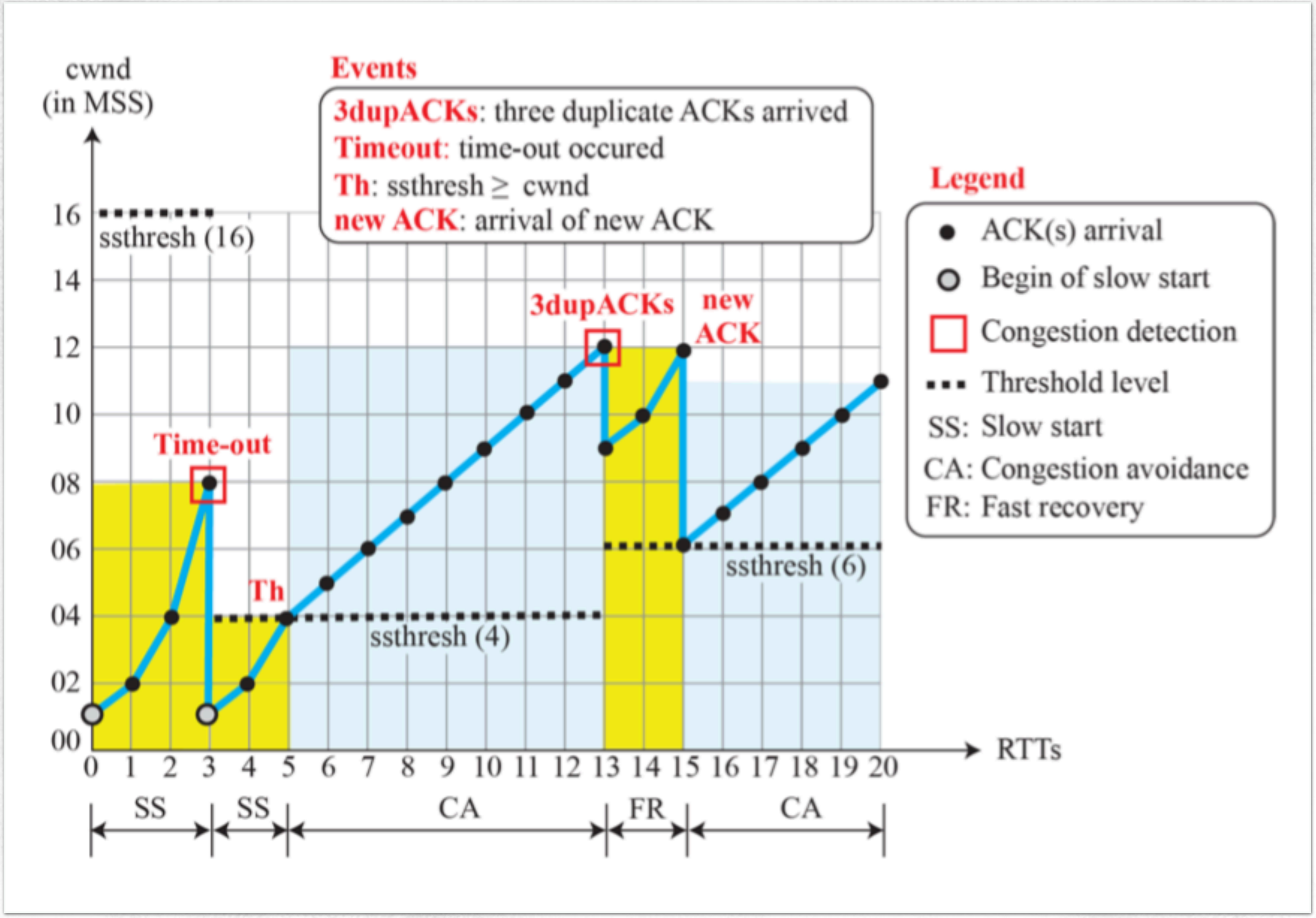
Congestion avoidance, additive increase



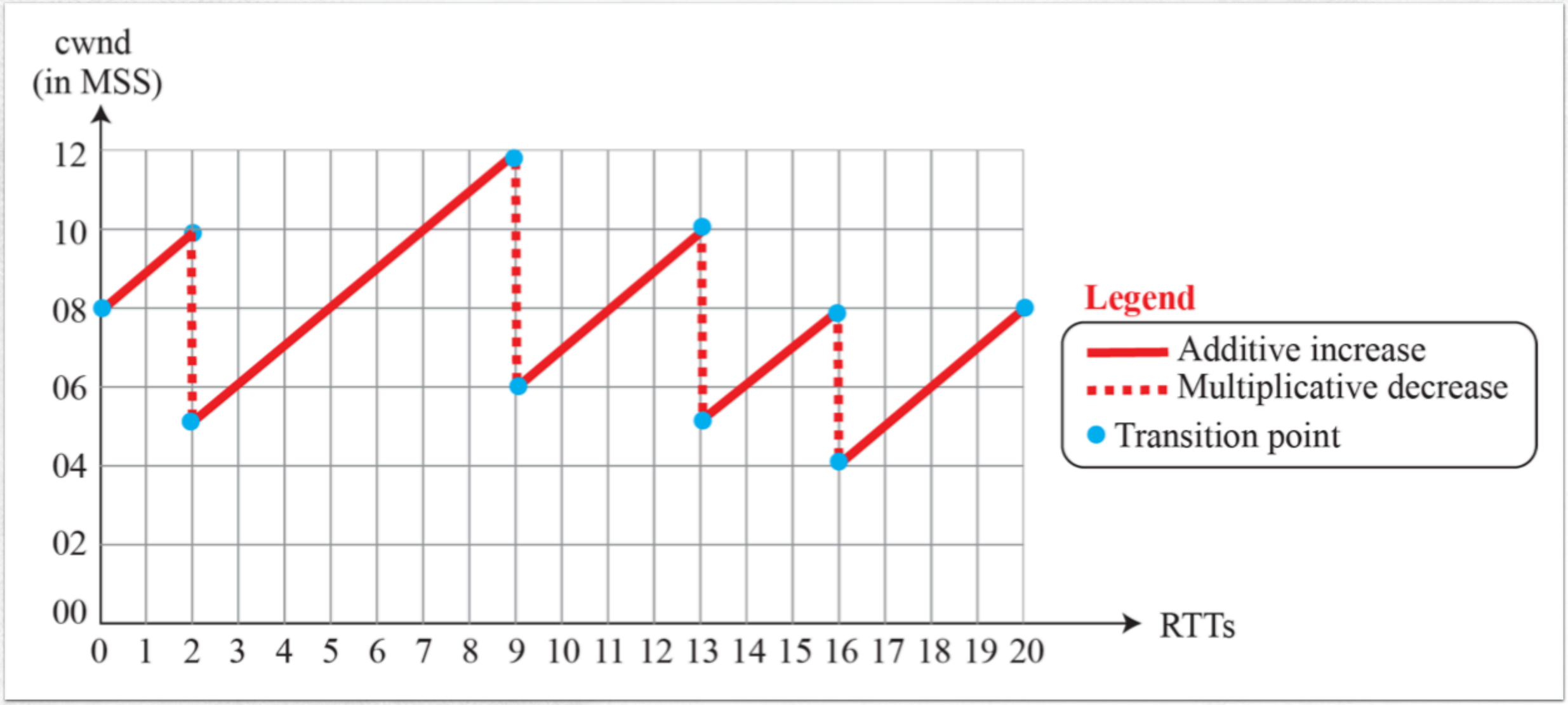
Example of Taho TCP



Example of Reno TCP



Additive increase, multiplicative decrease



Thank you

경희대학교
KYUNG HEE UNIVERSITY