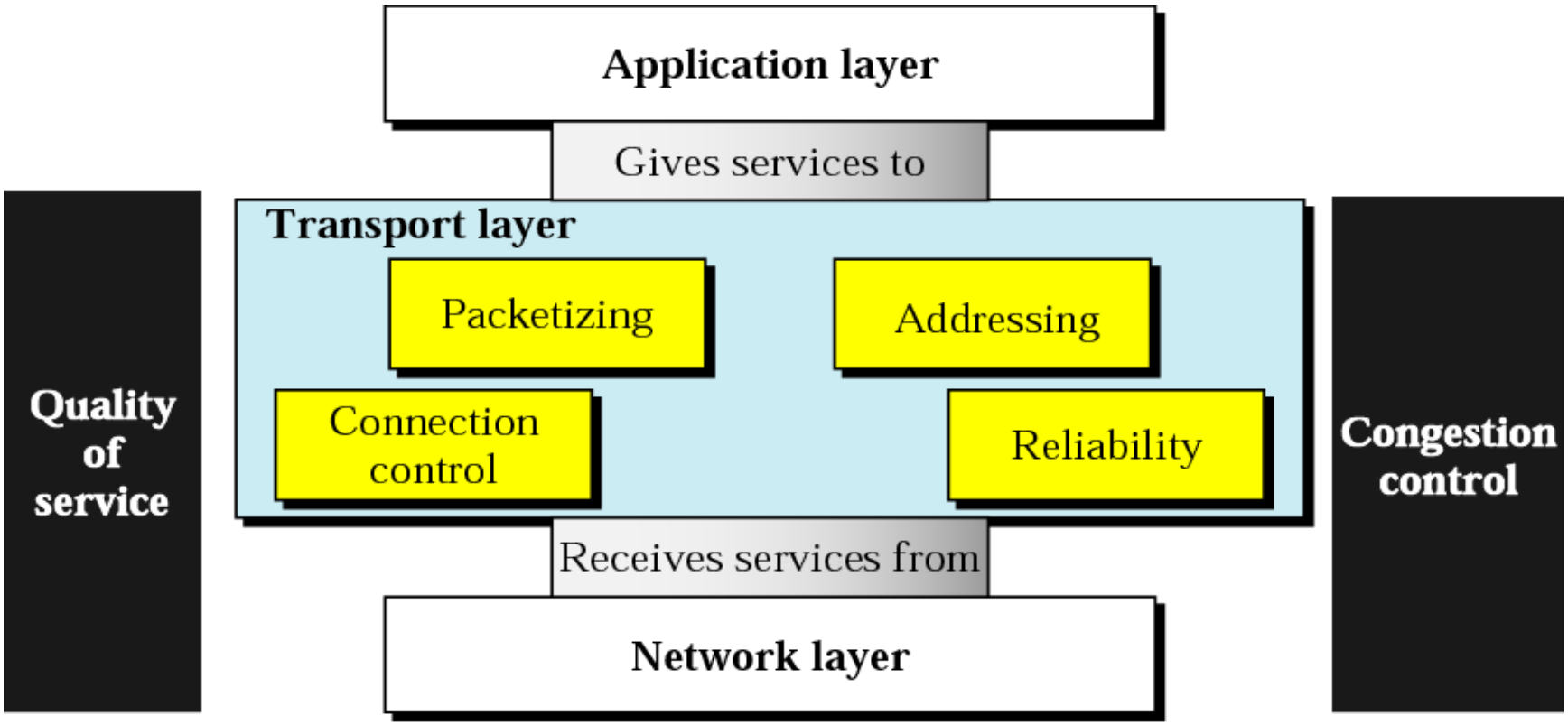
Transport layer: TCP and UDP

Computer Networks

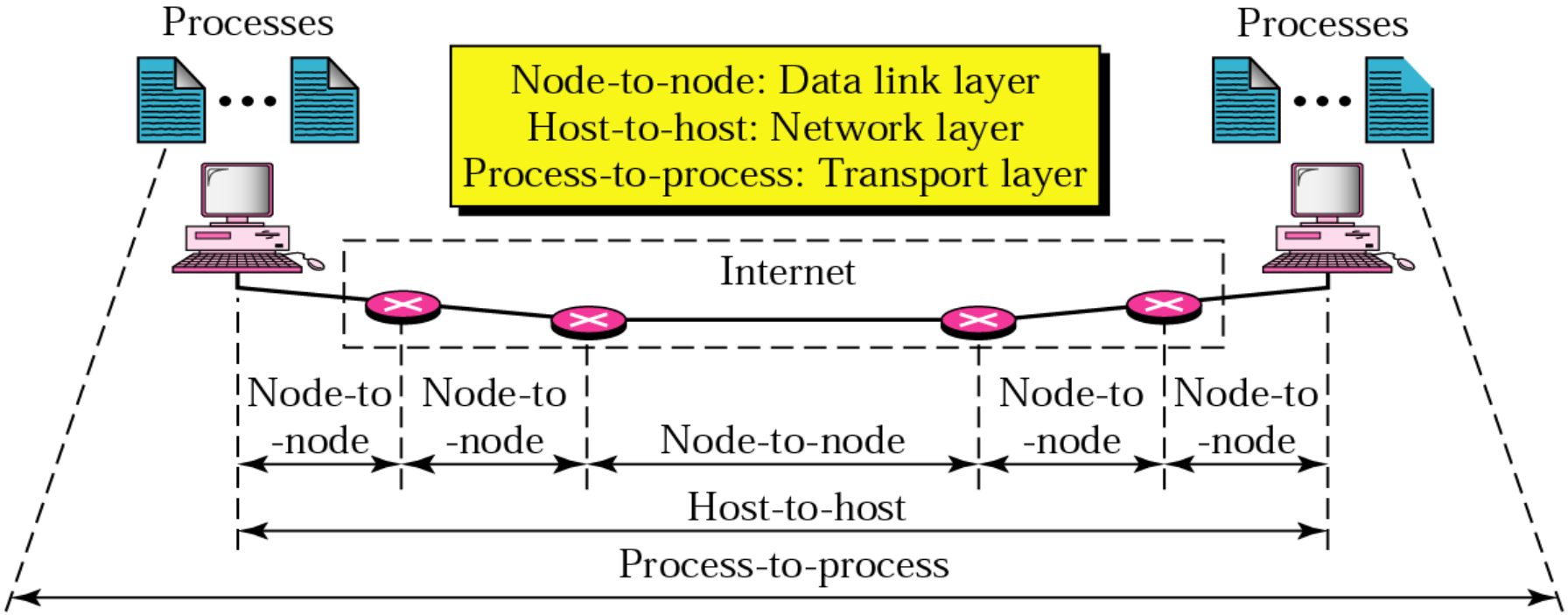
Faculty of Information Technology
Hanoi University

Position of Transport Layer

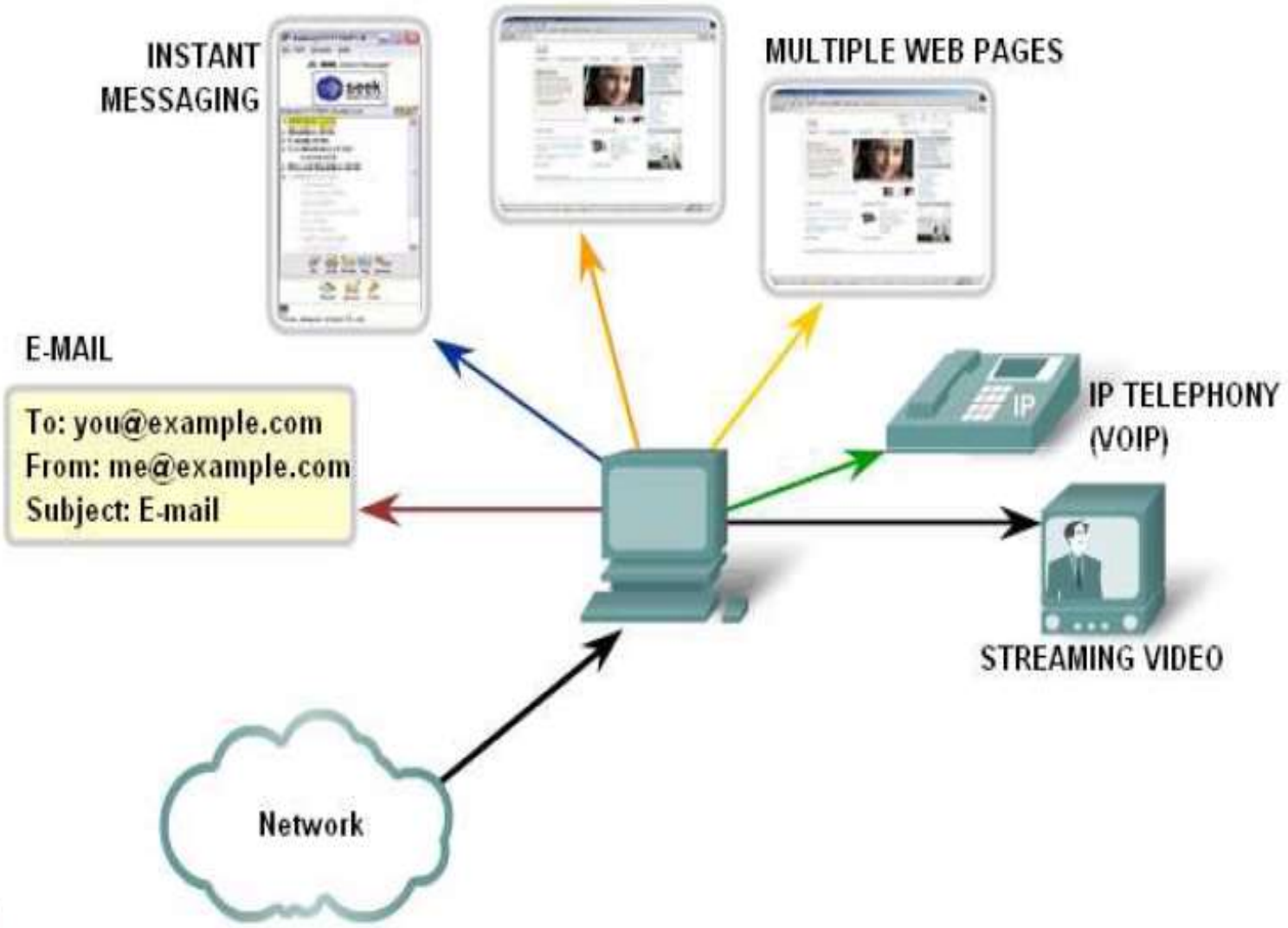


Process-Process Delivery

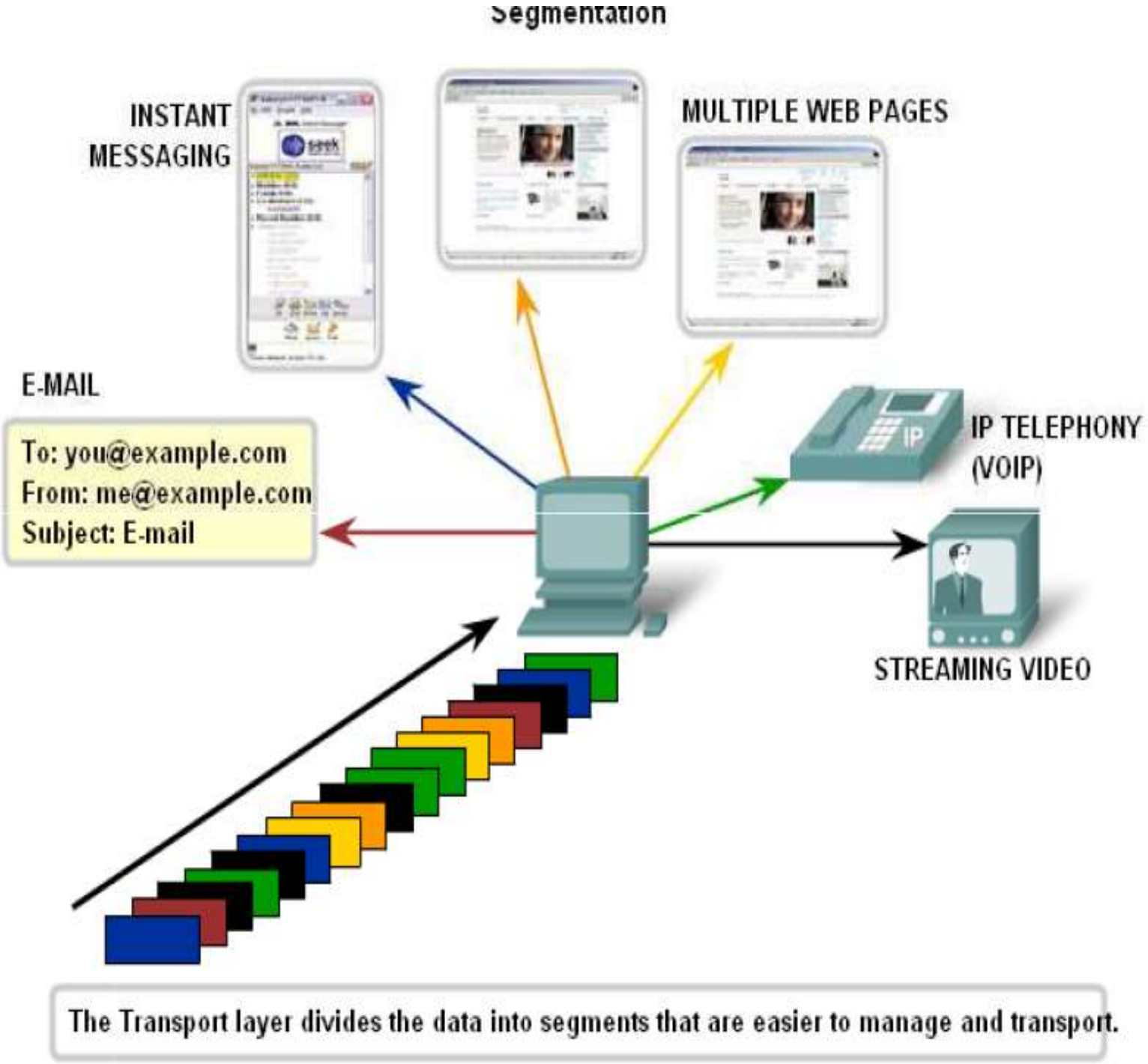
- The transport layer is responsible for process-to-process delivery



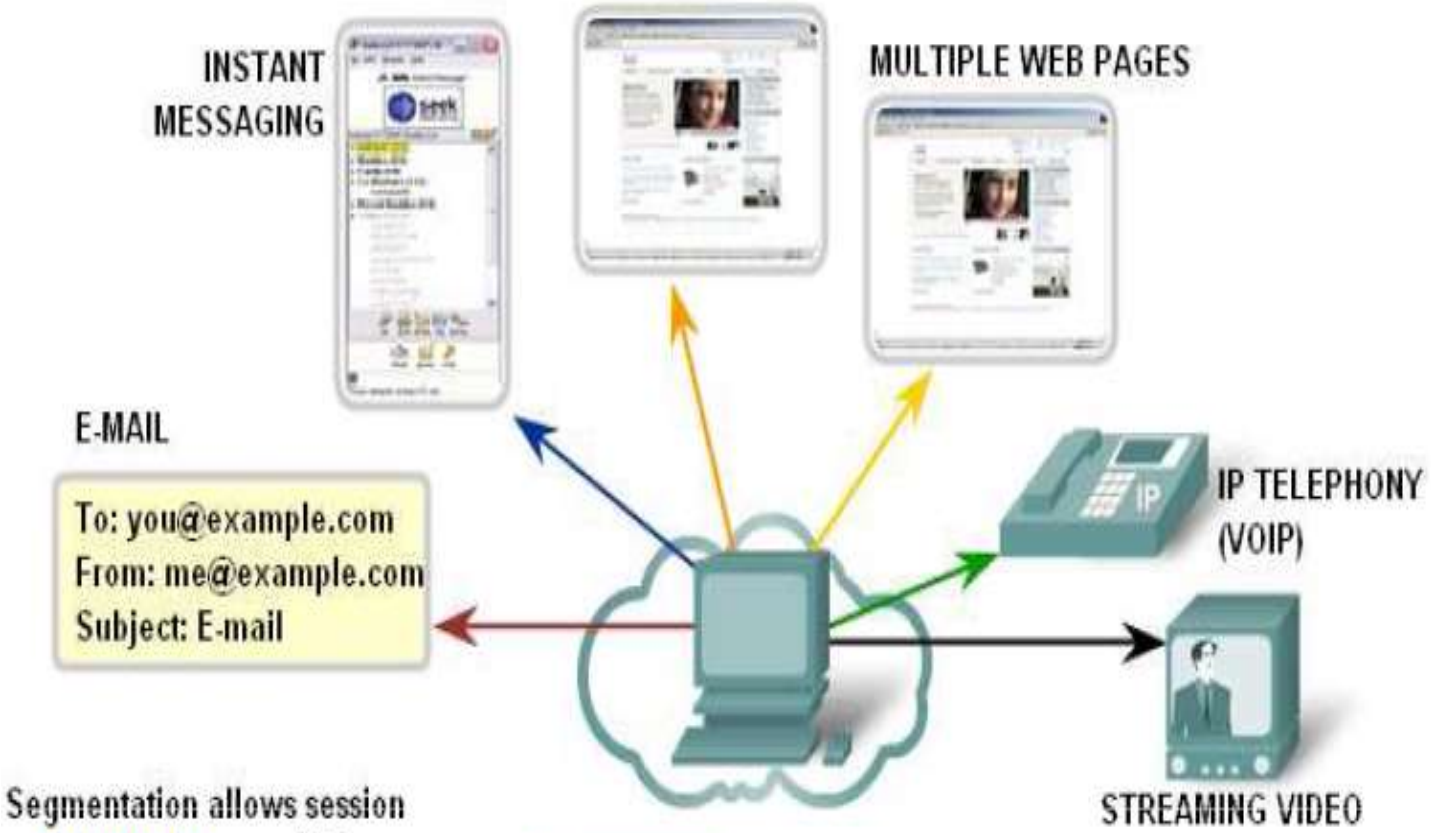
Tracking the Conversations



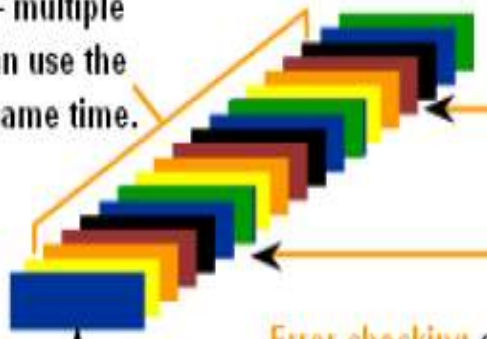
The Transport layer segments the data and manages the separation of data for different applications. Multiple applications running on a device receive the correct data.


Segmentation
INSTANT MESSAGING
MULTIPLE WEB PAGES
E-MAIL
To: you@example.com
From: me@example.com
Subject: E-mail
IP TELEPHONY (VOIP)
STREAMING VIDEO
The Transport layer divides the data into segments that are easier to manage and transport.

Transport Layer Services



Segmentation allows session multiplexing — multiple applications can use the network at the same time.



Data segmentation facilitates data carriage by the lower network layers.

Error checking can be performed on the data in the segment to check if the segment was changed during transmission.

Services Provided by Transport Layer

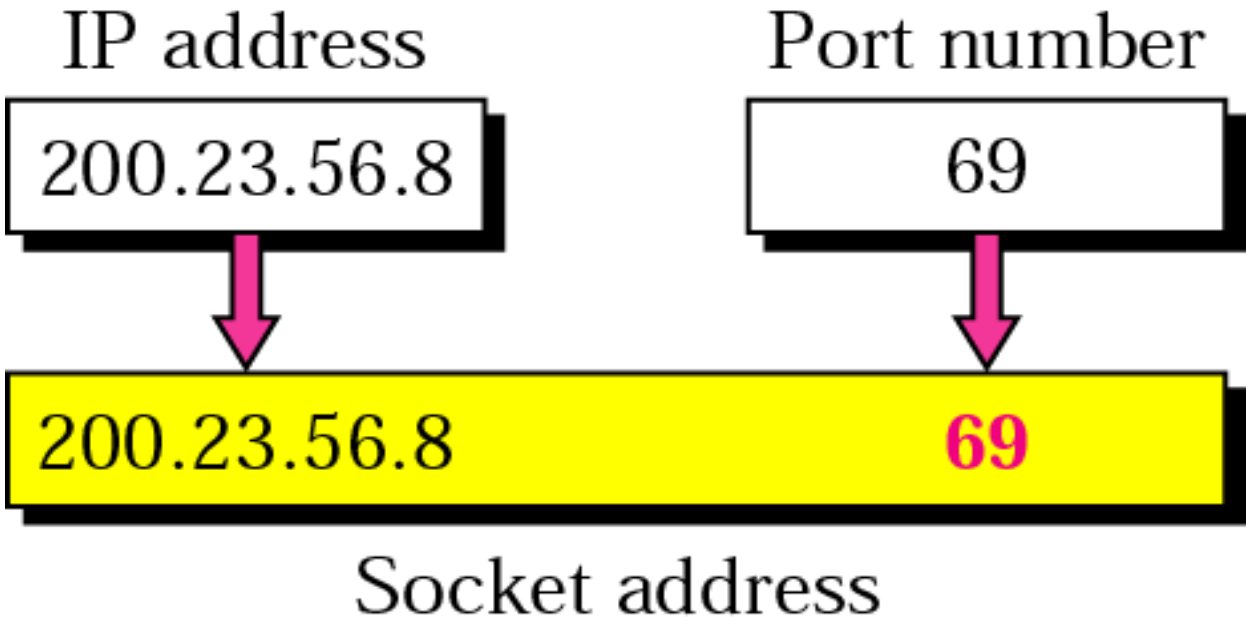
- The transport layer is responsible for completing the services of the underlying network to the extent that application development can take place
  - Unreliable connectionless service: UDP
  - Reliable connection-oriented service : TCP

Addressing

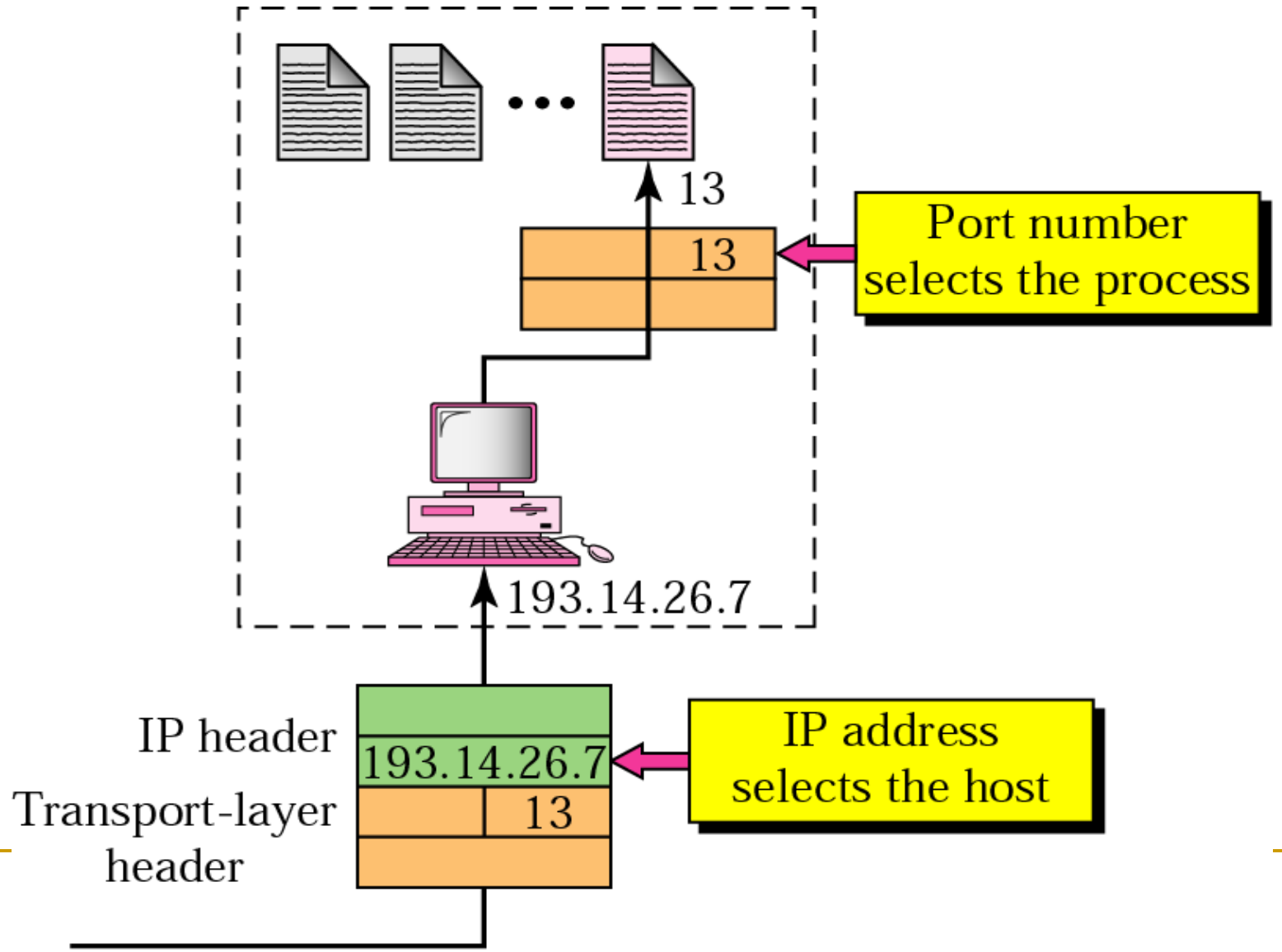
- For addressing, we use “transport addresses” on which processes can listen for connection requests : it is socket addresses
- A **socket address** is the combination of an IP address (the location of the computer) and a port (which is mapped to the application program process) into a single identity.

Sockets

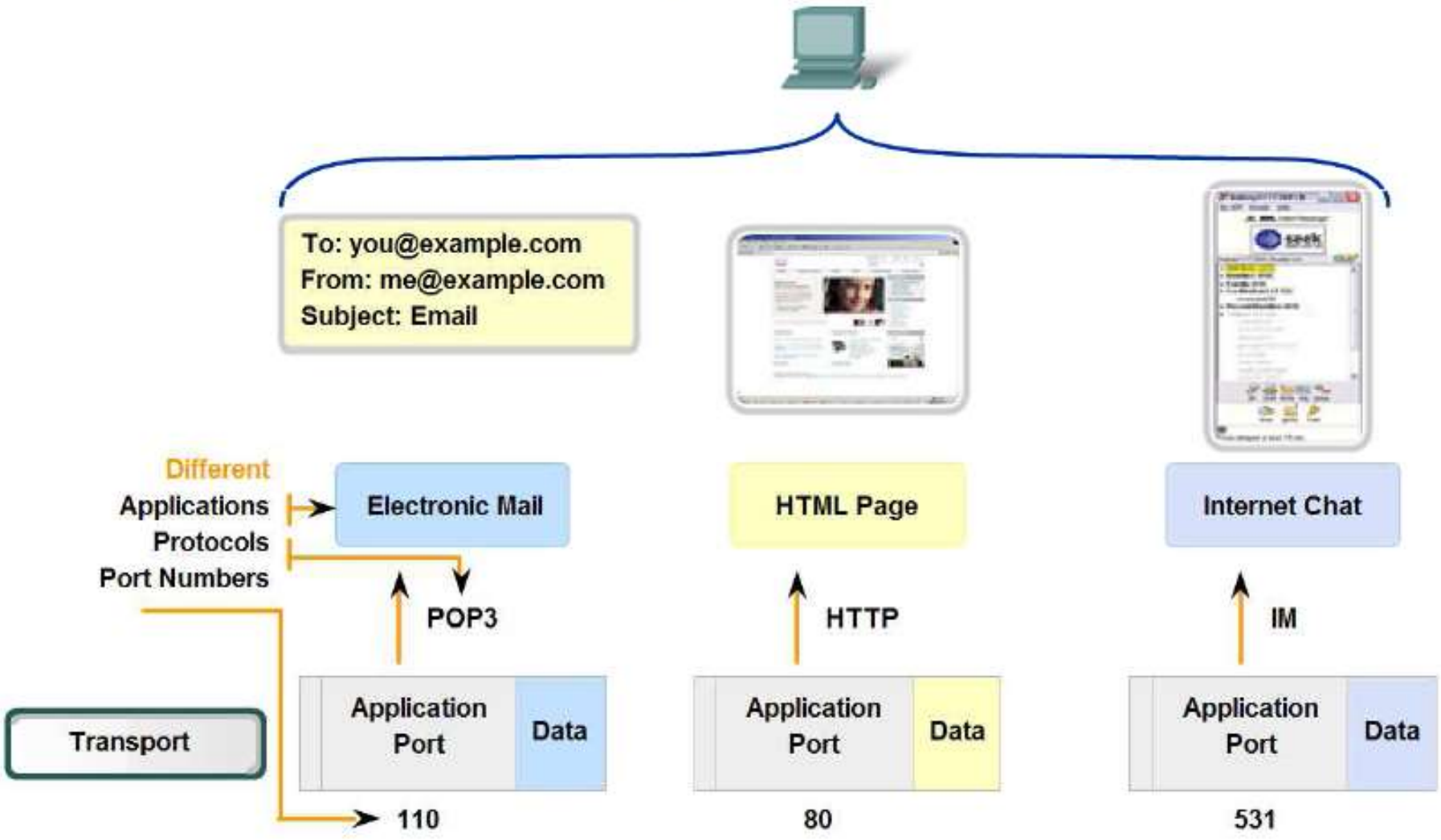
socket address = IP address + a port.



Sockets


13
13
Port number
selects the process
193.14.26.7
IP header
193.14.26.7
Transport-layer
header
13
IP address
selects the host

Port Addressing

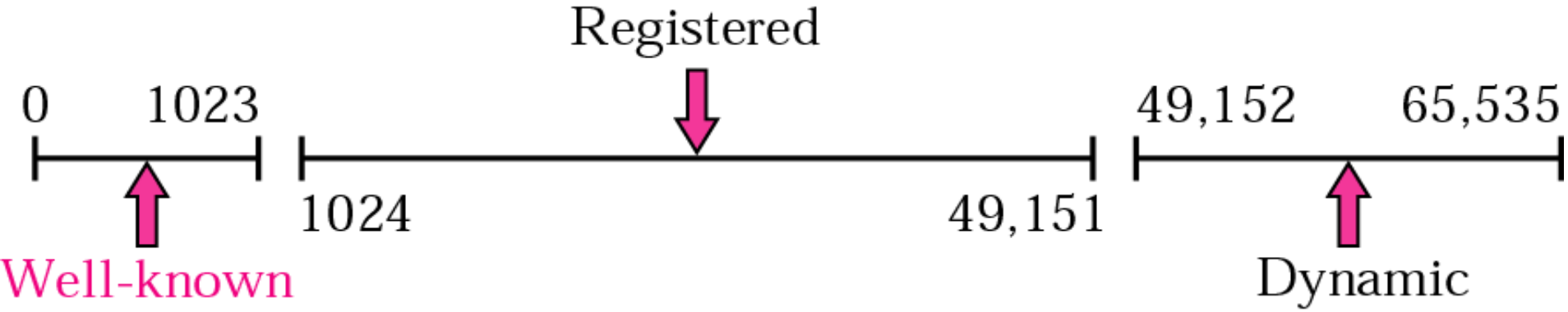


Data for different applications is directed to the correct application because each application has a unique port number.

Port Assignment

- How does a user process on a host knows on which destination port a server is running?
- Port is a 16-bit logical address (65536 ports)
- There are 3 port groups:

Port Assignment



http://www.ports-services.com/

Port Assignment

Port Number Range	Port Group
0 to 1023	Well Known (Contact) Ports
1024 to 49151	Registered Ports
49152 to 65535	Private and/or Dynamic Ports

Registered Ports:
1863 MSN Messenger
8008 Alternate HTTP
8080 Alternate HTTP

Well Known Ports

21	FTP
23	Telnet
25	SMTP
80	HTTP
110	POP3
194	Internet Relay Chat (IRC)
443	Secure HTTP (HTTPS)

3 port groups

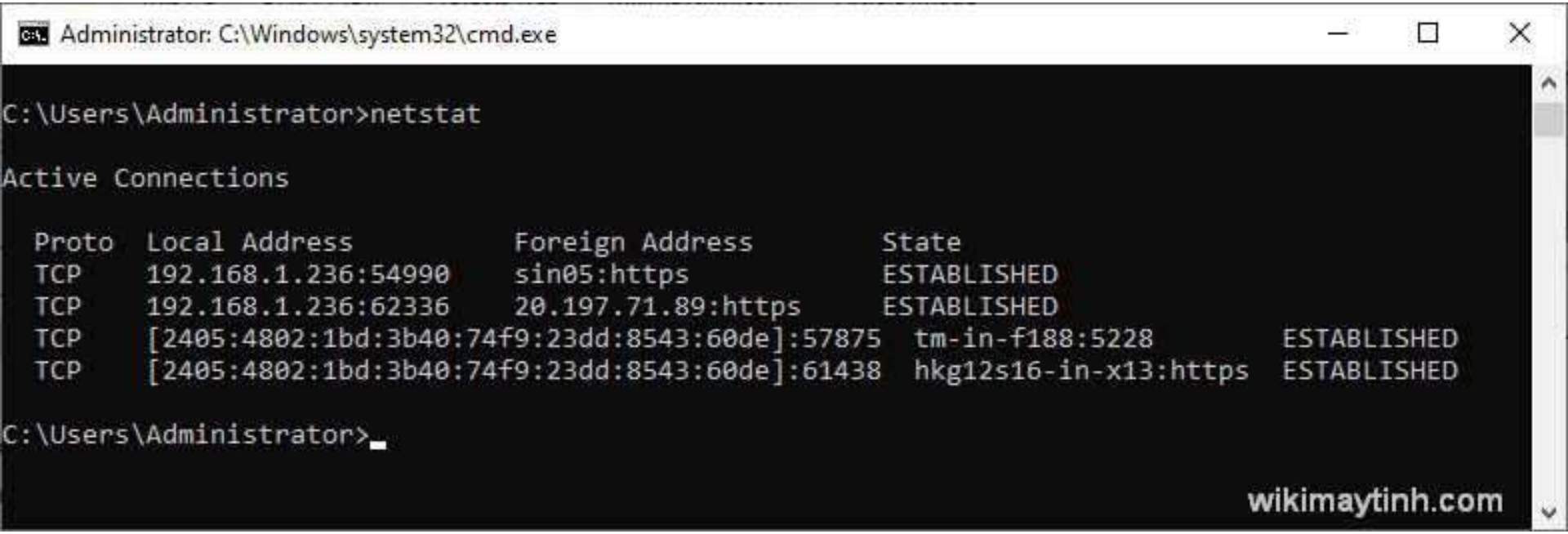
- **Well-known ports**: $0 - 1023$, they are assigned by IANA and used by system processes that provide widely-used types of network services.
- **Registered ports**: $1024 - 49151$, they are assigned by IANA for specific service upon application by a requesting entity.
- **Dynamic ports**: $49152 - 65535$, contains dynamic or private ports that cannot be registered with IANA. This range is used for custom or temporary purposes and for automatic allocation.

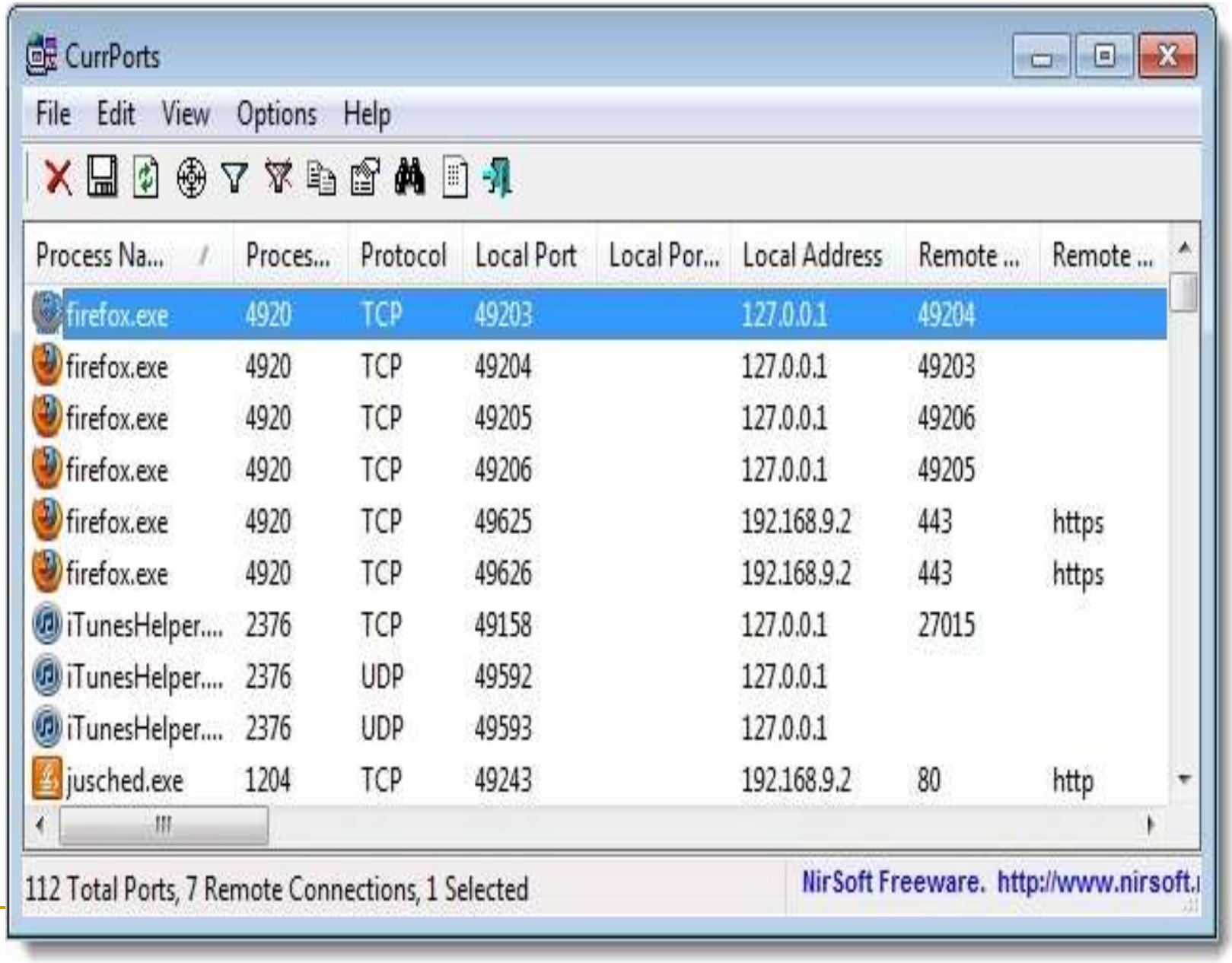
Well-known Ports

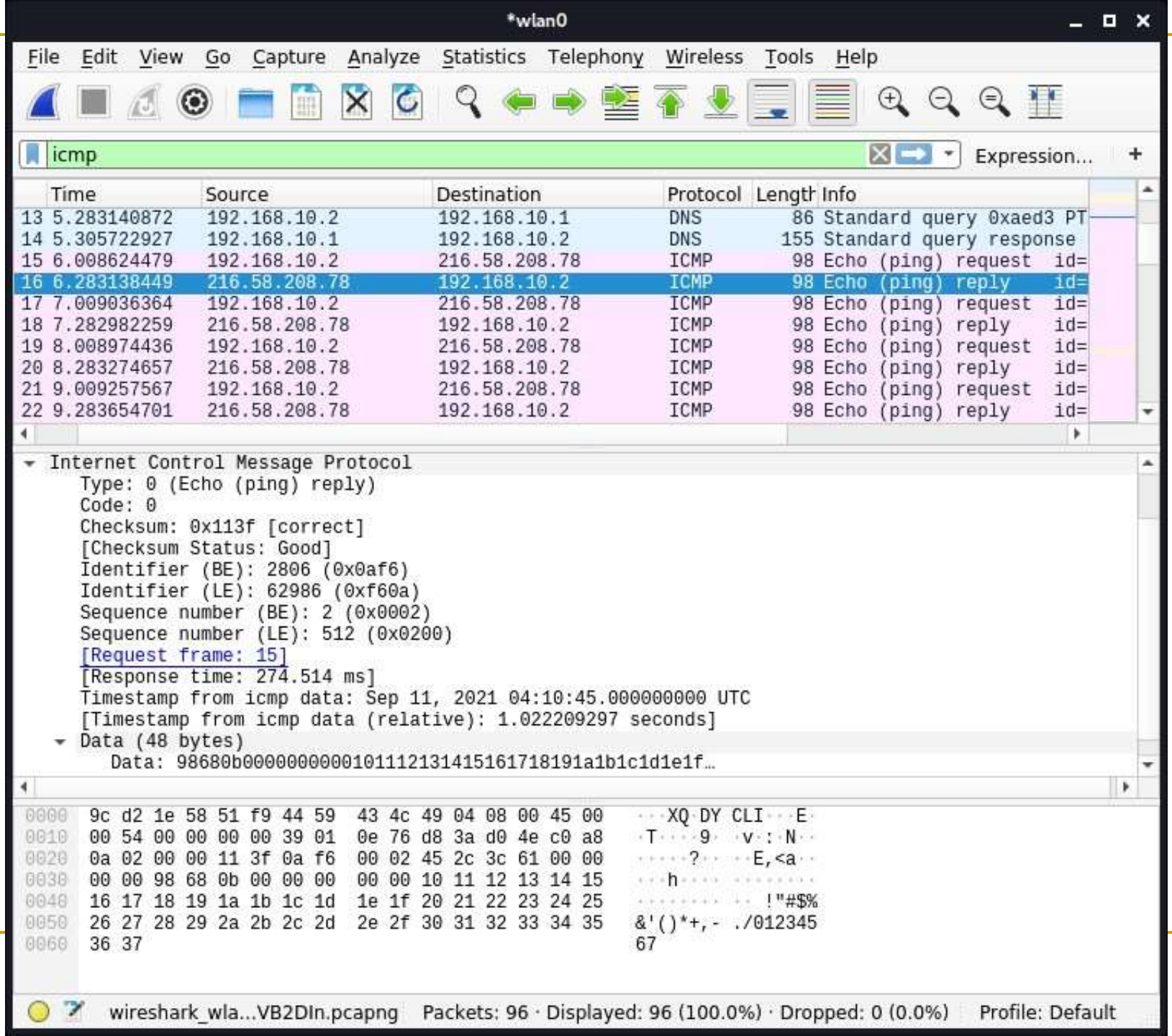
- assigned by IANA (Internet Assigned Numbers Authority)
- Some well-known ports:
  - HTTP 80
  - HTTPS 443
  - SMTP 25
  - DNS 53
  - POP3 110
  - POP3S 995
  - SSH 22
  - TELNET 23
  - FTP 21

Port Number	Transport Protocol	Service Name	RFC
20, 21	TCP	File Transfer Protocol (FTP)	RFC 959
22	TCP and UDP	Secure Shell (SSH)	RFC 4250-4256
23	TCP	Telnet	RFC 854
25	TCP	Simple Mail Transfer Protocol (SMTP)	RFC 5321
53	TCP and UDP	Domain Name Server (DNS)	RFC 1034-1035
67, 68	UDP	Dynamic Host Configuration Protocol (DHCP)	RFC 2131
69	UDP	Trivial File Transfer Protocol (TFTP)	RFC 1350
80	TCP	HyperText Transfer Protocol (HTTP)	RFC 2616
110	TCP	Post Office Protocol (POP3)	RFC 1939
119	TCP	Network News Transport Protocol (NNTP)	RFC 8977
123	UDP	Network Time Protocol (NTP)	RFC 5905
135-139	TCP and UDP	NetBIOS	RFC 1001-1002
143	TCP and UDP	Internet Message Access Protocol (IMAP4)	RFC 3501
161, 162	TCP and UDP	Simple Network Management Protocol (SNMP)	RFC 1901-1908, 3411-3418
179	TCP	Border Gateway Protocol (BGP)	RFC 4271
389	TCP and UDP	Lightweight Directory Access Protocol	RFC 4510
443	TCP and UDP	HTTP with Secure Sockets Layer (SSL)	RFC 2818
500	UDP	Internet Security Association and Key Management Protocol (ISAKMP) / Internet Key Exchange (IKE)	RFC 2408 - 2409
636	TCP and UDP	Lightweight Directory Access Protocol over TLS/SSL (LDAPS	RFC 4513
989/990	TCP	FTP over TLS/SSL	RFC 4217

How to see port used in a computer?



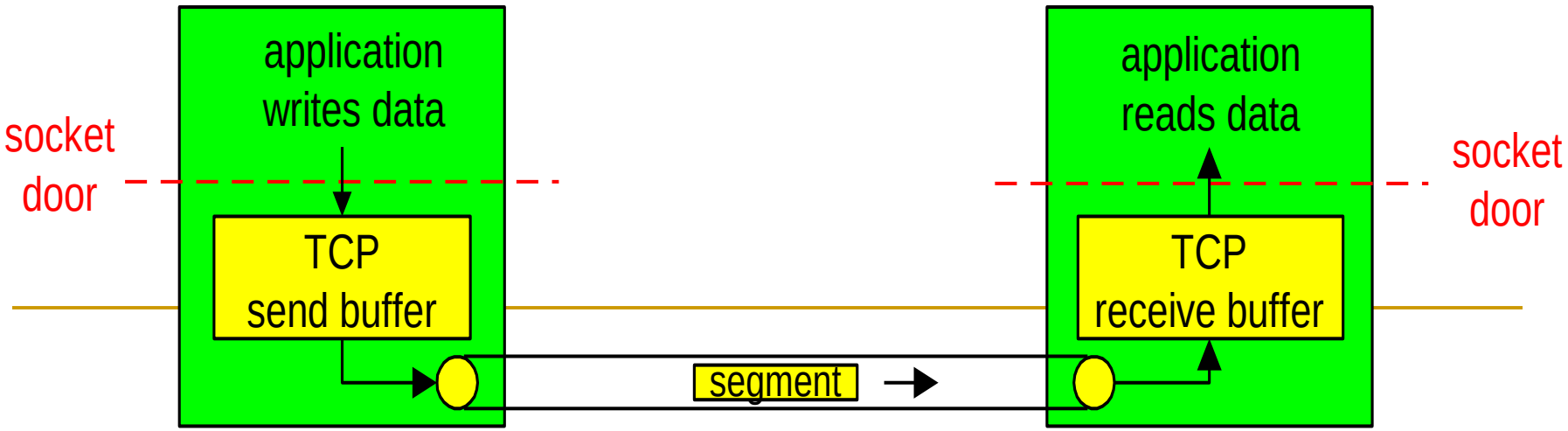

CurrPorts
File Edit View Options Help
Process Na... / Proces... Protocol Local Port Local Por... Local Address Remote ... Remote ...
firefox.exe 4920 TCP 49203 127.0.0.1 49204
firefox.exe 4920 TCP 49204 127.0.0.1 49203
firefox.exe 4920 TCP 49205 127.0.0.1 49206
firefox.exe 4920 TCP 49206 127.0.0.1 49205
firefox.exe 4920 TCP 49625 192.168.9.2 443 https
firefox.exe 4920 TCP 49626 192.168.9.2 443 https
iTunesHelper.... 2376 TCP 49158 127.0.0.1 27015
iTunesHelper.... 2376 UDP 49592 127.0.0.1
iTunesHelper.... 2376 UDP 49593 127.0.0.1
jusched.exe 1204 TCP 49243 192.168.9.2 80 http
112 Total Ports, 7 Remote Connections, 1 Selected
NirSoft Freeware. http://www.nirsoft.


*wlan0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
icmp
Expression...
Time Source Destination Protocol Length Info
13 5.283140872 192.168.10.2 192.168.10.1 DNS 86 Standard query 0xaed3 PT
14 5.305722927 192.168.10.1 192.168.10.2 DNS 155 Standard query response
15 6.008624479 192.168.10.2 216.58.208.78 ICMP 98 Echo (ping) request id=
16 6.283138449 216.58.208.78 192.168.10.2 ICMP 98 Echo (ping) reply id=
17 7.009036364 192.168.10.2 216.58.208.78 ICMP 98 Echo (ping) request id=
18 7.282982259 216.58.208.78 192.168.10.2 ICMP 98 Echo (ping) reply id=
19 8.008974436 192.168.10.2 216.58.208.78 ICMP 98 Echo (ping) request id=
20 8.283274657 216.58.208.78 192.168.10.2 ICMP 98 Echo (ping) reply id=
21 9.009257567 192.168.10.2 216.58.208.78 ICMP 98 Echo (ping) request id=
22 9.283654701 216.58.208.78 192.168.10.2 ICMP 98 Echo (ping) reply id=
Internet Control Message Protocol
Type: 0 (Echo (ping) reply)
Code: 0
Checksum: 0x113f [correct]
[Checksum Status: Good]
Identifier (BE): 2806 (0x0af6)
Identifier (LE): 62986 (0xf60a)
Sequence number (BE): 2 (0x0002)
Sequence number (LE): 512 (0x0200)
[Request frame: 15]
[Response time: 274.514 ms]
Timestamp from icmp data: Sep 11, 2021 04:10:45.000000000 UTC
[Timestamp from icmp data (relative): 1.022209297 seconds]
Data (48 bytes)
Data: 98680b000000000010111213141516171819a1b1c1d1e1f…
0000 9c d2 1e 58 51 f9 44 59 43 4c 49 04 08 00 45 00
0010 00 54 00 00 00 00 39 01 0e 76 d8 3a d0 4e c0 a8
0020 0a 02 00 00 11 3f 0a f6 00 02 45 2c 3c 61 00 00
0030 00 00 98 68 0b 00 00 00 00 00 10 11 12 13 14 15
0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25
0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35
0060 36 37
wireshark_wla...VB2DIn.pcapng Packets: 96 · Displayed: 96 (100.0%) · Dropped: 0 (0.0%) Profile: Default

TCP
(Tranmission control protocol)

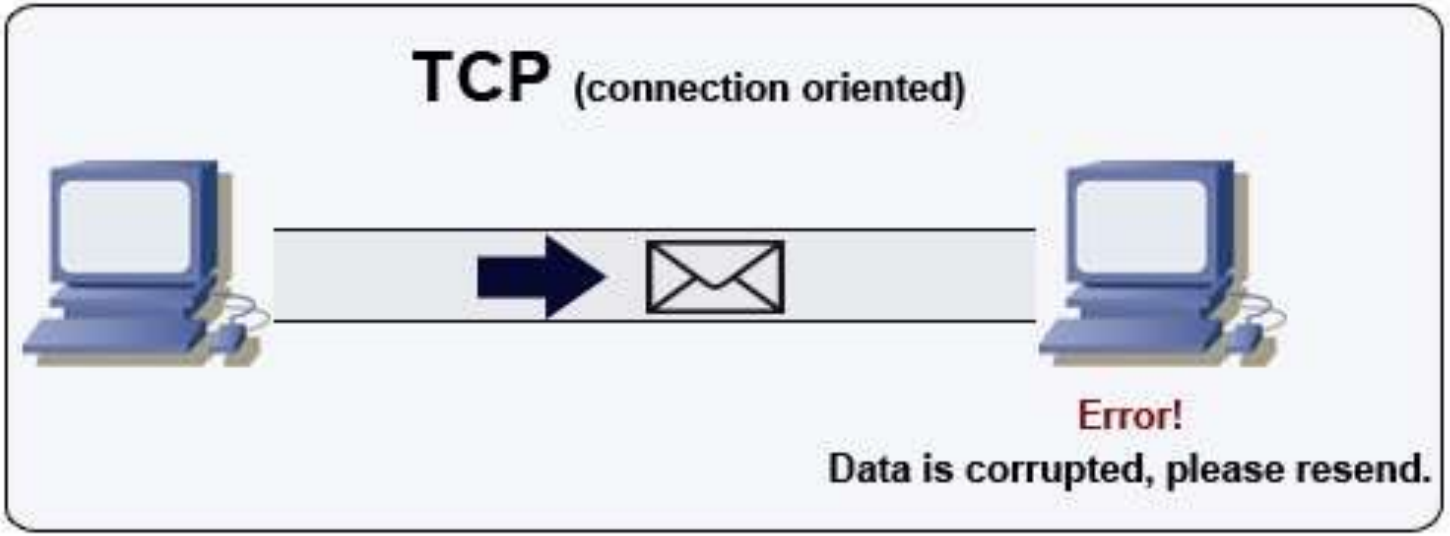
TCP

- RFCs 793, 1122, 1323, 2018, 2581
- Point-to-Point
  - one sender, one receiver
- Reliable, in-order byte stream
  - no “message boundaries”
- Pipelined
  - Send & Receive buffers/windows



TCP – Transmission Control Protocol

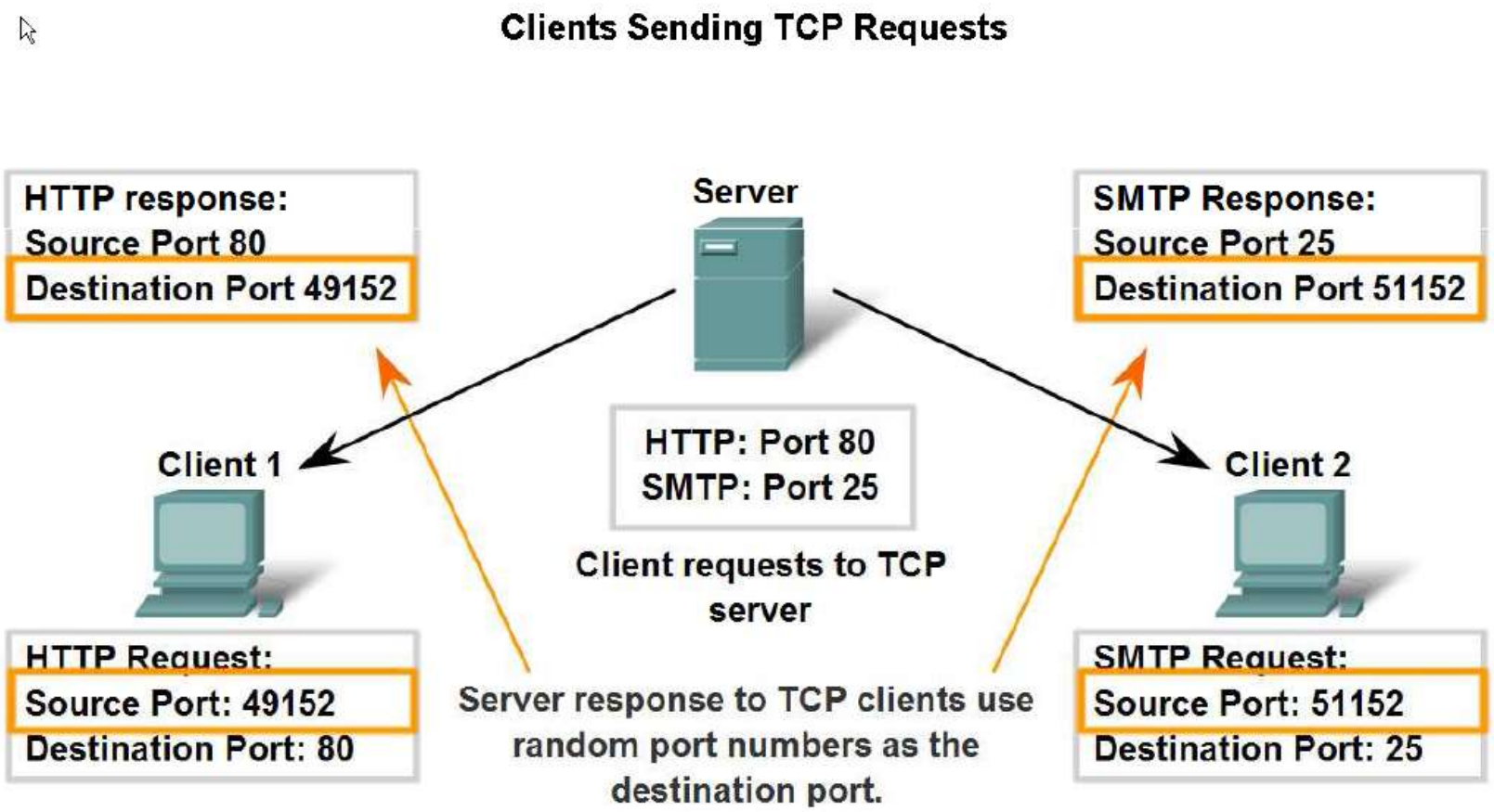
- Full duplex
  - bi-directional data flow in same connection
- Connection-oriented
  - Exchange of control messages initiates sender,



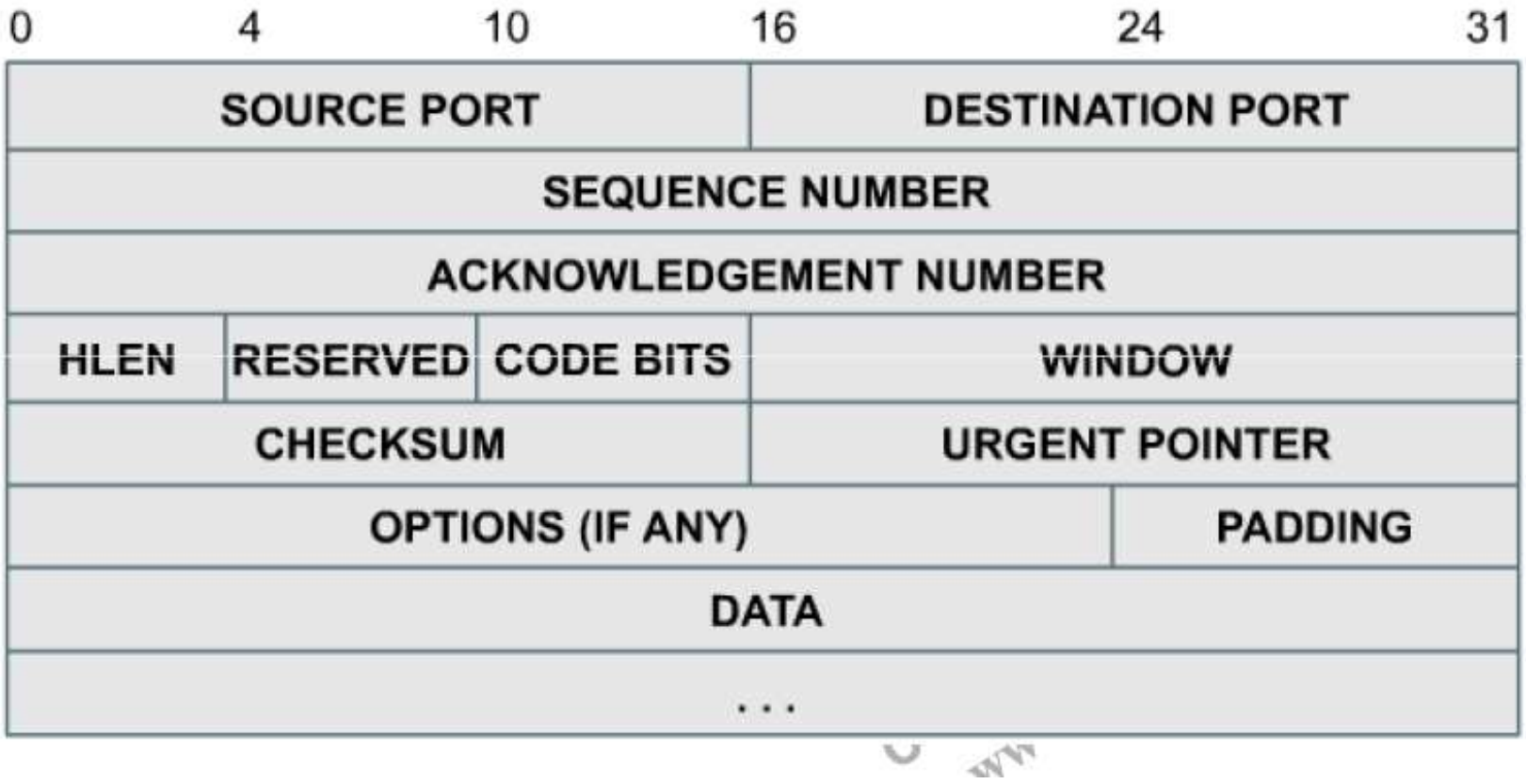
TCP – Transmission Control Protocol

- Flow control and Congestion Control
  - sender will not overwhelm receiver or the network
- End-to-end:
  - TCP does not support multicasting or broadcasting.

TCP Server


Clients Sending TCP Requests
HTTP response:
Source Port 80
Destination Port 49152
Server
SMTP Response:
Source Port 25
Destination Port 51152
HTTP: Port 80
SMTP: Port 25
Client 1
Client 2
Client requests to TCP server
HTTP Request:
Source Port: 49152
Destination Port: 80
Server response to TCP clients use random port numbers as the destination port.
SMTP Request:
Source Port: 51152
Destination Port: 25

TCP Header


0
4
10
16
24
31
SOURCE PORT
DESTINATION PORT
SEQUENCE NUMBER
ACKNOWLEDGEMENT NUMBER
HLEN
RESERVED
CODE BITS
WINDOW
CHECKSUM
URGENT POINTER
OPTIONS (IF ANY)
PADDING
DATA
. . .

Code bits

- 8 Flags of control to manage TCP activities :
  - URG : Urgent.
  - ACK : Acknowledgement.
  - PSH : Requests Push.
  - RST : Reset connection.
  - SYN : Synchronize sequence numbers.
  - FIN : sender finished.

TCP Connections

- TCP connections have three phases
  1. Connection establishment
  2. Data transfer
  3. Connection termination

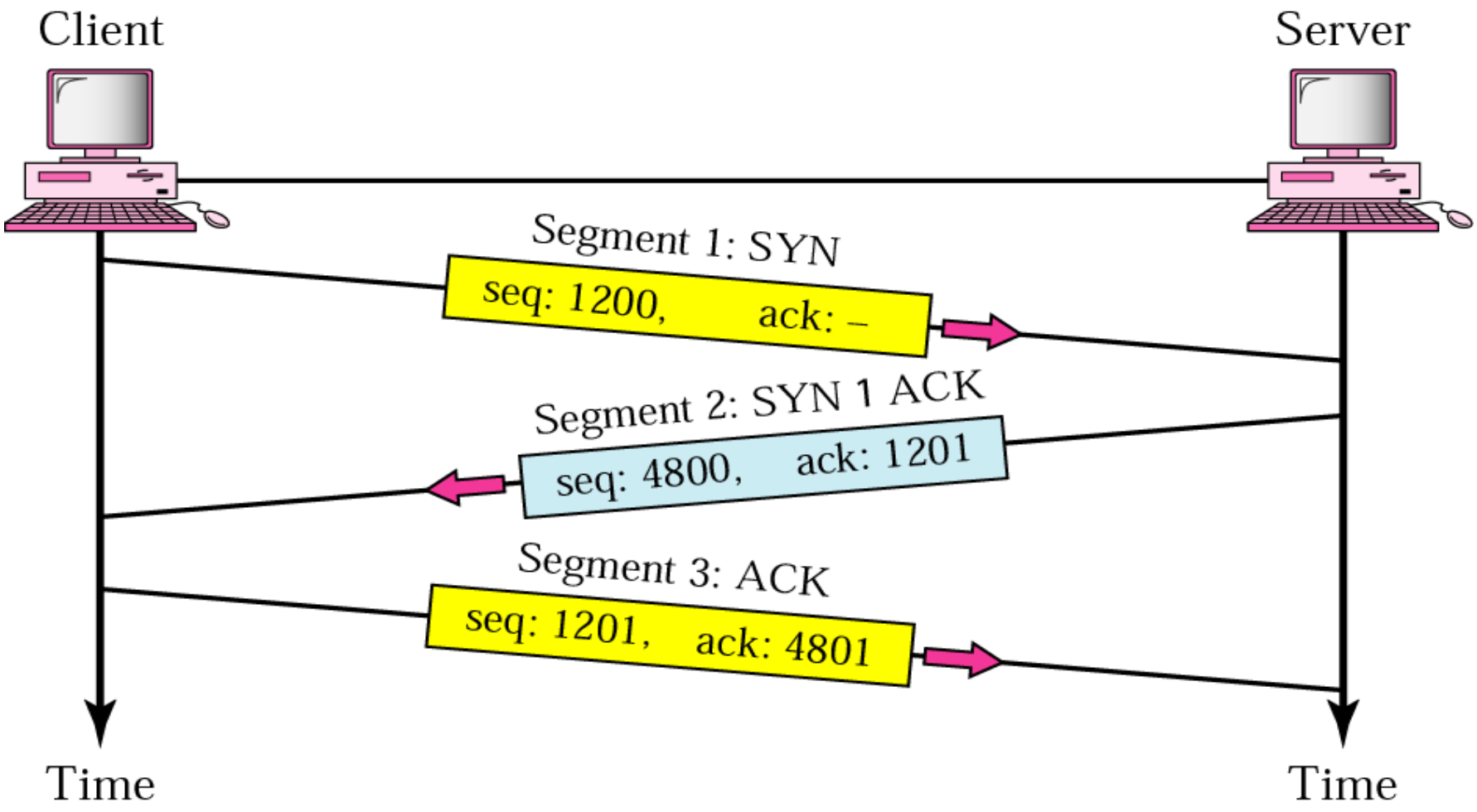
1. Connection establishment

- Passive Open
  - Before a client attempts to connect with a server, the server must first bind to a port to open it up for connections
- Active Open
  - Once the passive open is established, a client may initiate an active open
- To establish a connection, the three-way (or 3-step) handshake occurs

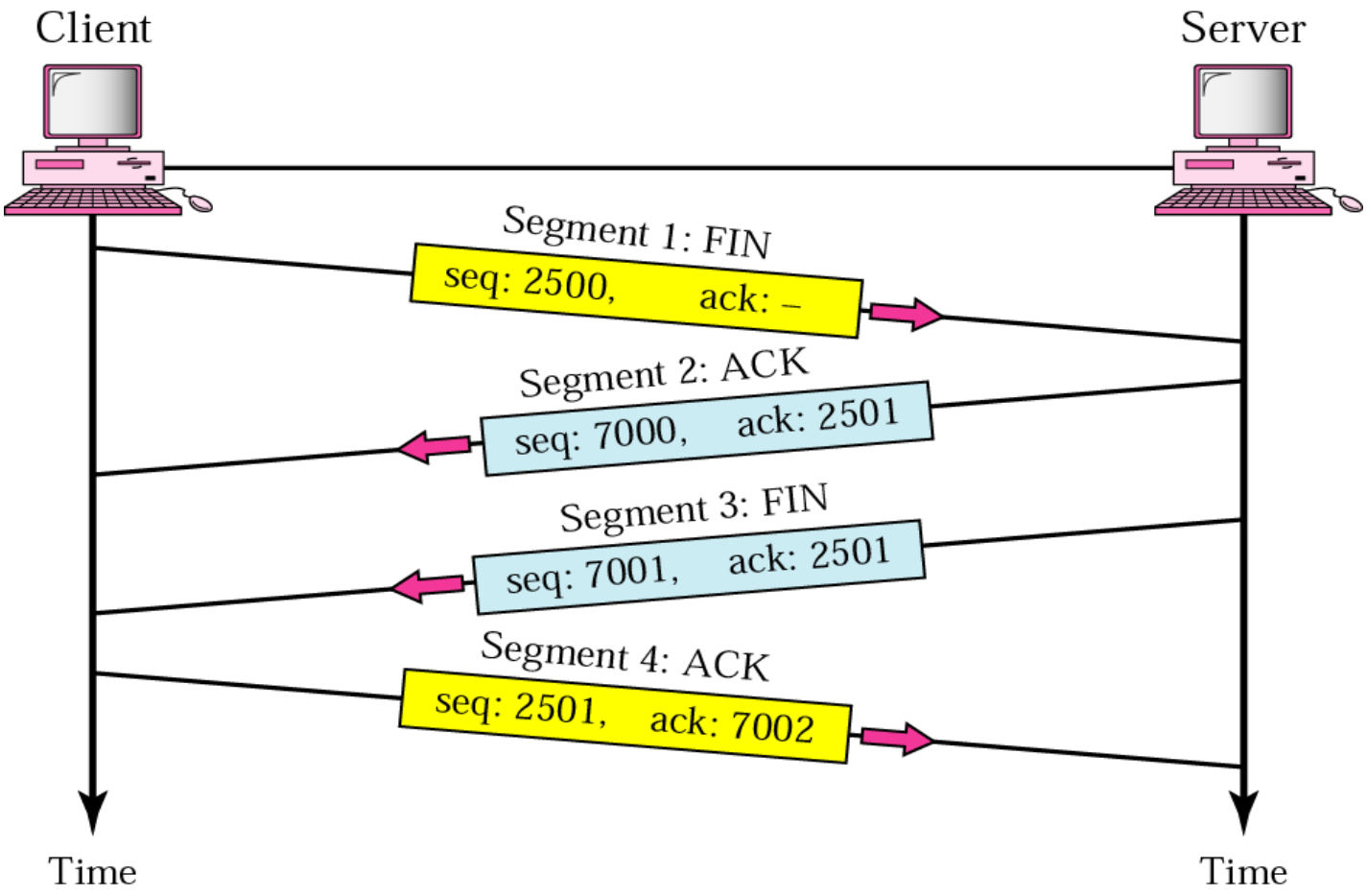
Three-way Handshake

1. The active open is performed by sending a SYN to the server
2. In response, the server replies with a SYN-ACK
3. Finally the client sends an ACK (usually called SYN-ACK-ACK) back to the server

Three-way Handshake

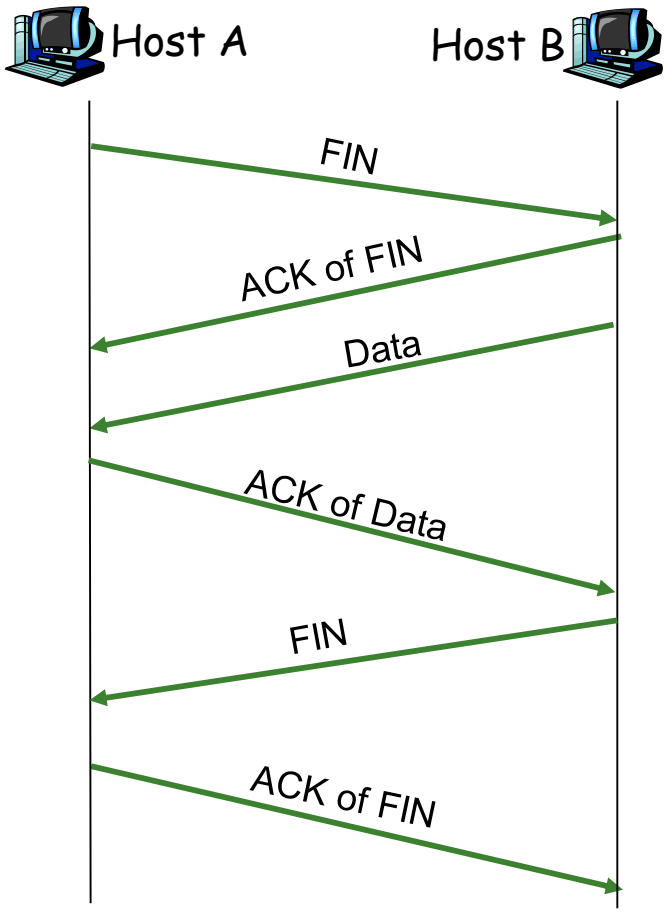


3. Connection Release



Half-Close Connections

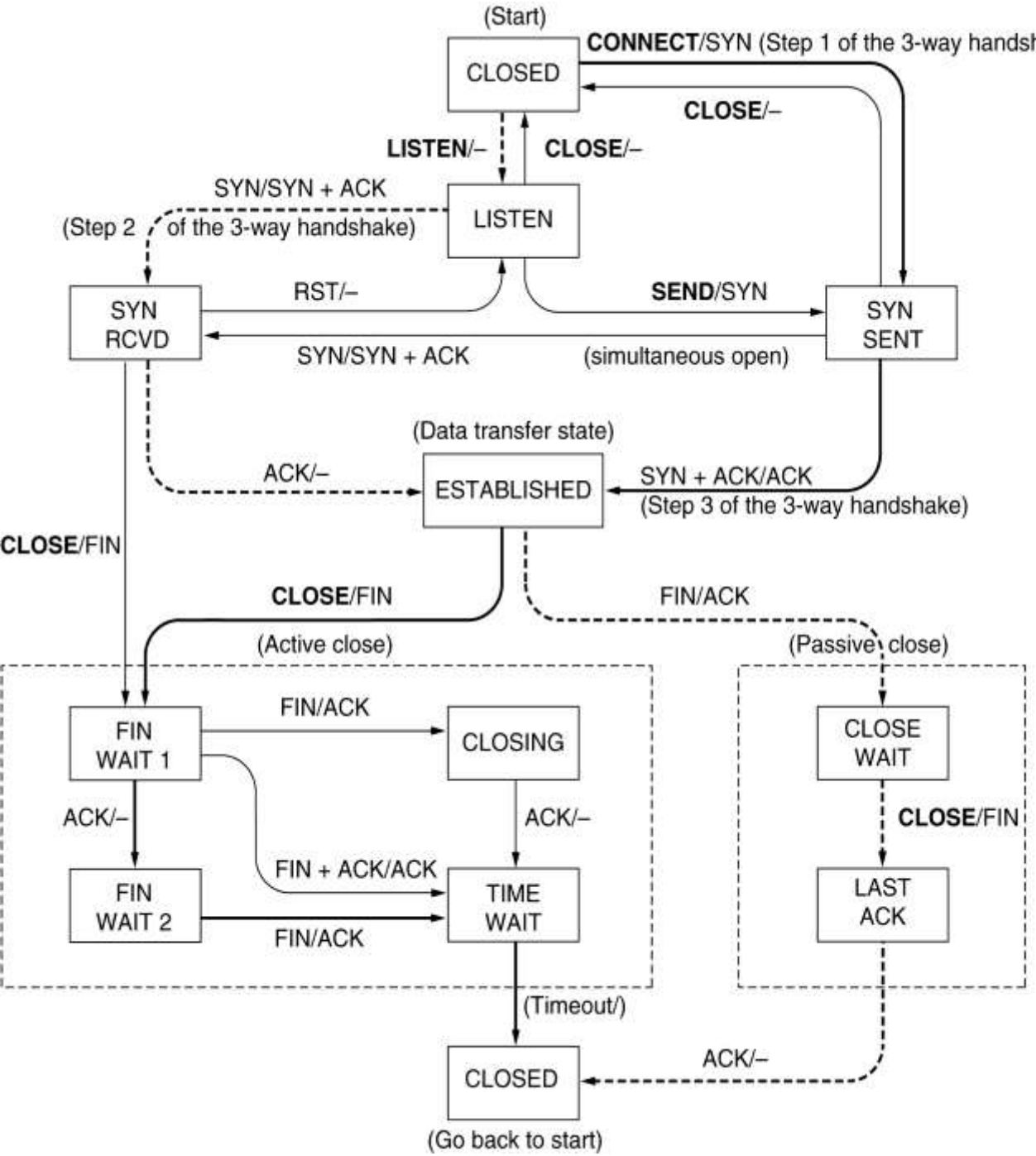
- One end can stop sending data while continuing to receive data



TCP Connection Management Modeling

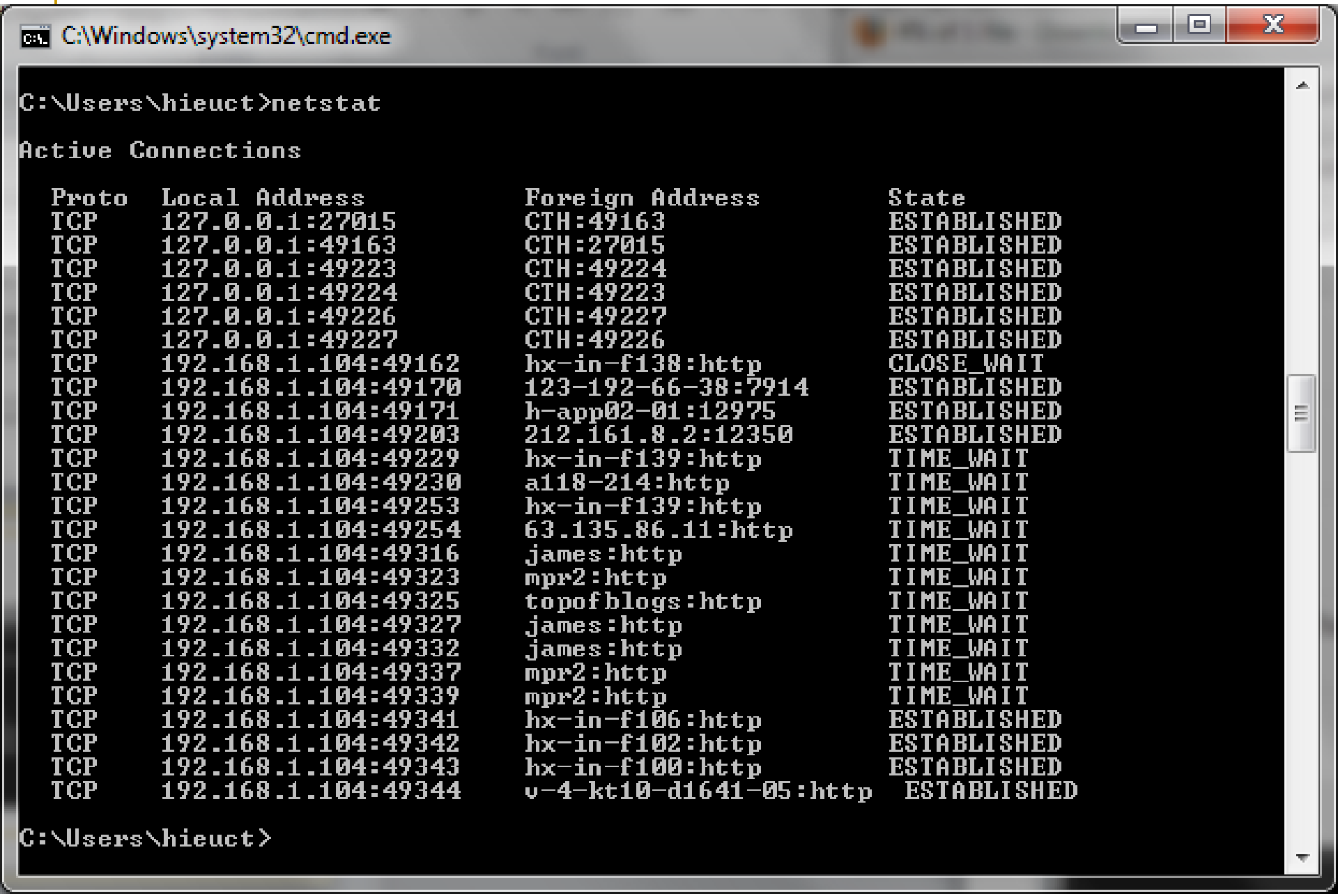
State	Description
CLOSED	No connection is active or pending
LISTEN	The server is waiting for an incoming call
SYN RCVD	A connection request has arrived; wait for ACK
SYN SENT	The application has started to open a connection
ESTABLISHED	The normal data transfer state
FIN WAIT 1	The application has said it is finished
FIN WAIT 2	The other side has agreed to release
TIMED WAIT	Wait for all packets to die off
CLOSING	Both sides have tried to close simultaneously
CLOSE WAIT	The other side has initiated a release
LAST ACK	Wait for all packets to die off

The states used in the TCP connection management finite state machine



TCP connection management finite state machine.

The heavy solid line is the normal path for a client. The heavy dashed line is the normal path for a server. The light lines are unusual events. Each transition is labeled by the event causing it and the action resulting from it, separated by a slash.

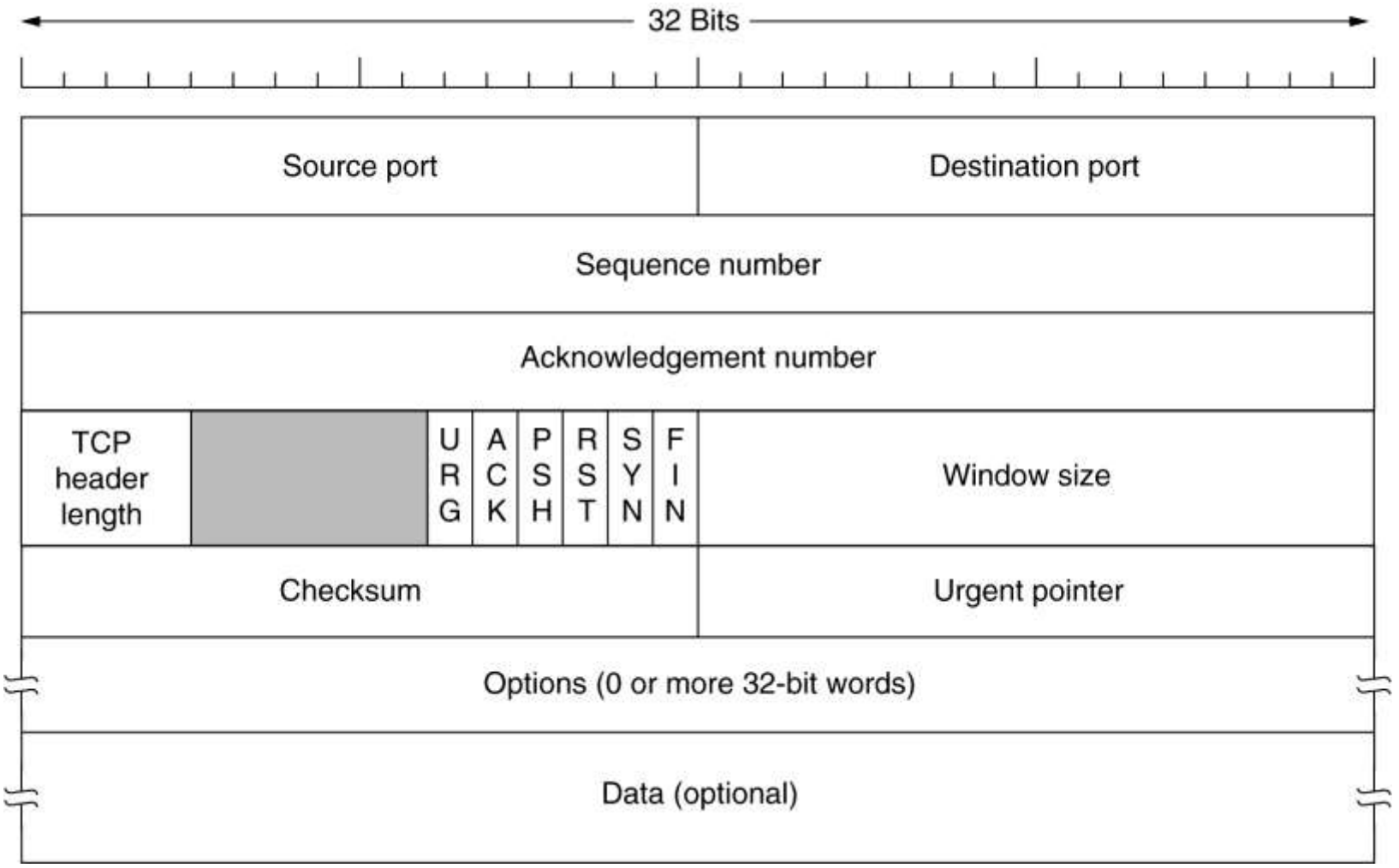

C:\Windows\system32\cmd.exe
C:\Users\hieuct>netstat
Active Connections
Proto Local Address Foreign Address State
TCP 127.0.0.1:27015 CTH:49163 ESTABLISHED
TCP 127.0.0.1:49163 CTH:27015 ESTABLISHED
TCP 127.0.0.1:49223 CTH:49224 ESTABLISHED
TCP 127.0.0.1:49224 CTH:49223 ESTABLISHED
TCP 127.0.0.1:49226 CTH:49227 ESTABLISHED
TCP 127.0.0.1:49227 CTH:49226 ESTABLISHED
TCP 192.168.1.104:49162 hx-in-f138:http CLOSE_WAIT
TCP 192.168.1.104:49170 123-192-66-38:7914 ESTABLISHED
TCP 192.168.1.104:49171 h-app02-01:12975 ESTABLISHED
TCP 192.168.1.104:49203 212.161.8.2:12350 ESTABLISHED
TCP 192.168.1.104:49229 hx-in-f139:http TIME_WAIT
TCP 192.168.1.104:49230 a118-214:http TIME_WAIT
TCP 192.168.1.104:49253 hx-in-f139:http TIME_WAIT
TCP 192.168.1.104:49254 63.135.86.11:http TIME_WAIT
TCP 192.168.1.104:49316 james:http TIME_WAIT
TCP 192.168.1.104:49323 mpr2:http TIME_WAIT
TCP 192.168.1.104:49325 topofblogs:http TIME_WAIT
TCP 192.168.1.104:49327 james:http TIME_WAIT
TCP 192.168.1.104:49332 james:http TIME_WAIT
TCP 192.168.1.104:49337 mpr2:http TIME_WAIT
TCP 192.168.1.104:49339 mpr2:http TIME_WAIT
TCP 192.168.1.104:49341 hx-in-f106:http ESTABLISHED
TCP 192.168.1.104:49342 hx-in-f102:http ESTABLISHED
TCP 192.168.1.104:49343 hx-in-f100:http ESTABLISHED
TCP 192.168.1.104:49344 v-4-kt10-d1641-05:http ESTABLISHED
C:\Users\hieuct>

2. Data Transfer in TCP

- In-Order data transfer
- Retransmission of lost packets
- Discarding duplicate packets
- Error Control
- Flow Control
- Congestion Control

TCP Flow and Congestion Control

TCP Header

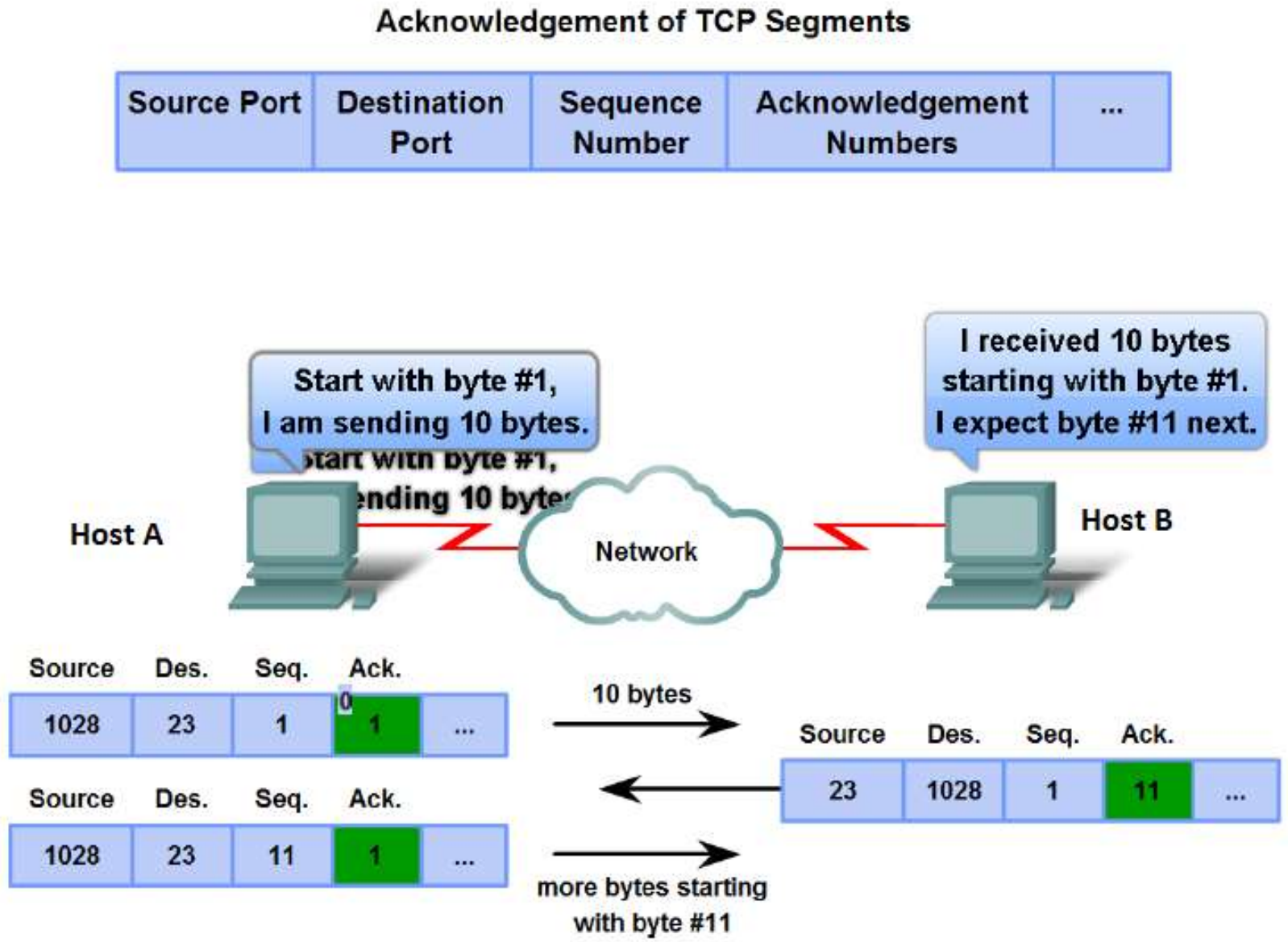

32 Bits
Source port
Destination port
Sequence number
Acknowledgement number
TCP header length
URG
ACK
PSH
RST
SYN
FIN
Window size
Checksum
Urgent pointer
Options (0 or more 32-bit words)
Data (optional)

TCP Header

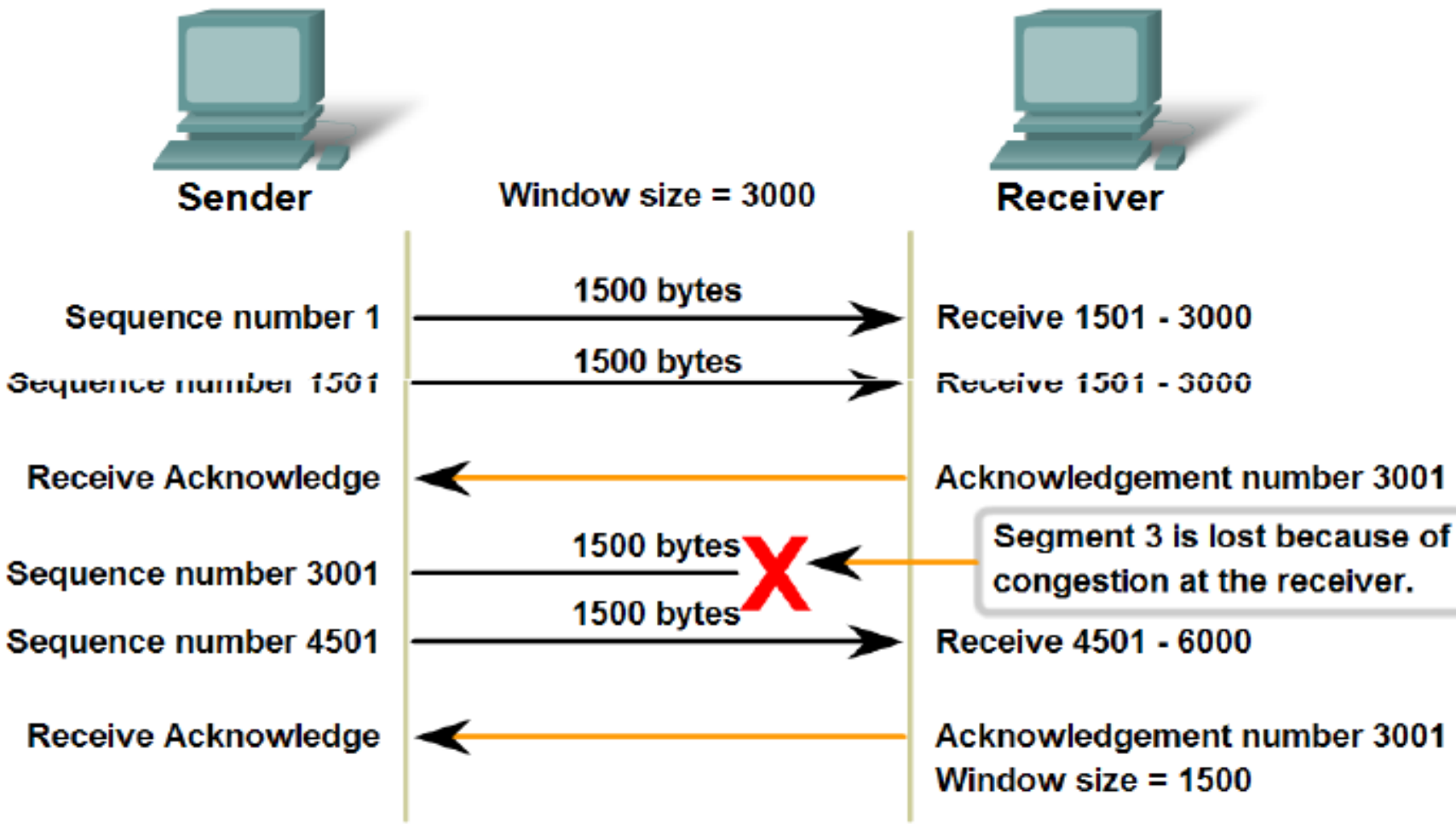
Flag	*Description*
URG	The value of the urgent pointer field is valid
ACK	The value of the acknowledgment field is valid
PSH	Push the data
RST	The connection must be reset
SYN	Synchronize sequence numbers during connection
FIN	Terminate the connection

- **Windows size**: specifies the amount of data we can accept.

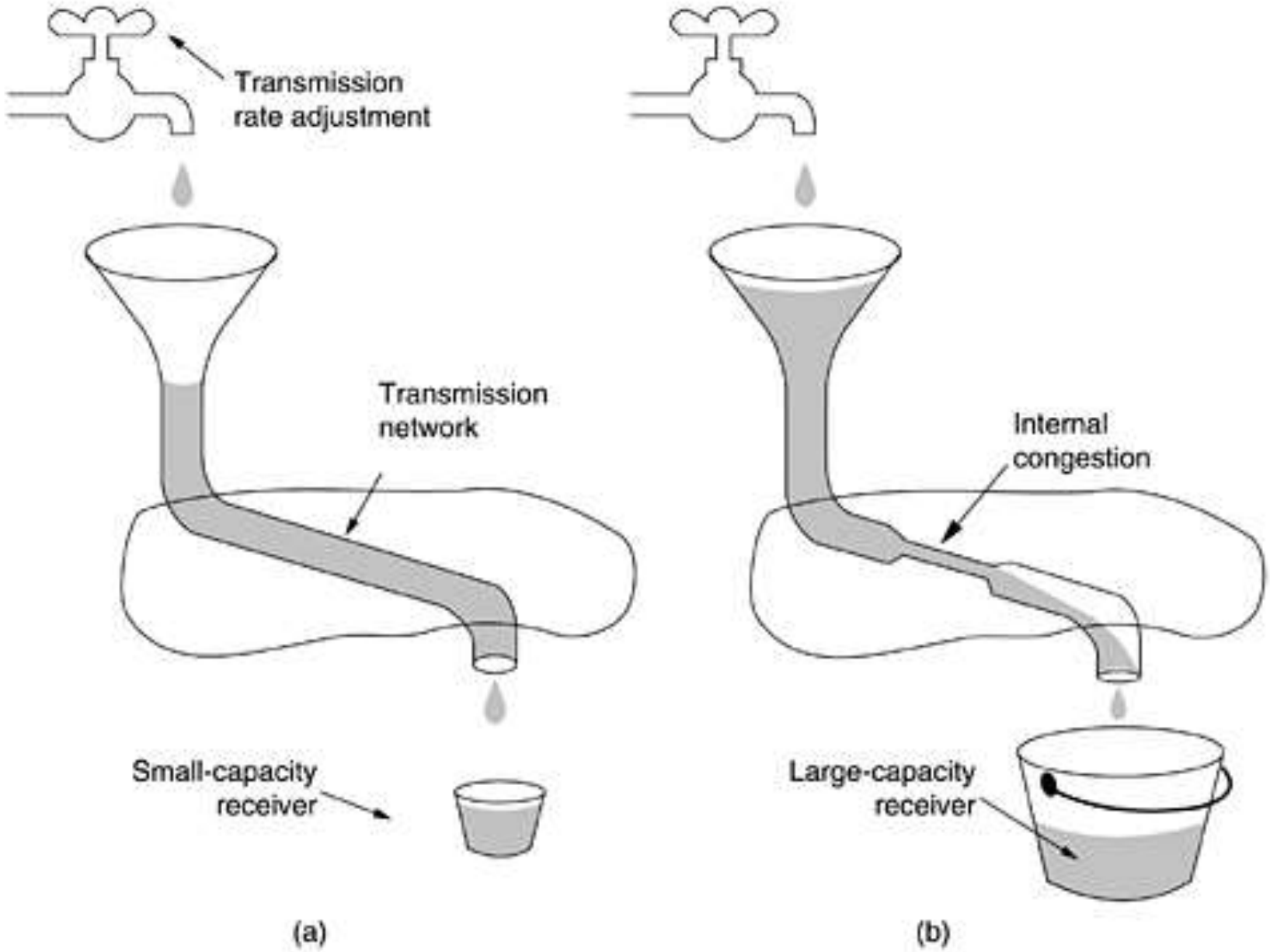
Windowing


Acknowledgement of TCP Segments
Source Port
Destination Port
Sequence Number
Acknowledgement Numbers
...
Start with byte #1, I am sending 10 bytes.
I received 10 bytes starting with byte #1. I expect byte #11 next.
Host A
Network
Host B
Source
Des.
Seq.
Ack.
1028
23
1
0
1
...
10 bytes
Source
Des.
Seq.
Ack.
23
1028
1
11
...
Source
Des.
Seq.
Ack.
1028
23
11
1
...
more bytes starting with byte #11

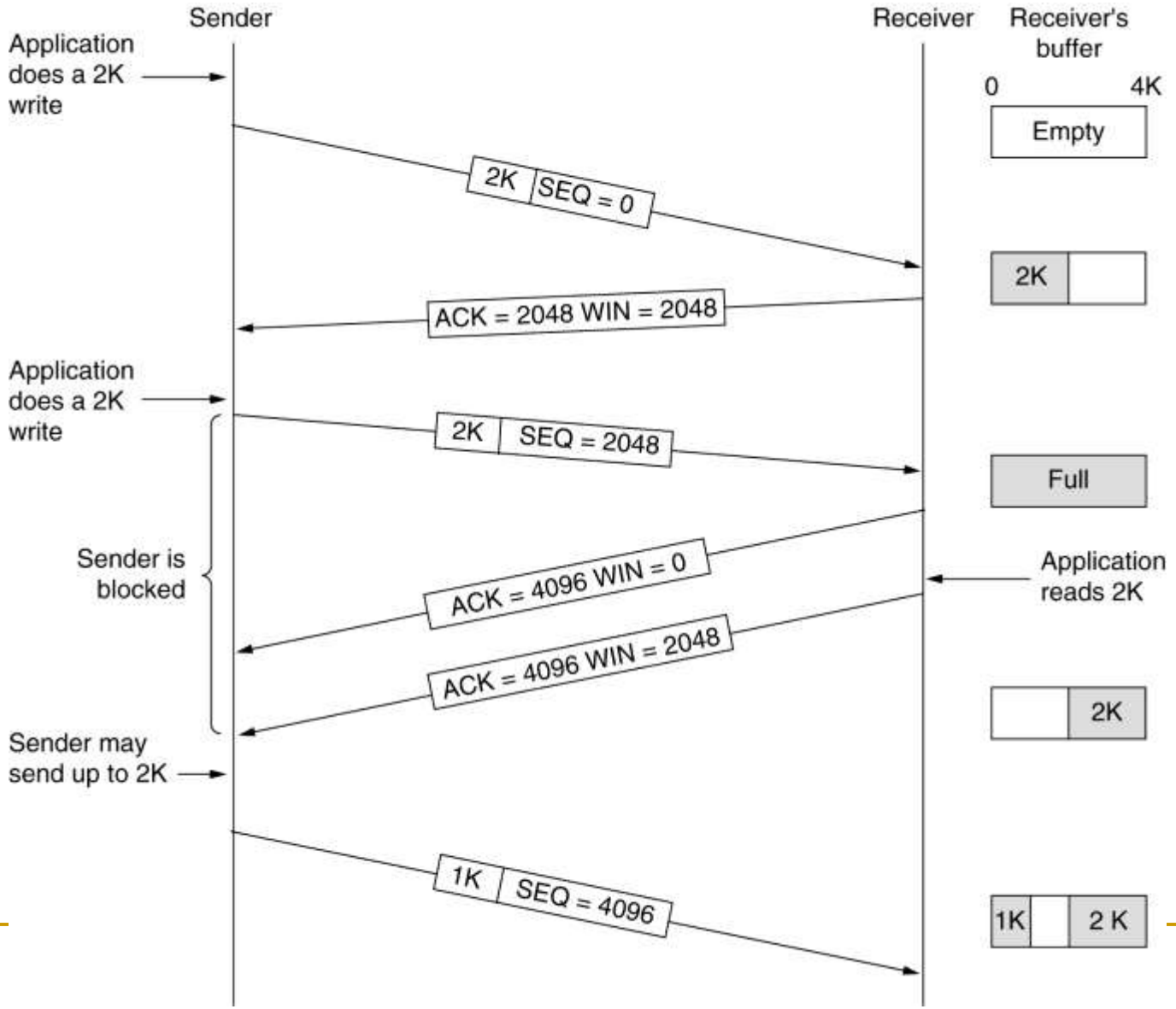
TCP Congestion and Flow Control



If segments are lost because of congestion, the Receiver will acknowledge the last received sequential segment and reply with a reduced window size.


Transmission
rate adjustment
Transmission
network
Small-capacity
receiver
(a)
Internal
congestion
Large-capacity
receiver
(b)

Summary of Flow Control



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's sliding windows are byte oriented.

EXAMPLE 1

What is the value of the receiver window (rwnd) for host A if the receiver, host B, has a buffer size of 5,000 bytes and 1,000 bytes of received and unprocessed data?

Solution

The value of rwnd = 5,000 − 1,000 = 4,000. Host B can receive only 4,000 bytes of data before overflowing its buffer. Host B advertises this value in its next segment to A.

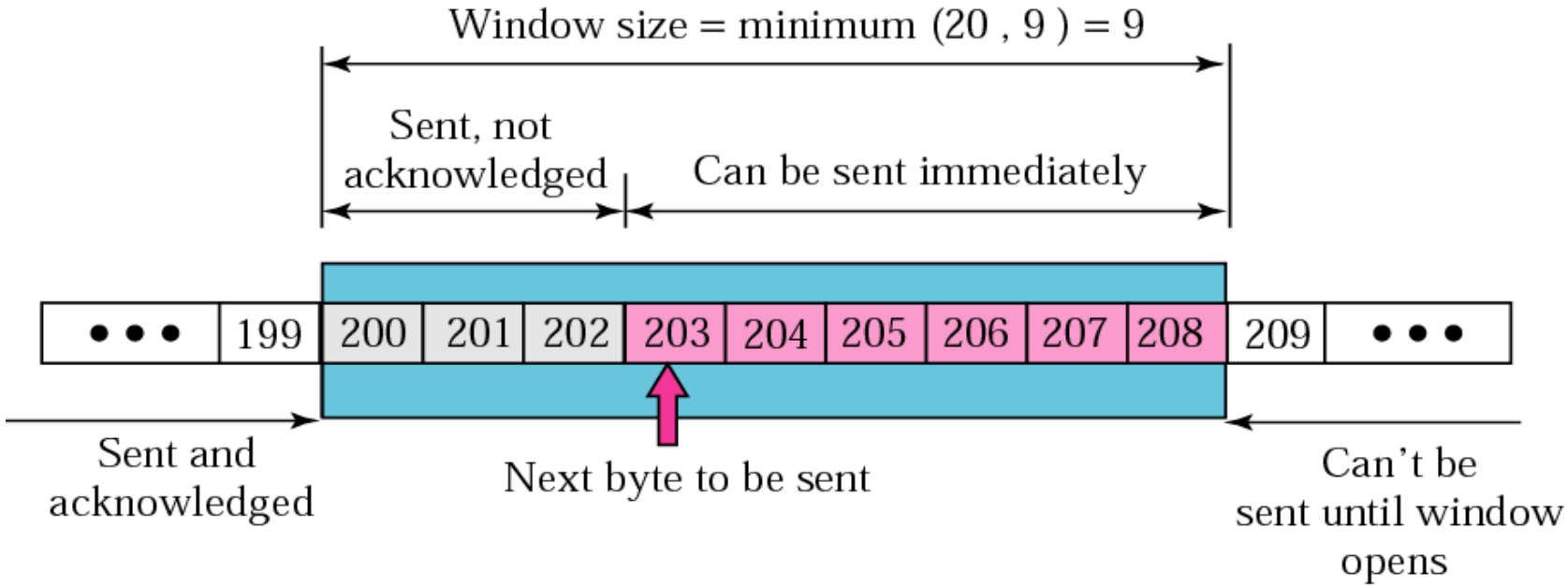
EXAMPLE 2

What is the size of the window for host A if the value of rwnd is 3,000 bytes and the value of cwnd is 3,500 bytes?

Solution

The size of the window is the smaller of rwnd and cwnd, which is 3,000 bytes.

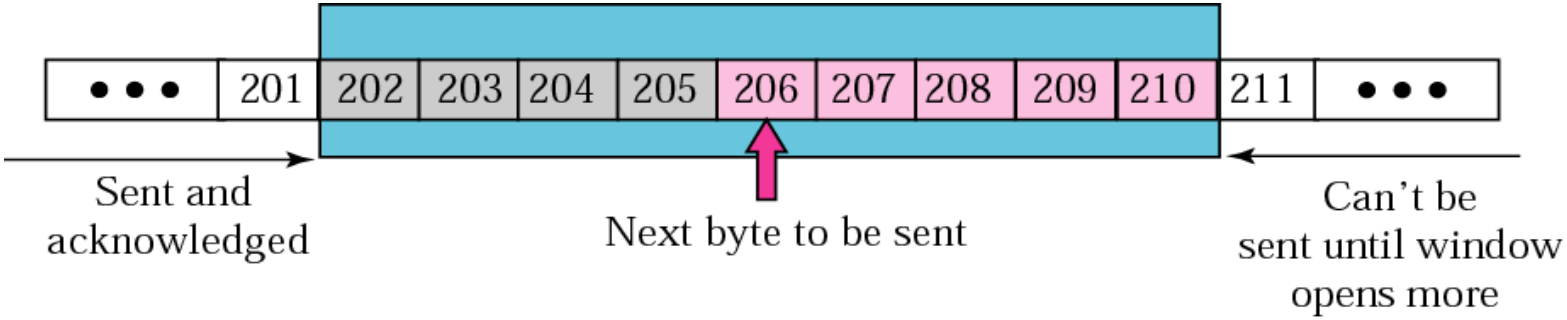
EXAMPLE 3



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.

EXAMPLE 4

The server receives a packet with an acknowledgment value of 202 and an rwnd of 9. The host has already sent bytes 203, 204, and 205. The value of cwnd is still 20. Show the new window.

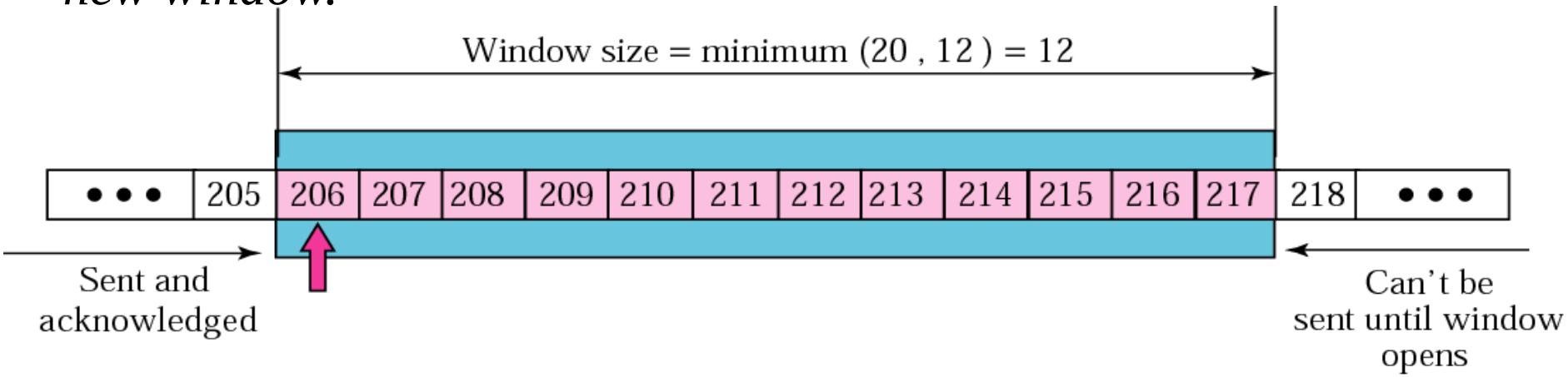


Solution

Figure shows the new window. Note that this is a case in which the window closes from the left and opens from the right by an equal number of bytes; the size of the window has not been changed. The acknowledgment value, 202, declares that bytes 200 and 201 have been received and the sender needs not worry about them; the window can slide over them.

EXAMPLE 5

In this figure the sender receives a packet with an acknowledgment value of 206 and an rwnd of 12. The host has not sent any new bytes. The value of cwnd is still 20. Show the new window.

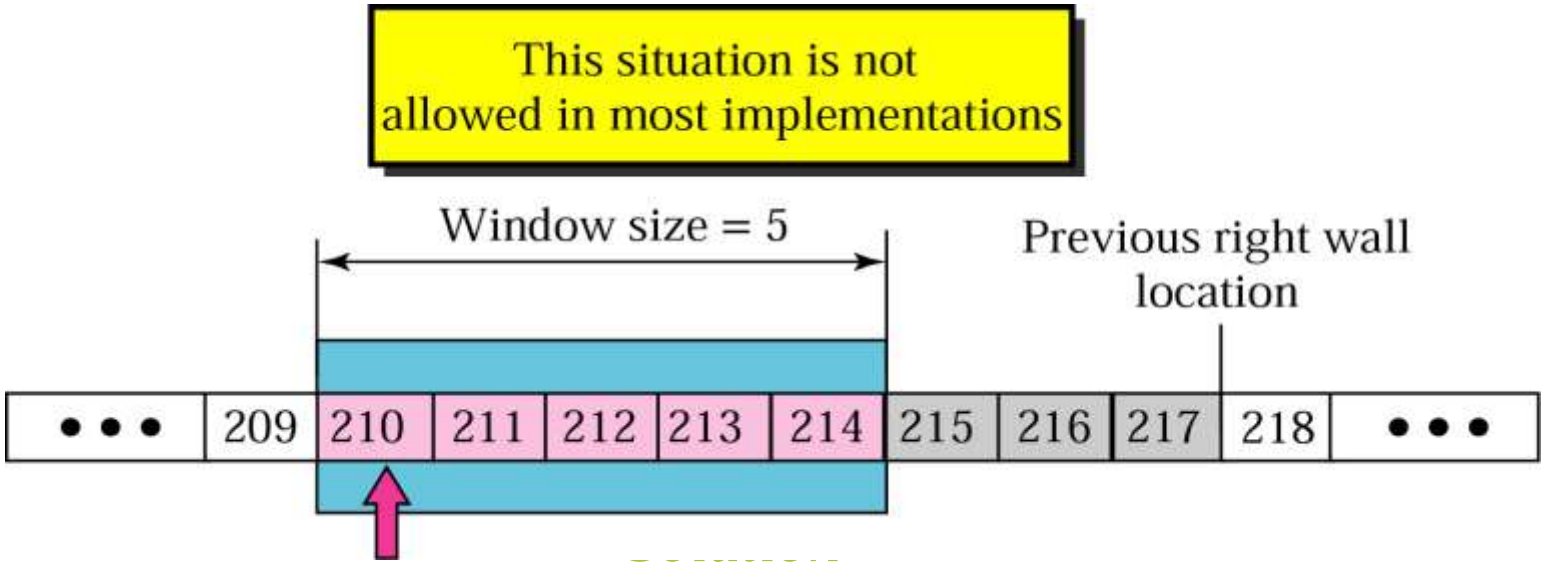


Solution

The value of rwnd is less than cwnd, so the size of the window is 12. Figure shows the new window. Note that the window has been opened from the right by 7 and closed from the left by 4; the size of the window has increased.

EXAMPLE 6

In Figure, the host receives a packet with an acknowledgment value of 210 and an rwnd of 5. The host has sent bytes 206, 207, 208, and 209. The value of cwnd is still 20. Show the new window.



The value of rwnd is less than cwnd, so the size of the window is 5. Figure shows the situation. Note that this is a case not allowed by most implementations. Although the sender has not sent bytes 215 to 217, the receiver does not know this.

Window Shutdown

- rwnd = 0
  - When receiver does not wants to receive data for some time
  - Sender stops sending data until the new advertisement arrives from receiver
    - If there is no data, receiver still sends an ACK with new rwnd value

TCP – Conclusion

- Reliable connection-oriented service
- HTTP, FTP, POP3… are protocols based on TCP

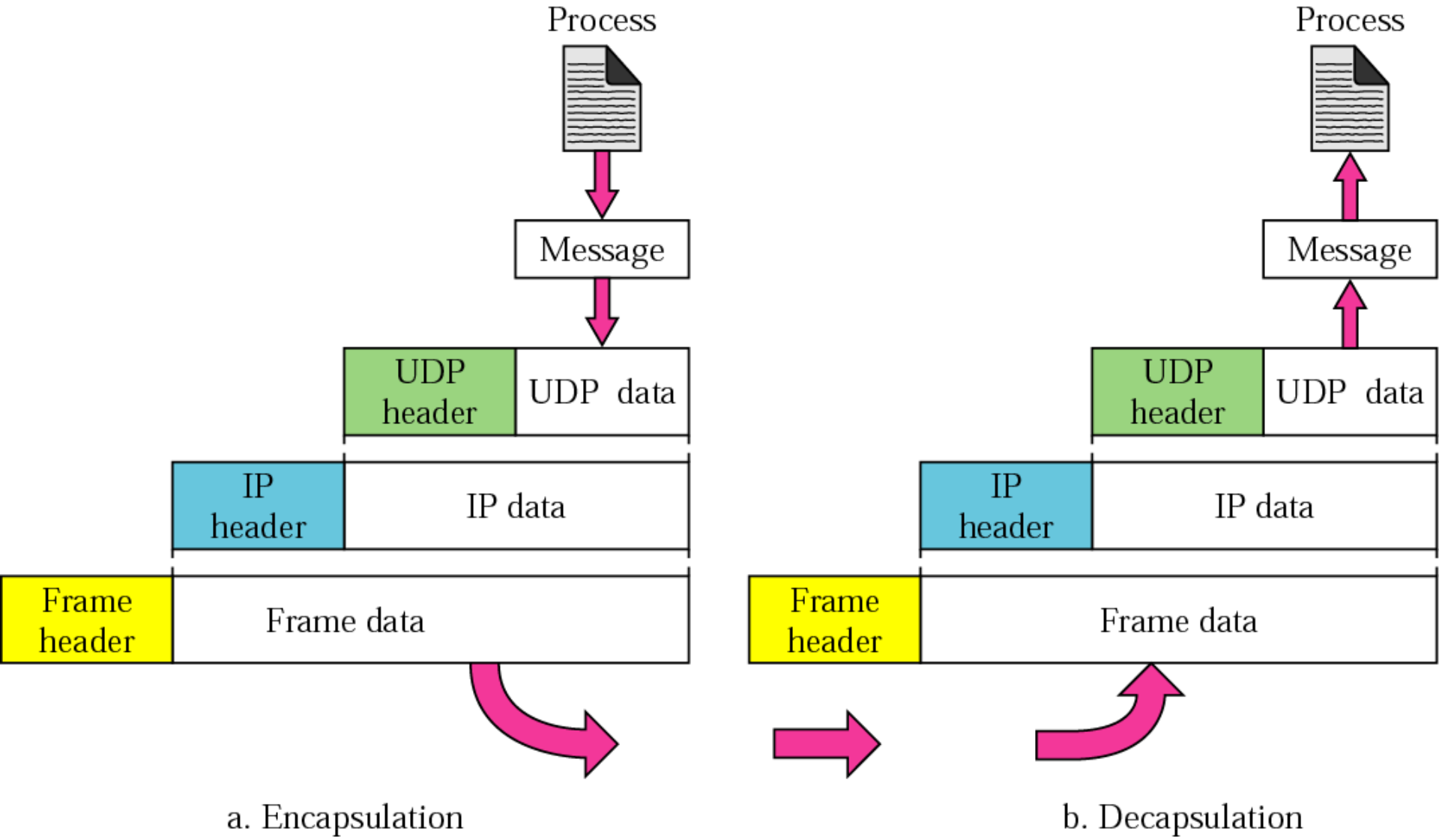
UDP
(User Datagram Protocol)

UDP

- Simplicity itself
  - No connection setup is needed in advance
  - No release at the end
  - No flow control
  - No retransmissions
  - Error Control is optional

  -> don’t waste network bandwidth

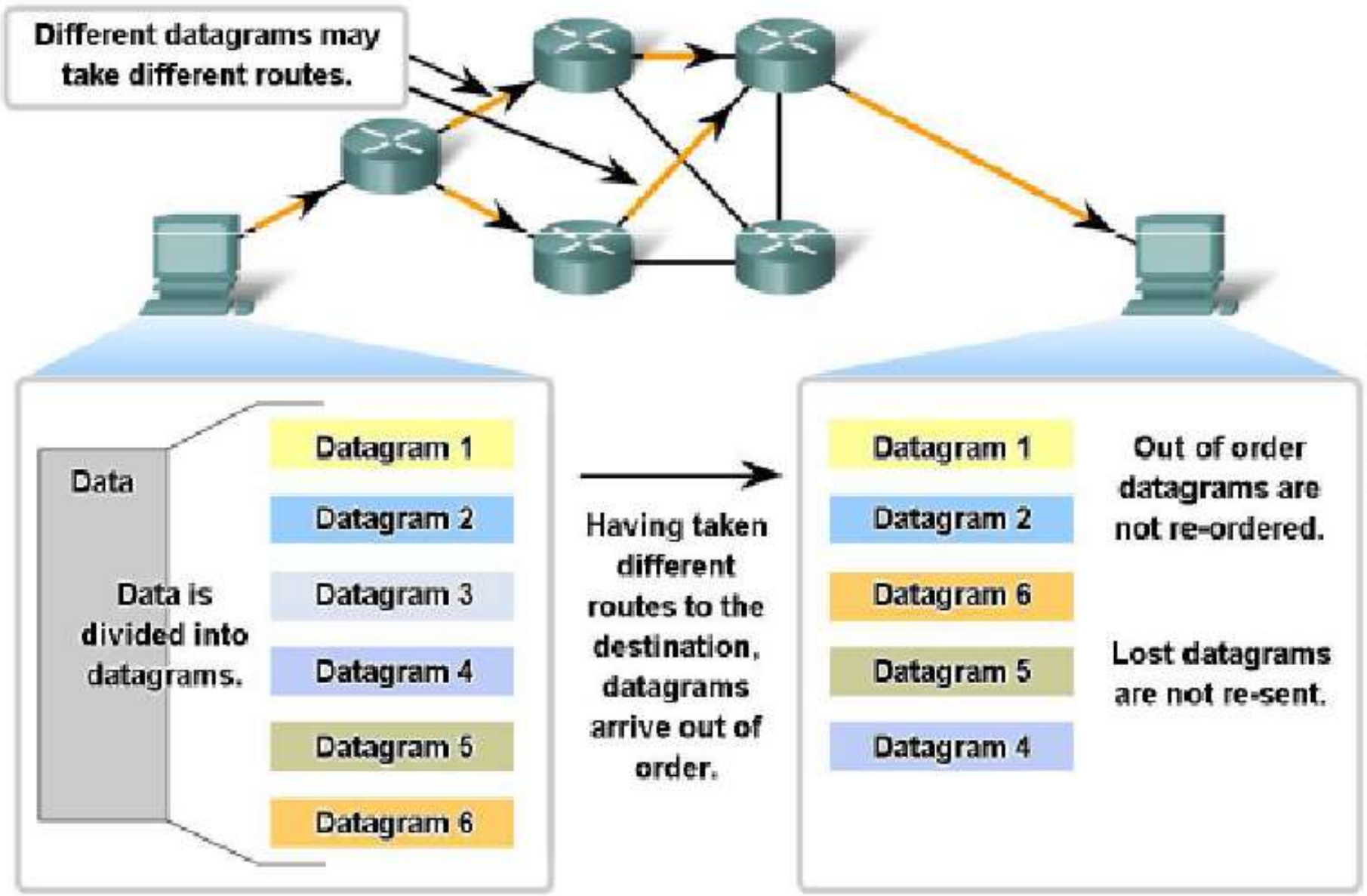
UDP: Encapsulation , Decapsulation



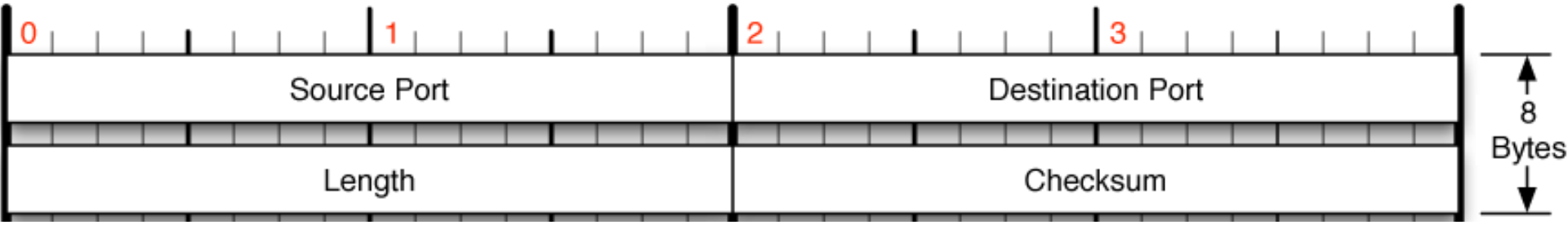
UDP

- Sender: Data is divided into datagrams. Sender just sends datagrams.
- Receiver: Datagram are merged based on the arriving order.
  - Lost datagrams are not resent.

UDP: Connectionless and Unreliable

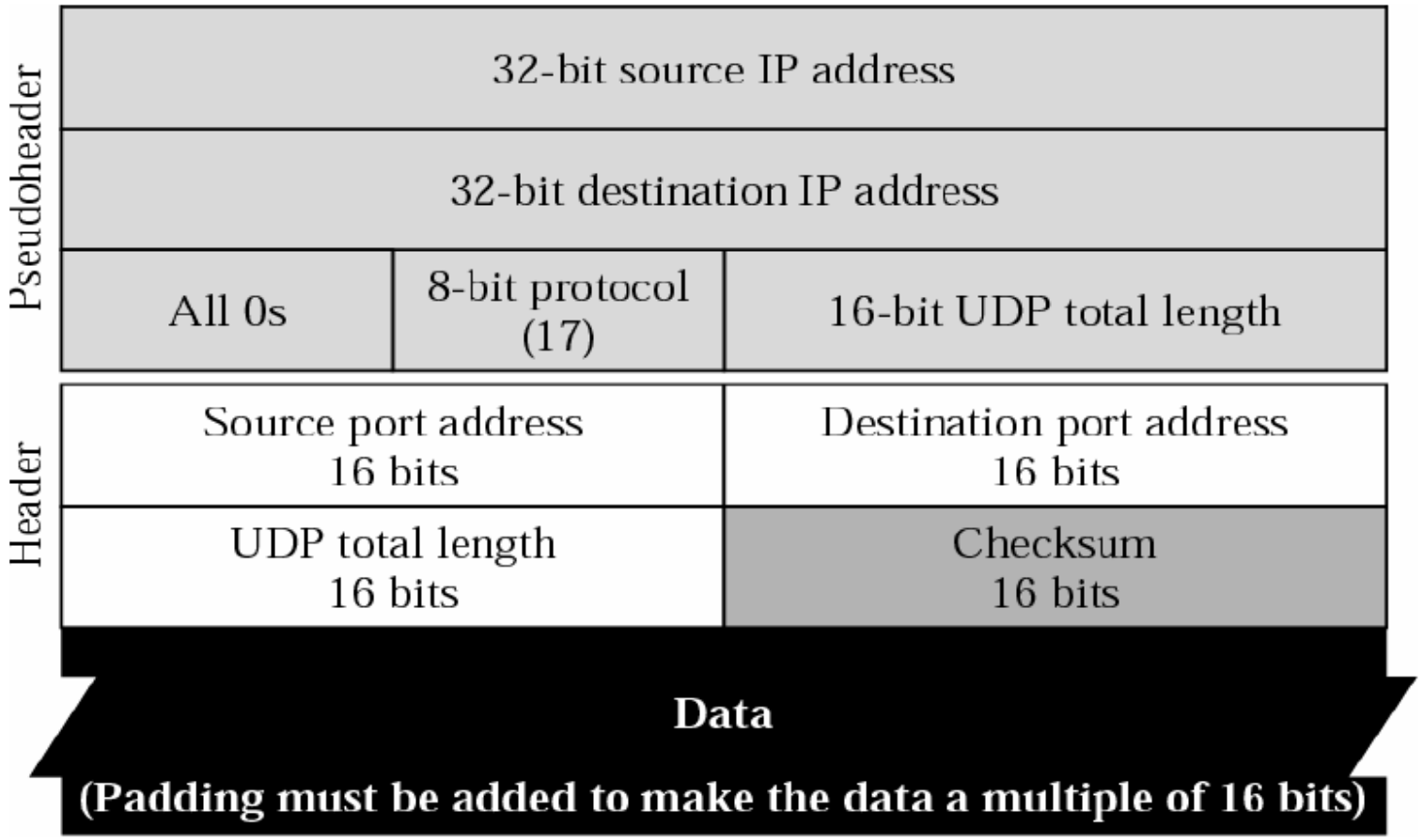


UDP Header



- Length
  - Length of the entire UDP datagram (Header + Payload)
- Checksum
  - Checksum of (Pseudo-header + Header + Payload)
  - Checksum field is optional in UDP
    - If it is not used, it is set to a value of all zeroes

Pseudo-header



UDP Checksum

- The checksum is computed over the combination of the pseudo header and the real UDP message, and the value is placed into the *Checksum* field
- The pseudo header is used only for this calculation and is then discarded; it is not actually transmitted

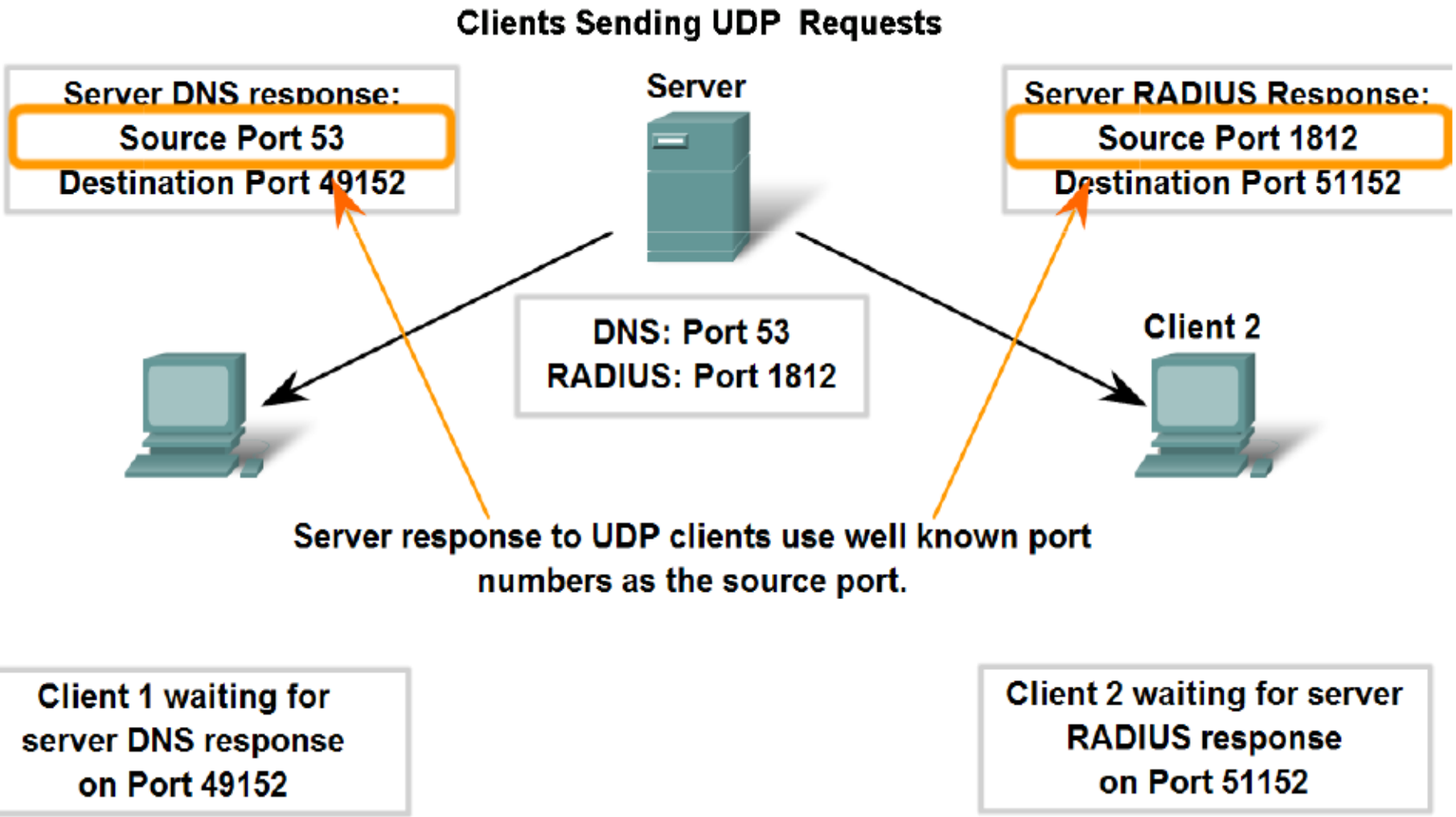
+	Bits 0 - 7	8 - 15	16 - 23	24 - 31
0	Source address			
32	Destination address			
64	Zeros	Protocol	UDP length	
96	Source Port		Destination Port	
128	Length		Checksum	
160	Data			

Pseudo-header

UDP Header

UDP Data Payload

UDP client processes



UDP – Conclusion

- Unreliable connectionless service
- Used by some applications:
  - Domain Name System (DNS)
  - Simple Network Management Protocol (SNMP)
  - Dynamic Host Configuration Protocol (DHCP)
  - Routing Information Protocol (RIP)
  - Trivial File Transfer Protocol (TFTP)
  - Game online
  - video conference, livestream video…

UDP – Conclusion

- Used for
  1. Short requests and replies
  2. Where performance is more important than correctness
  3. For multicast or broadcast applications

THANK YOU!

Questions & Comments