Transport Layer:
Process-to-Process Delivery
(UDP, TCP)

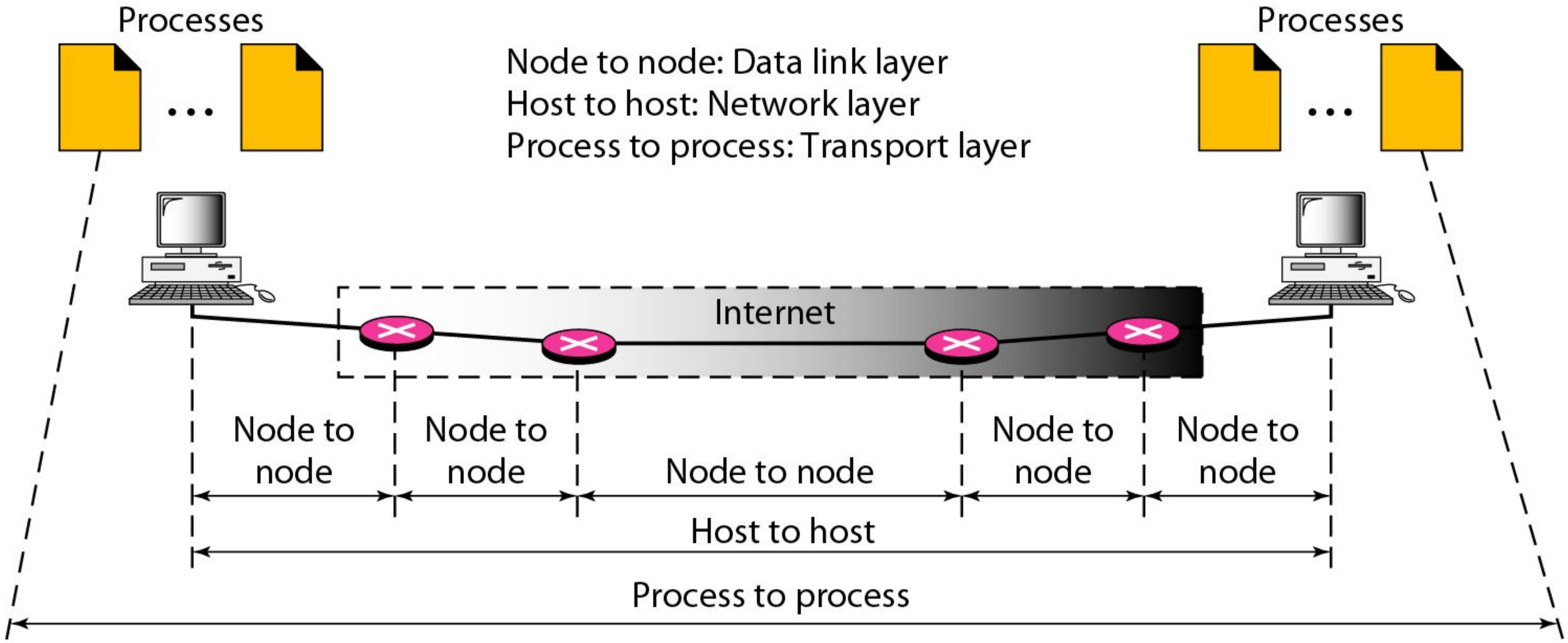
PROCESS-TO-PROCESS DELIVERY

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, as we will see later.

Topics discussed in this section:

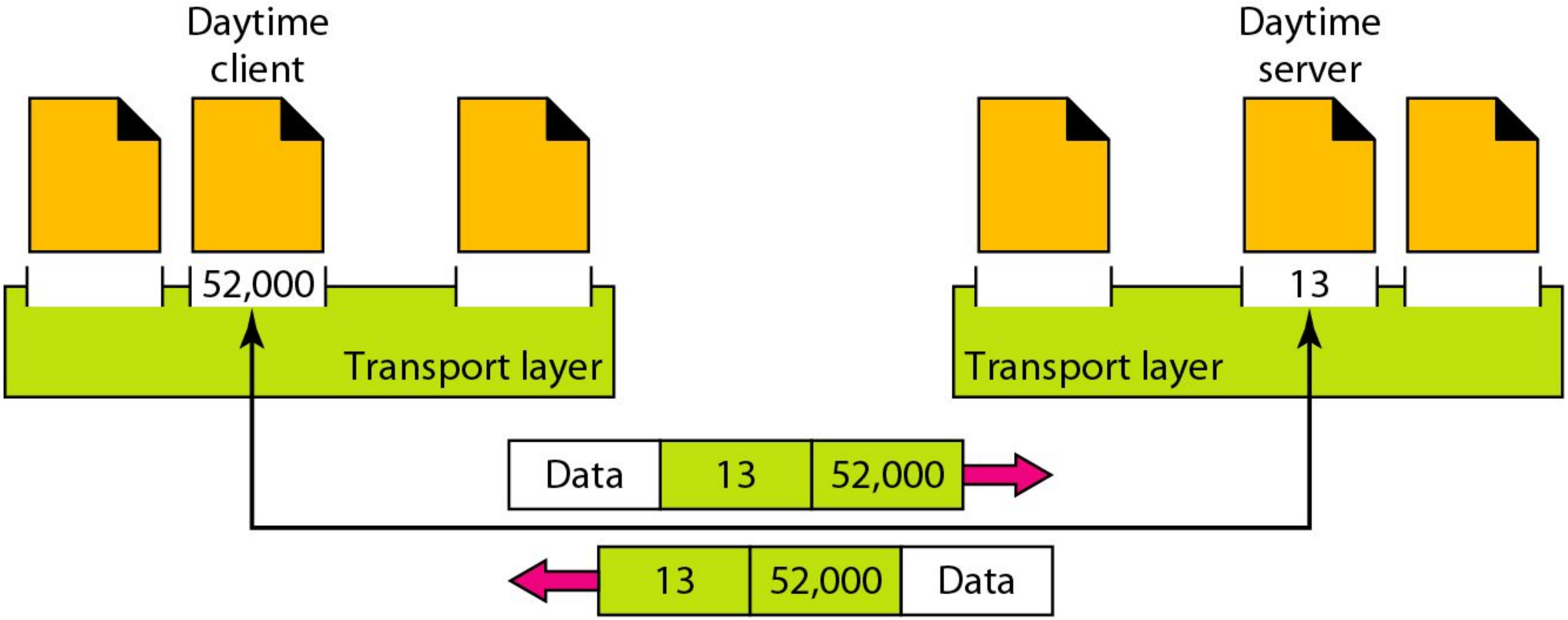
- Client/Server Paradigm
- Multiplexing and Demultiplexing
- Connectionless Versus Connection-Oriented Service
- Reliable Versus Unreliable
- Three Protocols

Types of data deliveries

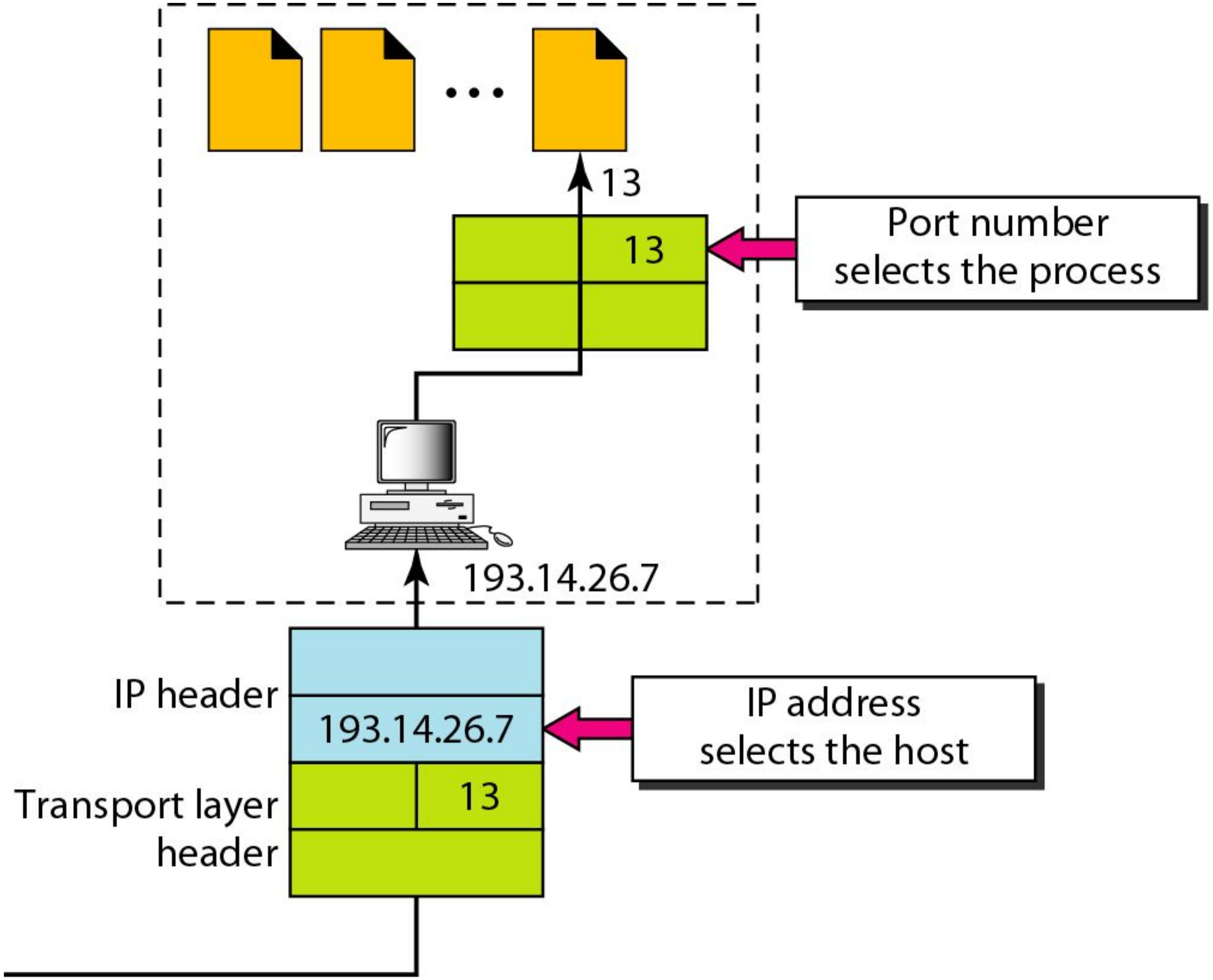


The transport layer is responsible for process-to-process delivery.

Port numbers



IP addresses versus port numbers



Internet Corporation for Assigned Names and Numbers

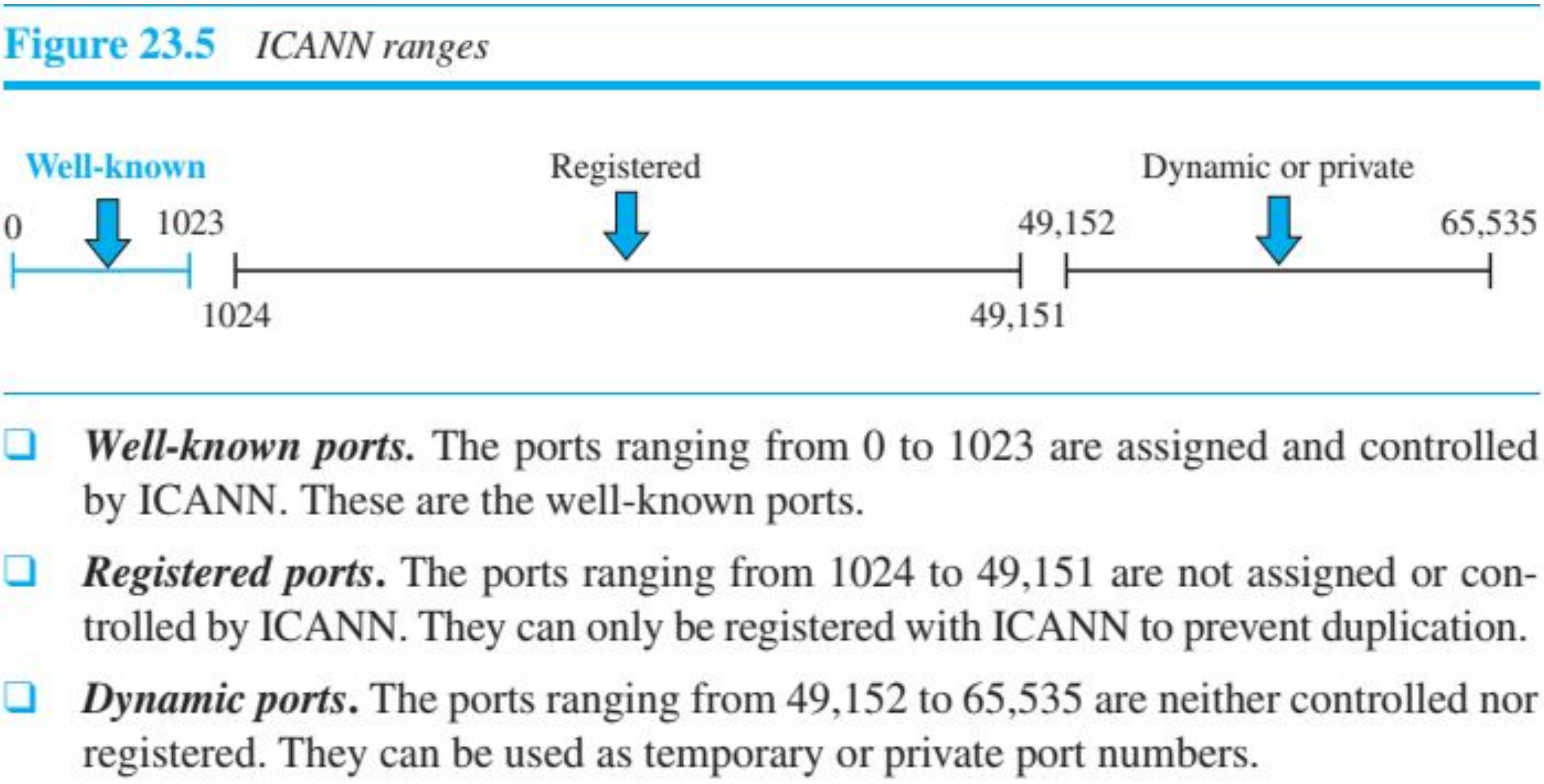


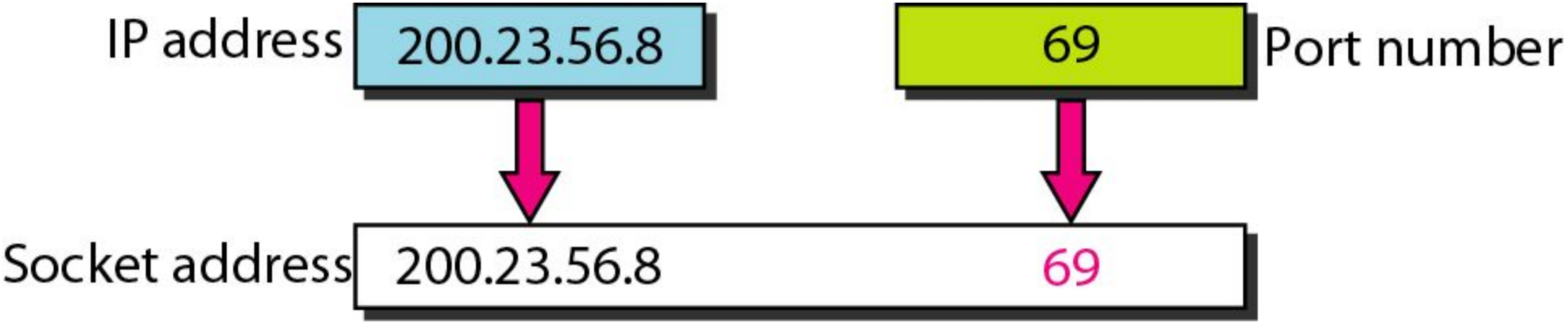
Figure 23.5 *ICANN ranges*

Well-known
Registered
Dynamic or private
0
1023
1024
49,152
49,151
65,535

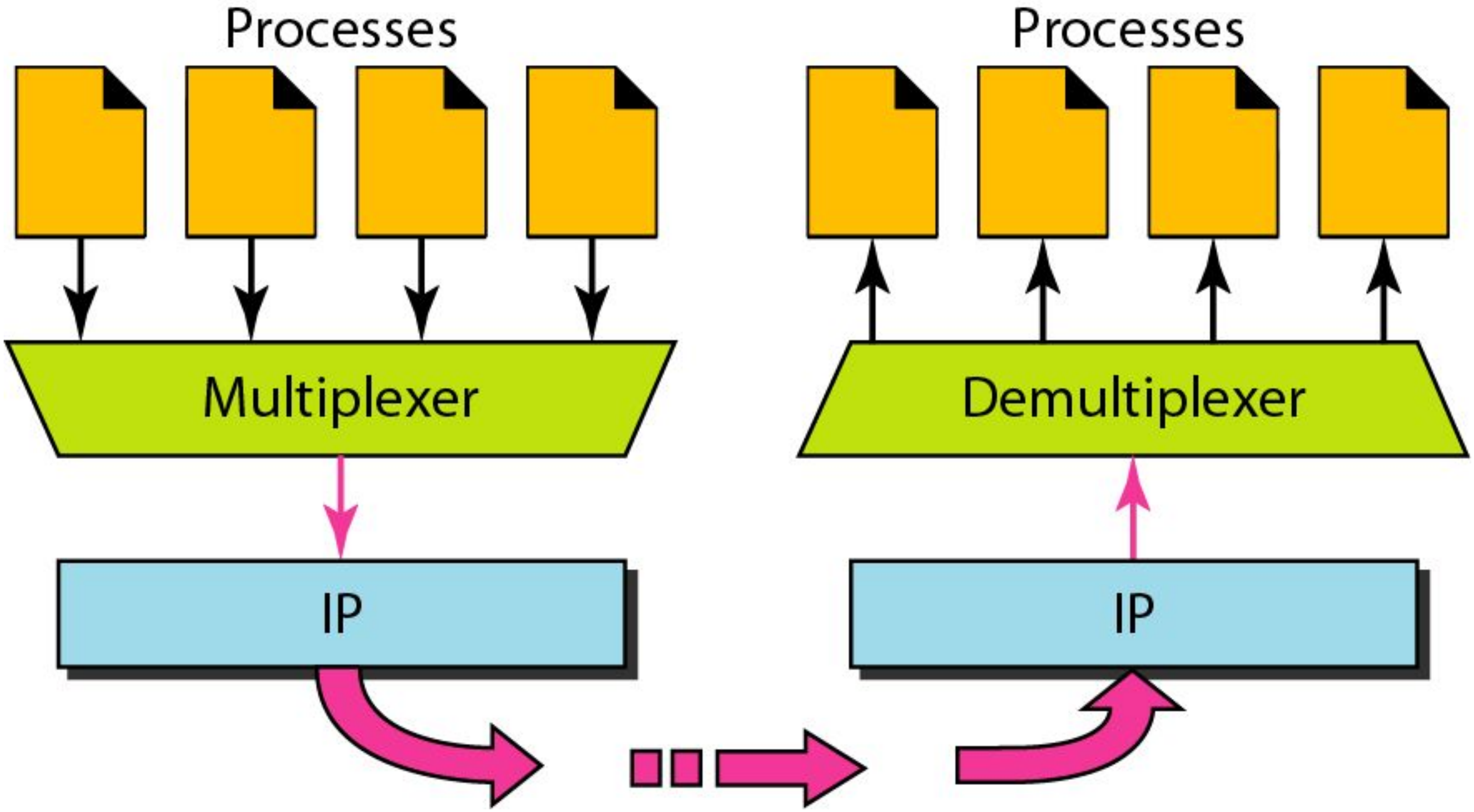
- ❑ ***Well-known ports.*** The ports ranging from 0 to 1023 are assigned and controlled by ICANN. These are the well-known ports.
- ❑ ***Registered ports.*** The ports ranging from 1024 to 49,151 are not assigned or controlled by ICANN. They can only be registered with ICANN to prevent duplication.
- ❑ ***Dynamic ports.*** The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used as temporary or private port numbers.

Socket address

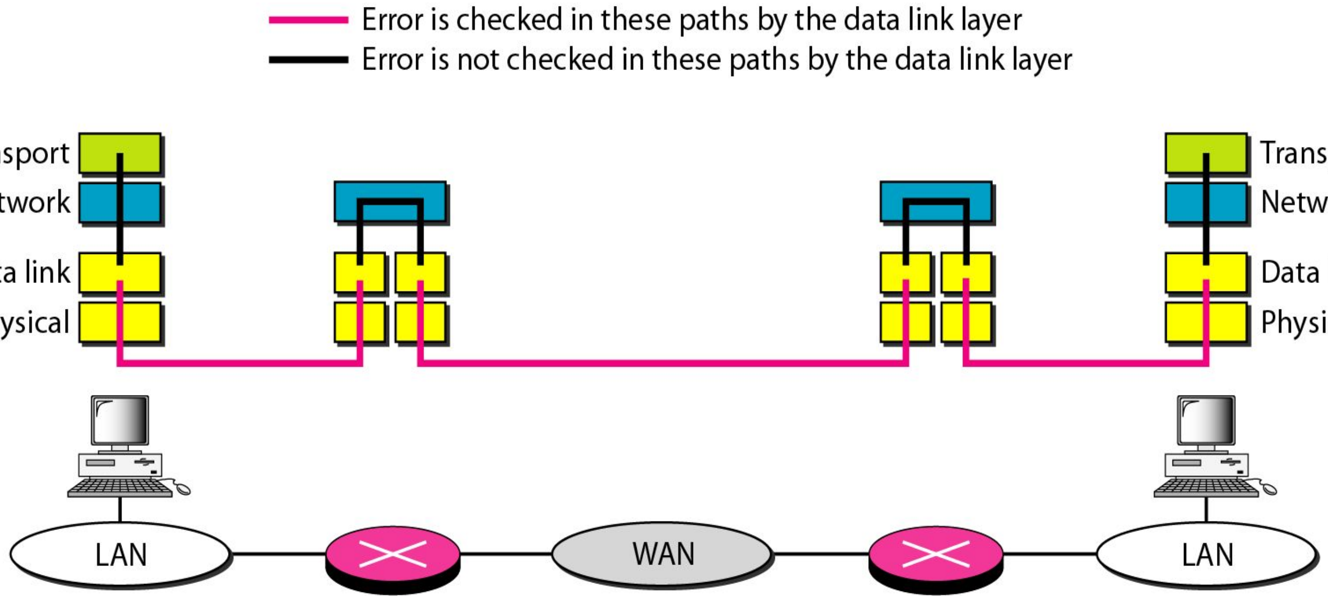
The combination of an IP address and a port number is called a ***socket address.***



Multiplexing and Demultiplexing



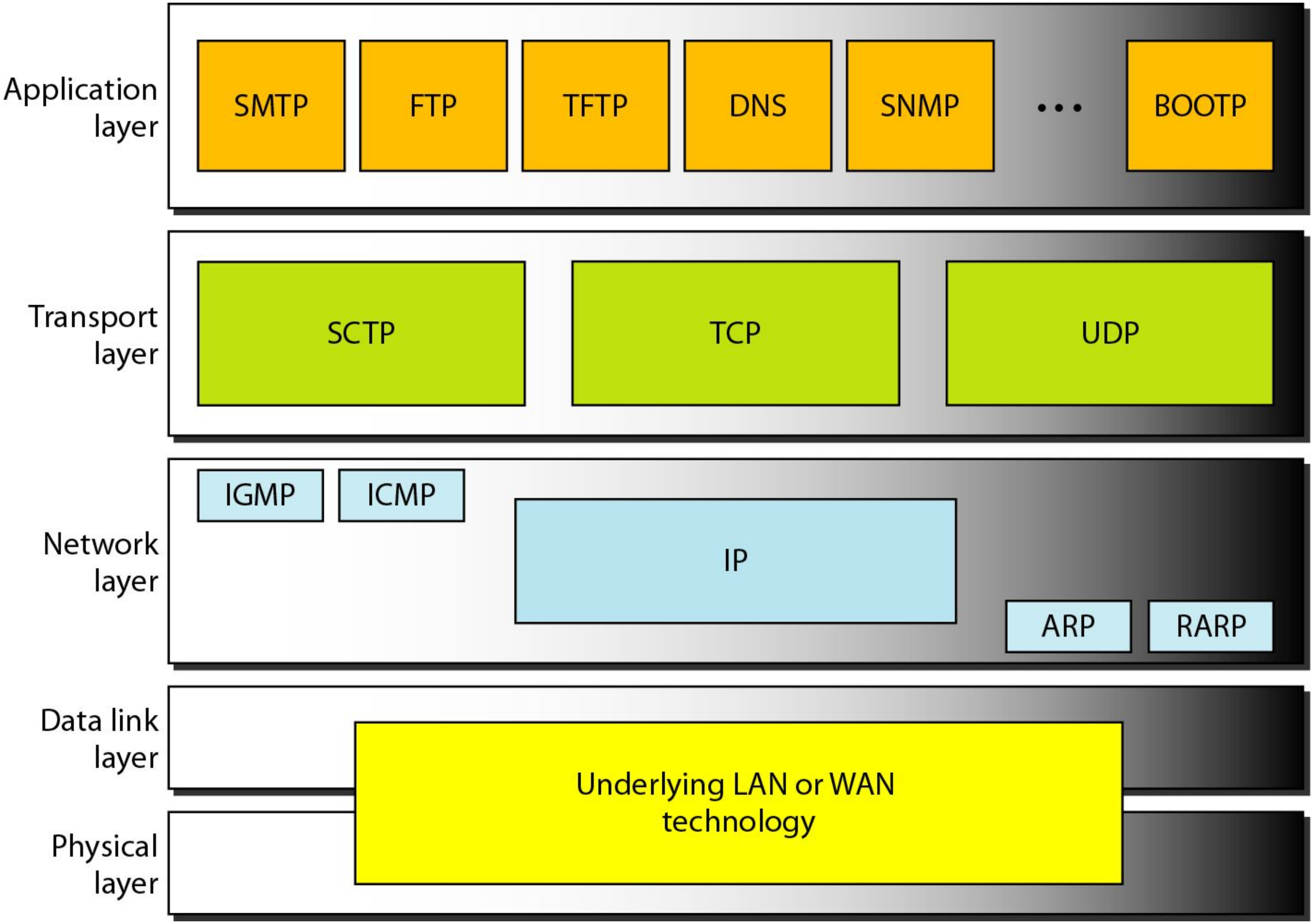
Error control


Error is checked in these paths by the data link layer
Error is not checked in these paths by the data link layer
Transport
Network
Data link
Physical
Transport
Network
Data link
Physical
LAN
WAN
LAN

Error control at the transport layer is responsible for

1. Detecting and discarding corrupted packets.
2. Keeping track of lost and discarded packets and resending them.
3. Recognizing duplicate packets and discarding them.
4. Buffering out-of-order packets until the missing packets arrive.

Position of UDP, TCP in TCP/IP suite



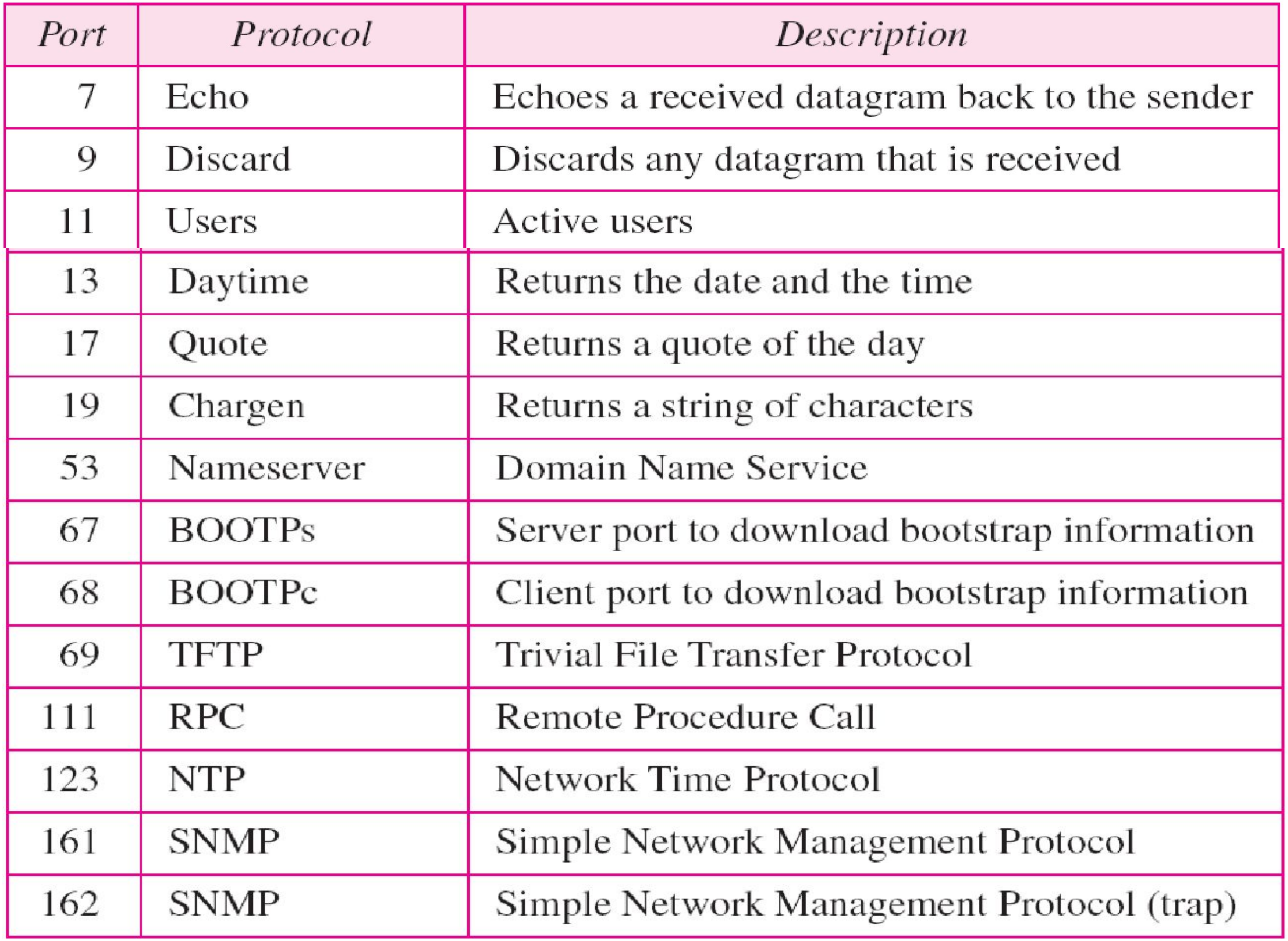
User Datagram Protocol (UDP)

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.

Topics discussed in this section:

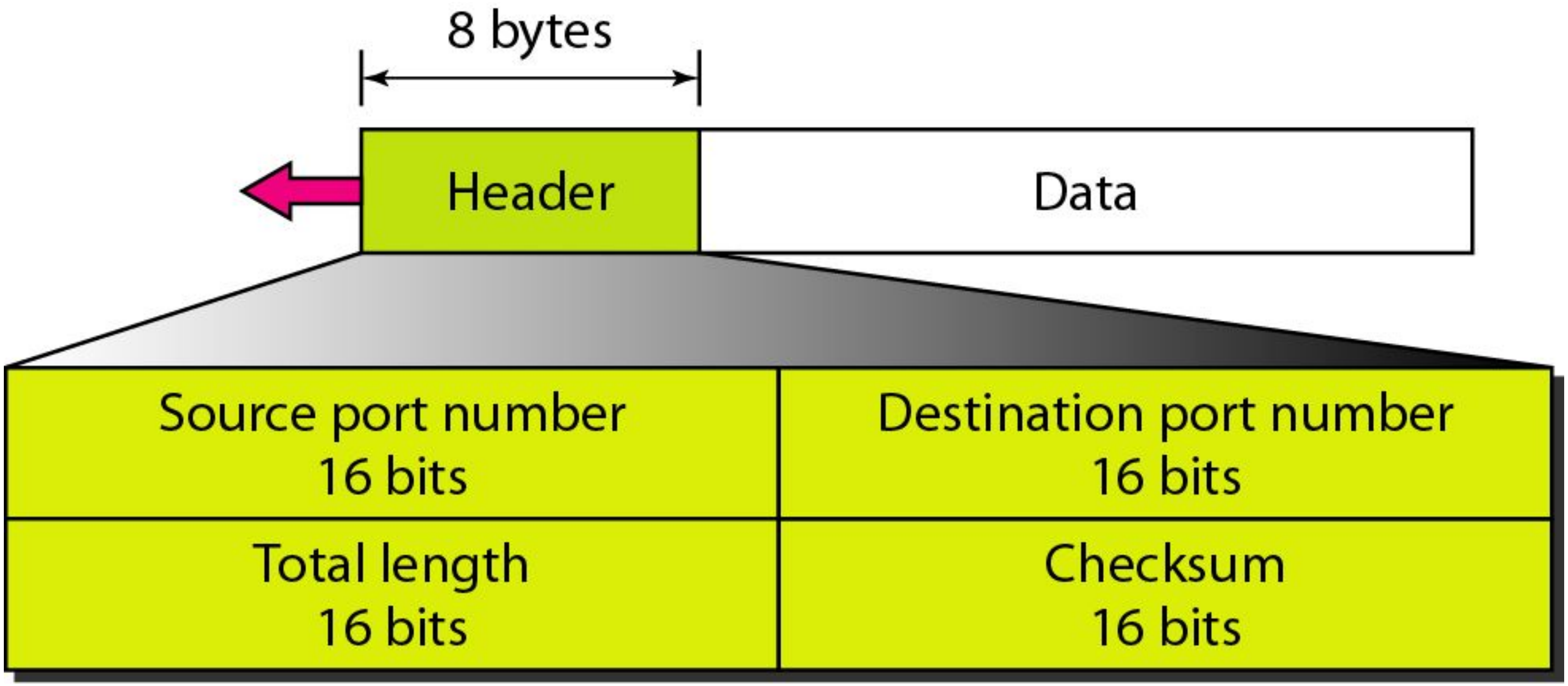
- Well-Known Ports for UDP
- User Datagram
- Checksum
- UDP Operation
- Use of UDP

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)

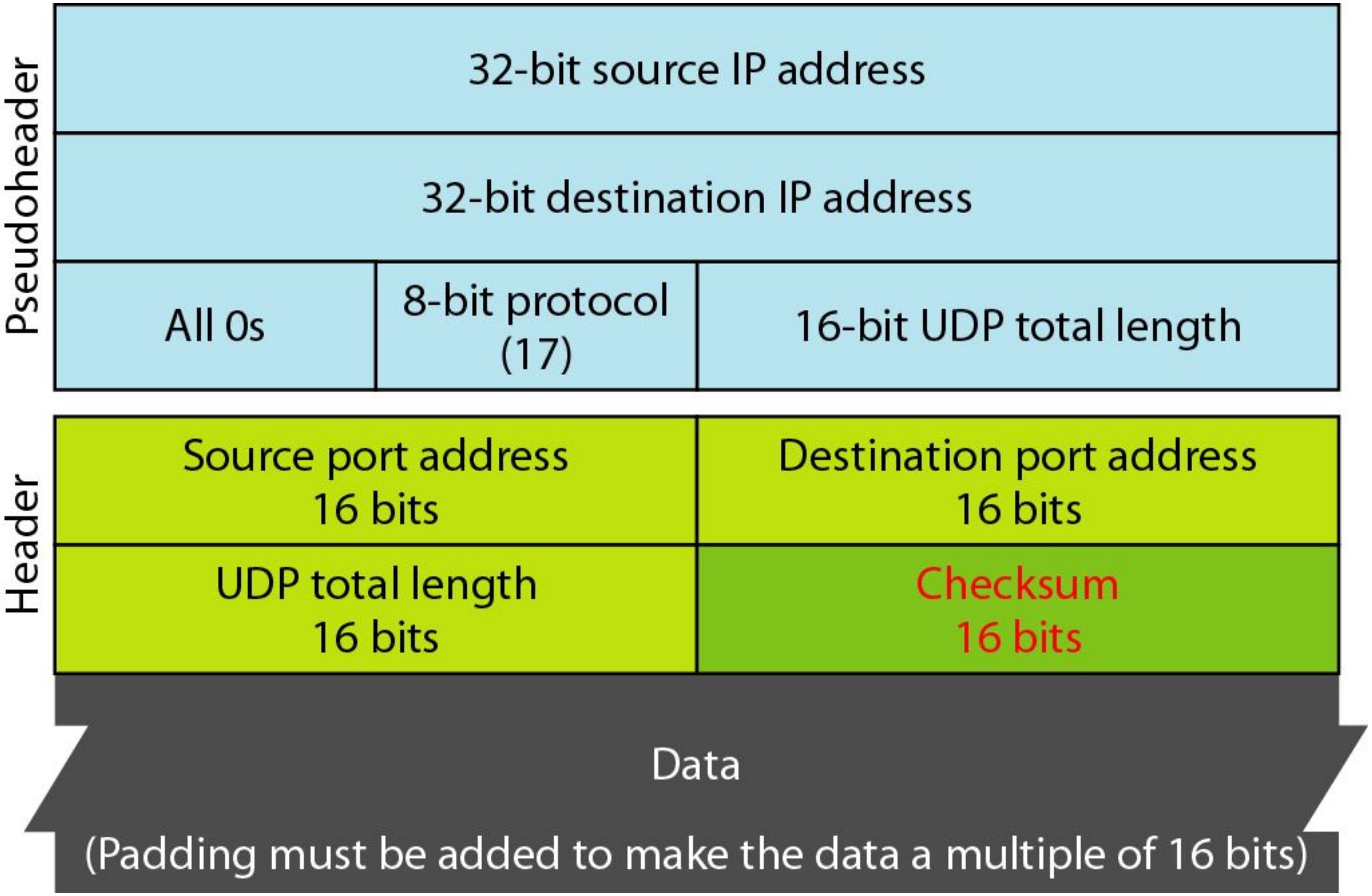
User datagram format



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

UDP length = IP length – IP header’s length

Pseudo header for checksum calculation



Pseudoheader

32-bit source IP address

32-bit destination IP address

All 0s

8-bit protocol (17)

16-bit UDP total length

Header

Source port address 16 bits

Destination port address 16 bits

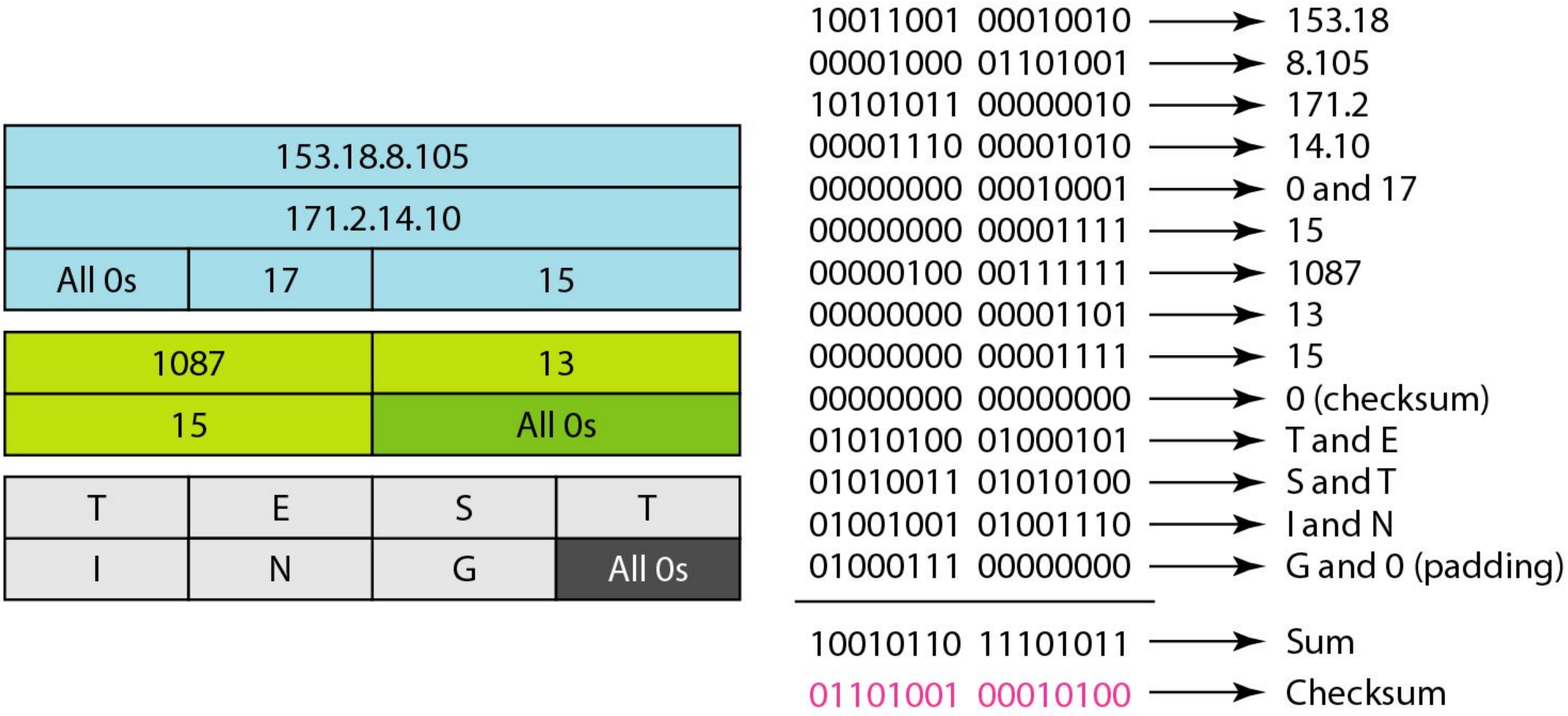
UDP total length 16 bits

Checksum 16 bits

Data

(Padding must be added to make the data a multiple of 16 bits)

Checksum calculation of a UDP



153.18.8.105

171.2.14.10

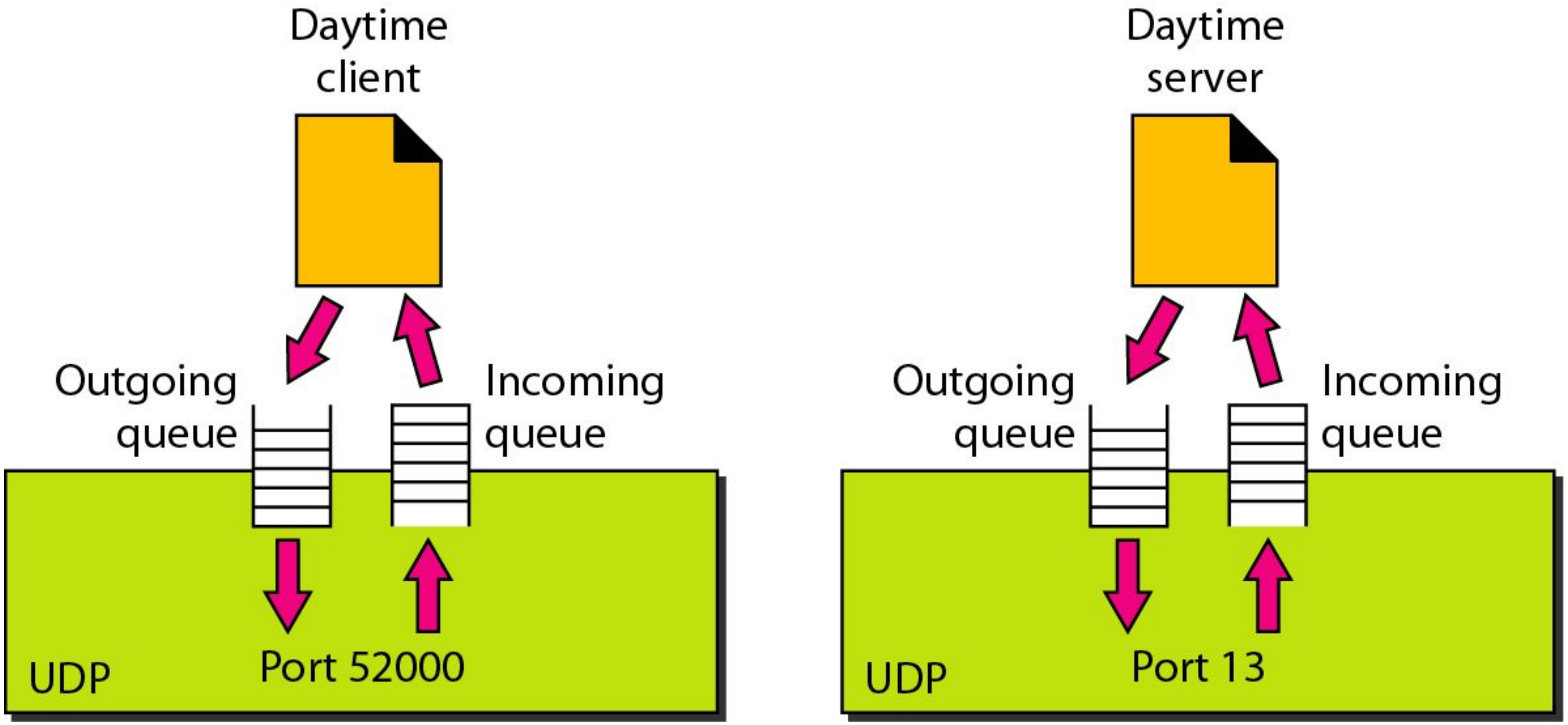
All 0s	17	15

1087	13
15	All 0s

T	E	S	T
I	N	G	All 0s

10011001 00010010	153.18
00001000 01101001	8.105
10101011 00000010	171.2
00001110 00001010	14.10
00000000 00010001	0 and 17
00000000 00001111	15
00000100 00111111	1087
00000000 00001101	13
00000000 00001111	15
00000000 00000000	0 (checksum)
01010100 01000101	T and E
01010011 01010100	S and T
01001001 01001110	I and N
01000111 00000000	G and 0 (padding)
10010110 11101011	Sum
01101001 00010100	Checksum

Queues in UDP



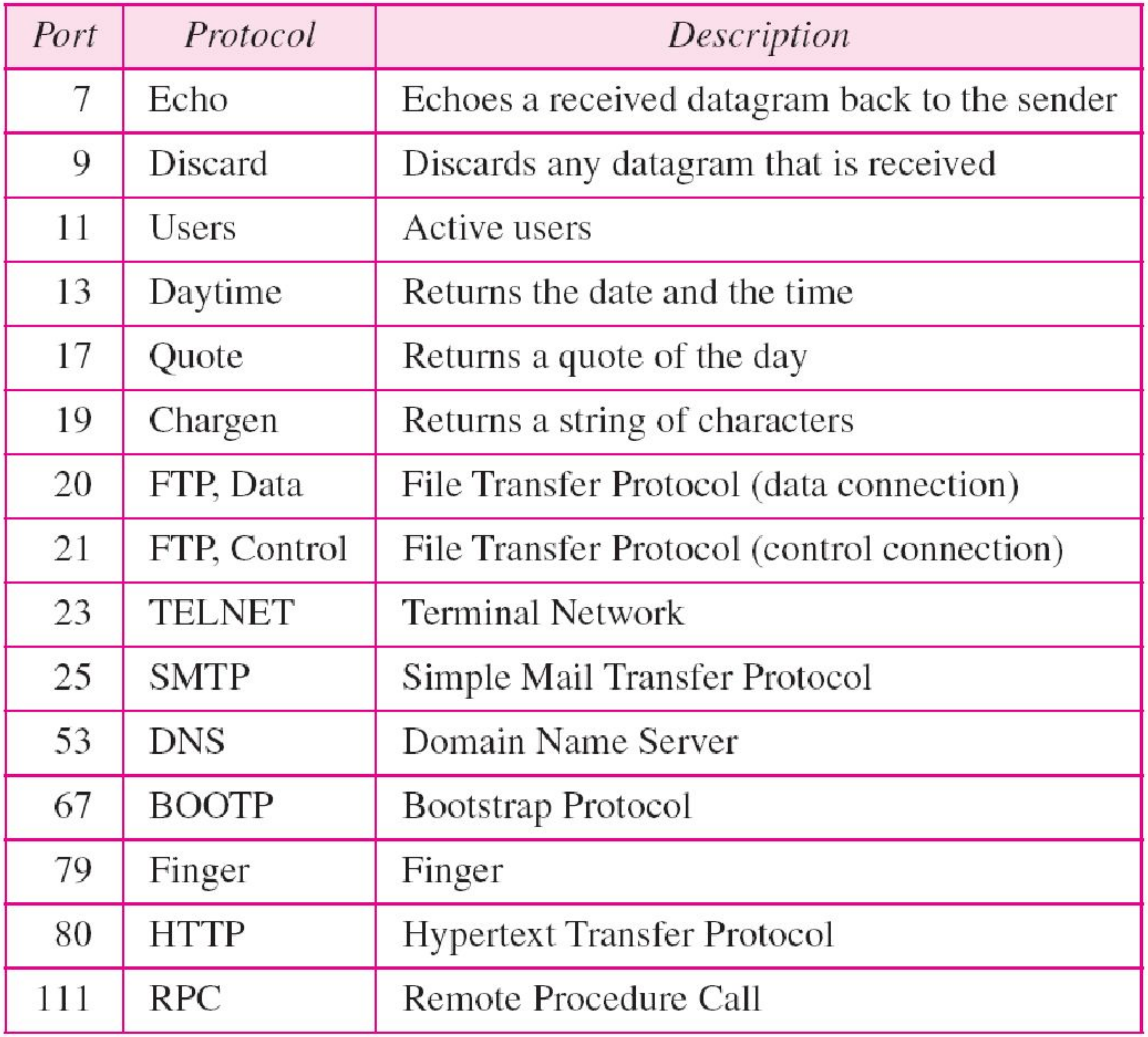
TCP

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.

Topics discussed in this section:

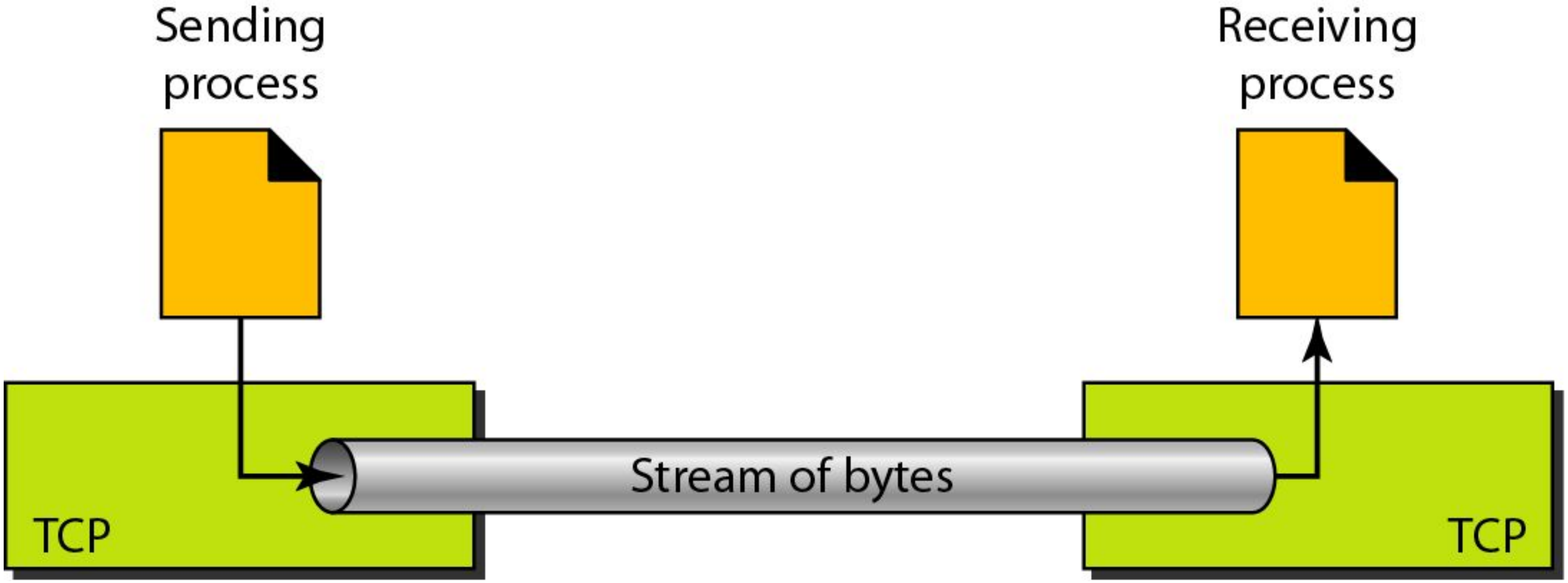
- TCP Services
- TCP Features
- Segment
- A TCP Connection
- Flow Control
- Error Control

Well-known ports used by TCP



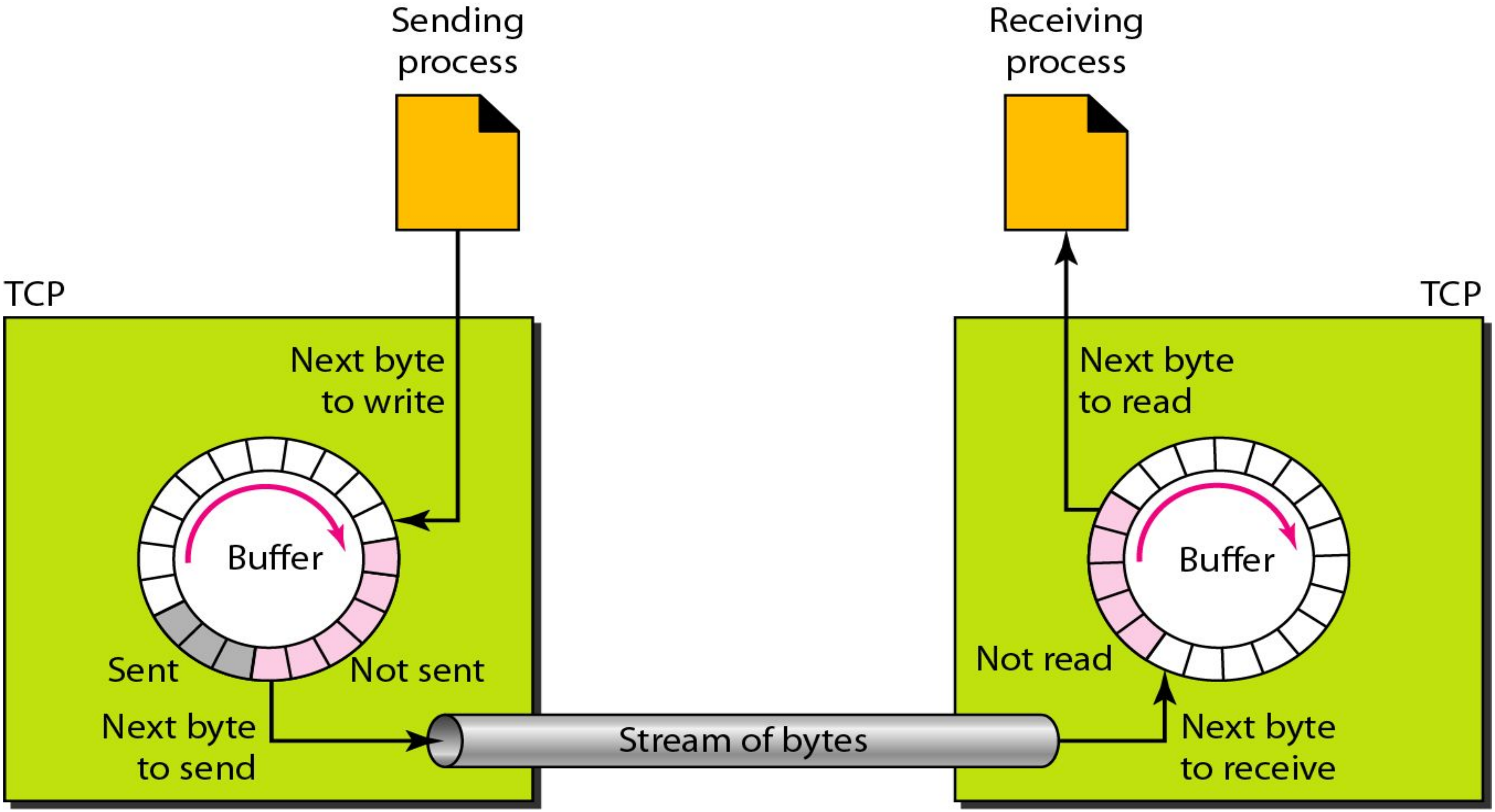
Port	Protocol	Description
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol (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

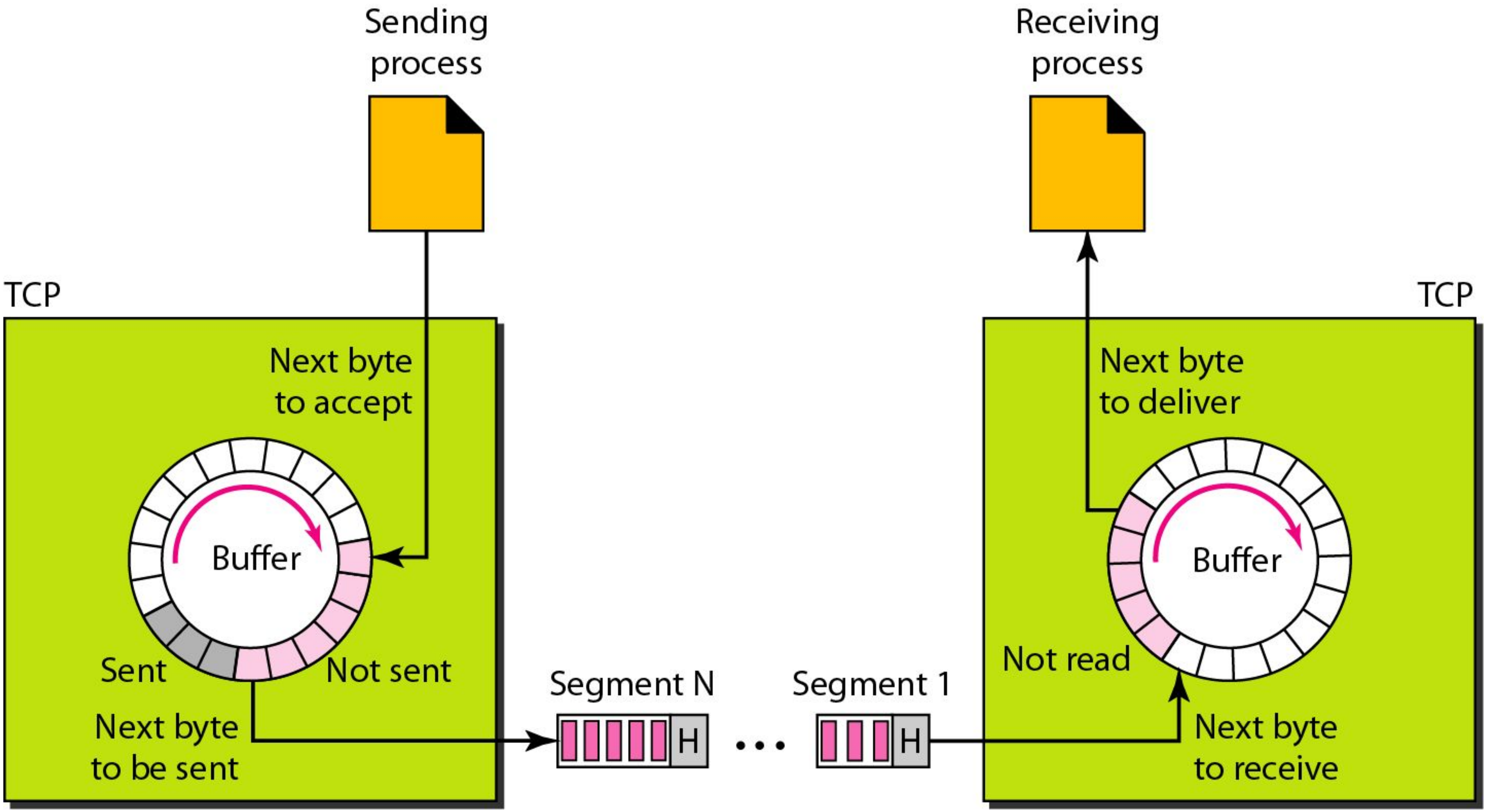


The bytes of data being transferred in each connection are numbered by TCP.
The numbering starts with a randomly generated number.

Sending and receiving buffers

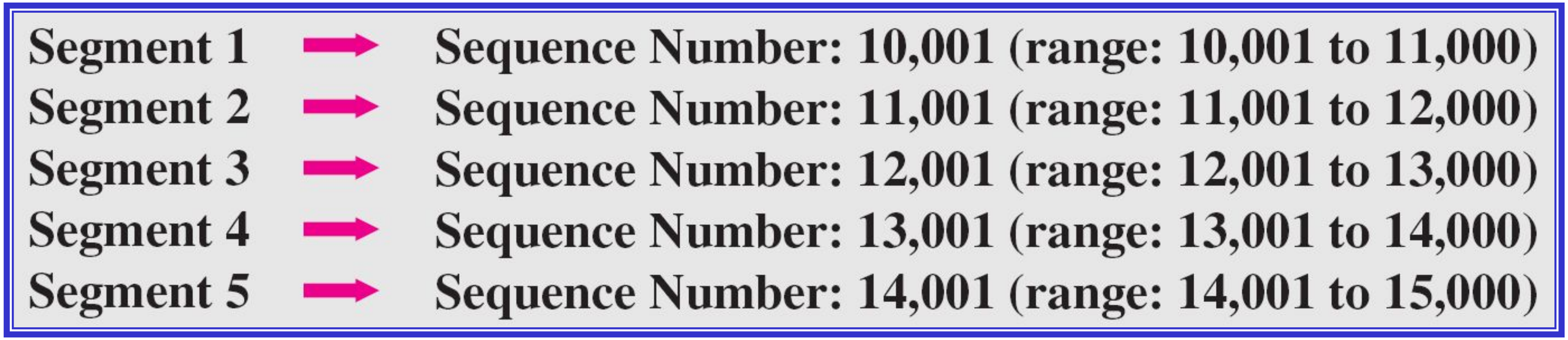


TCP segments



TCP segments and Sequence Number

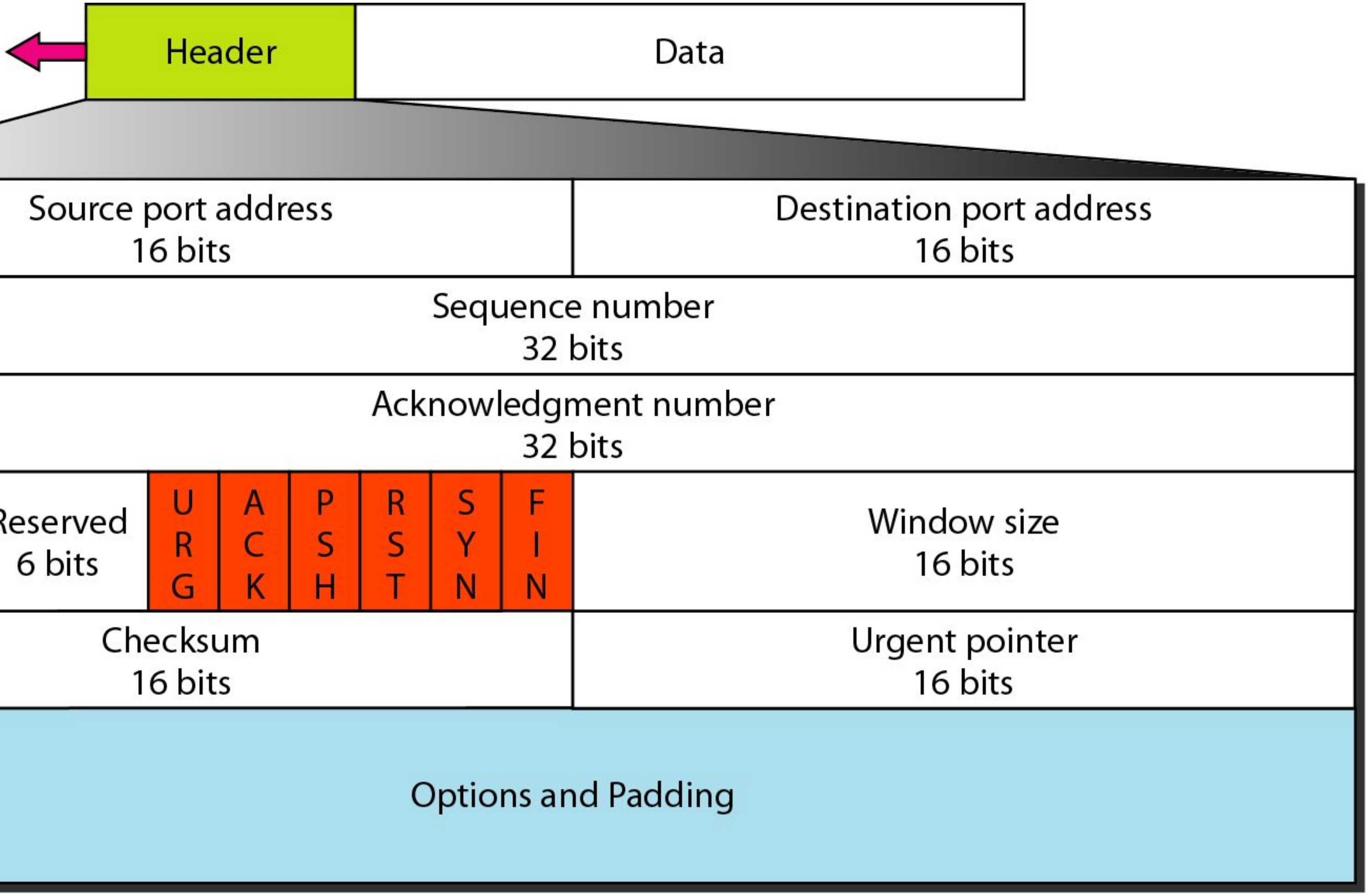
The following shows the sequence number for each segment:



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)

The value in the sequence number field of a segment defines the number of the first data byte contained in that segment.

TCP segment format



Header

Data

Source port address
16 bits

Destination port address
16 bits

Sequence number
32 bits

Acknowledgment number
32 bits

HLEN
4 bits

Reserved
6 bits

URG

ACK

PSH

RST

SYN

FIN

Window size
16 bits

Checksum
16 bits

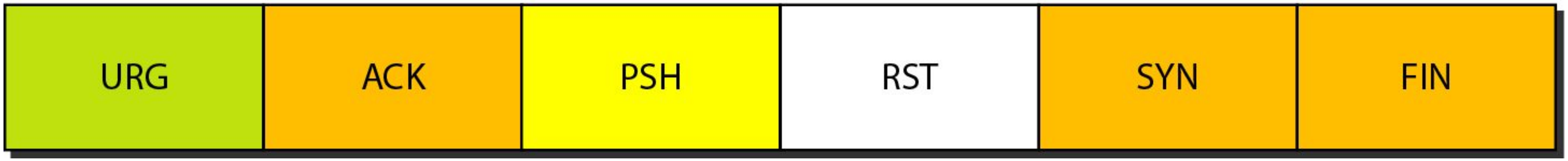
Urgent pointer
16 bits

Options and Padding

Control field

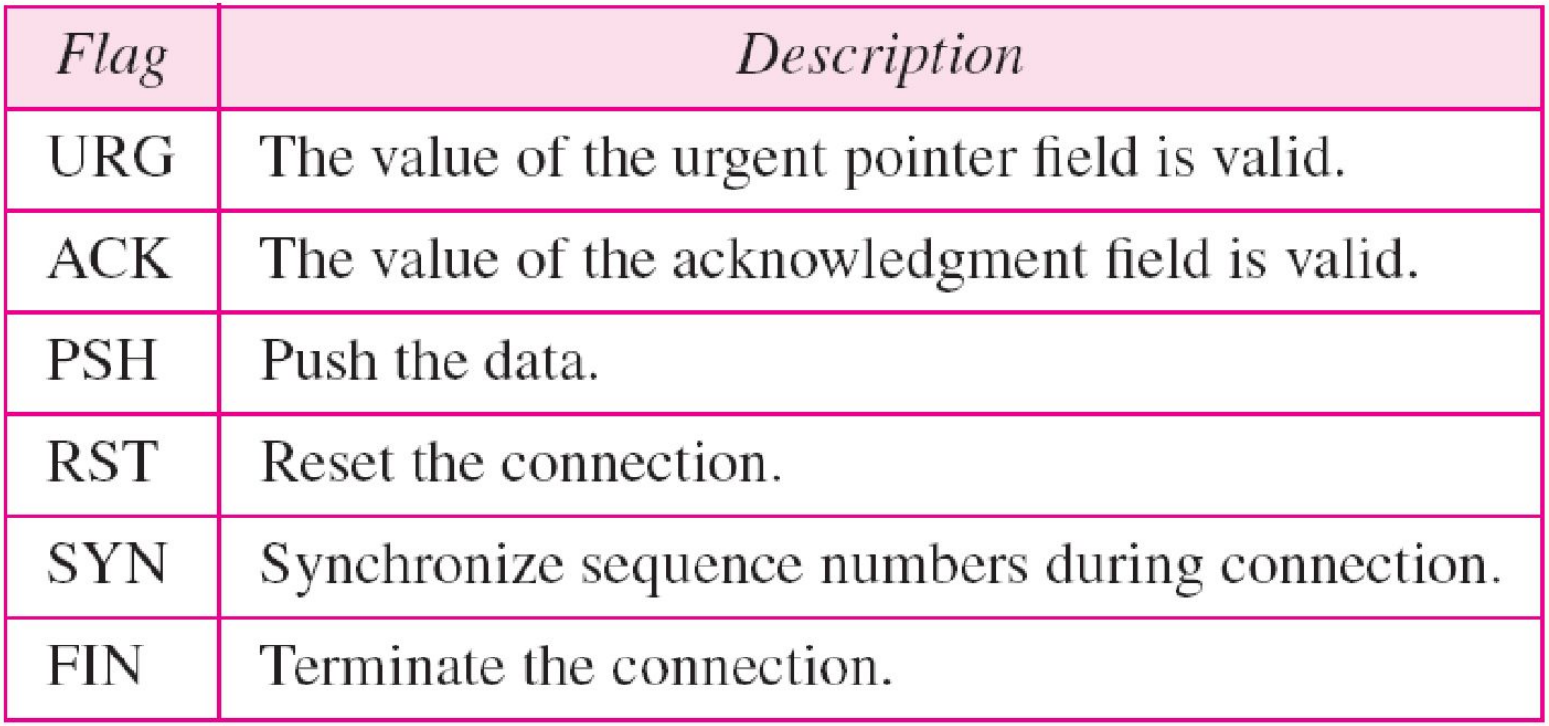
URG: Urgent pointer is valid
ACK: Acknowledgment is valid
PSH: Request for push

RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: Terminate the connection



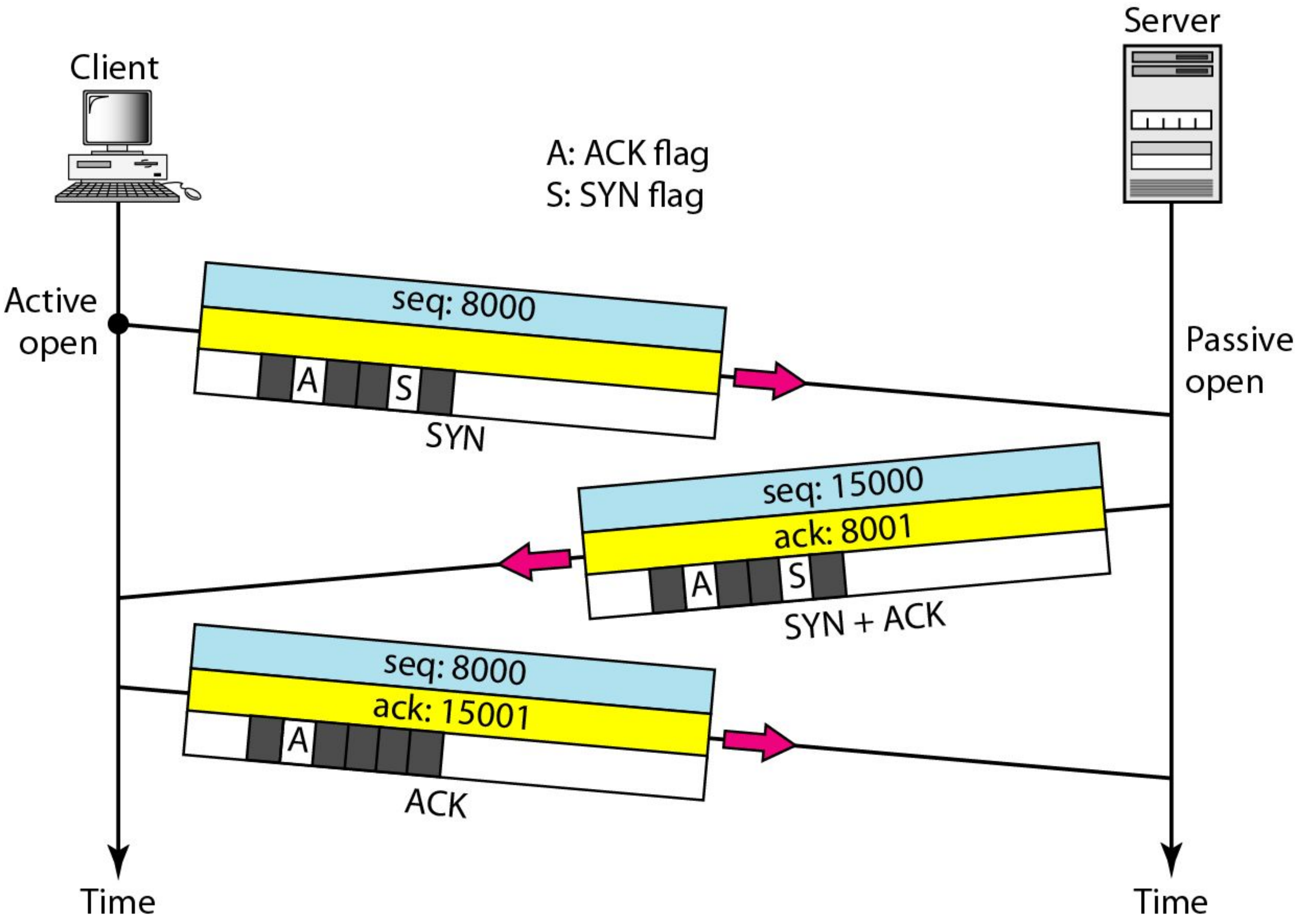
URG	ACK	PSH	RST	SYN	FIN

Description of flags in the control field



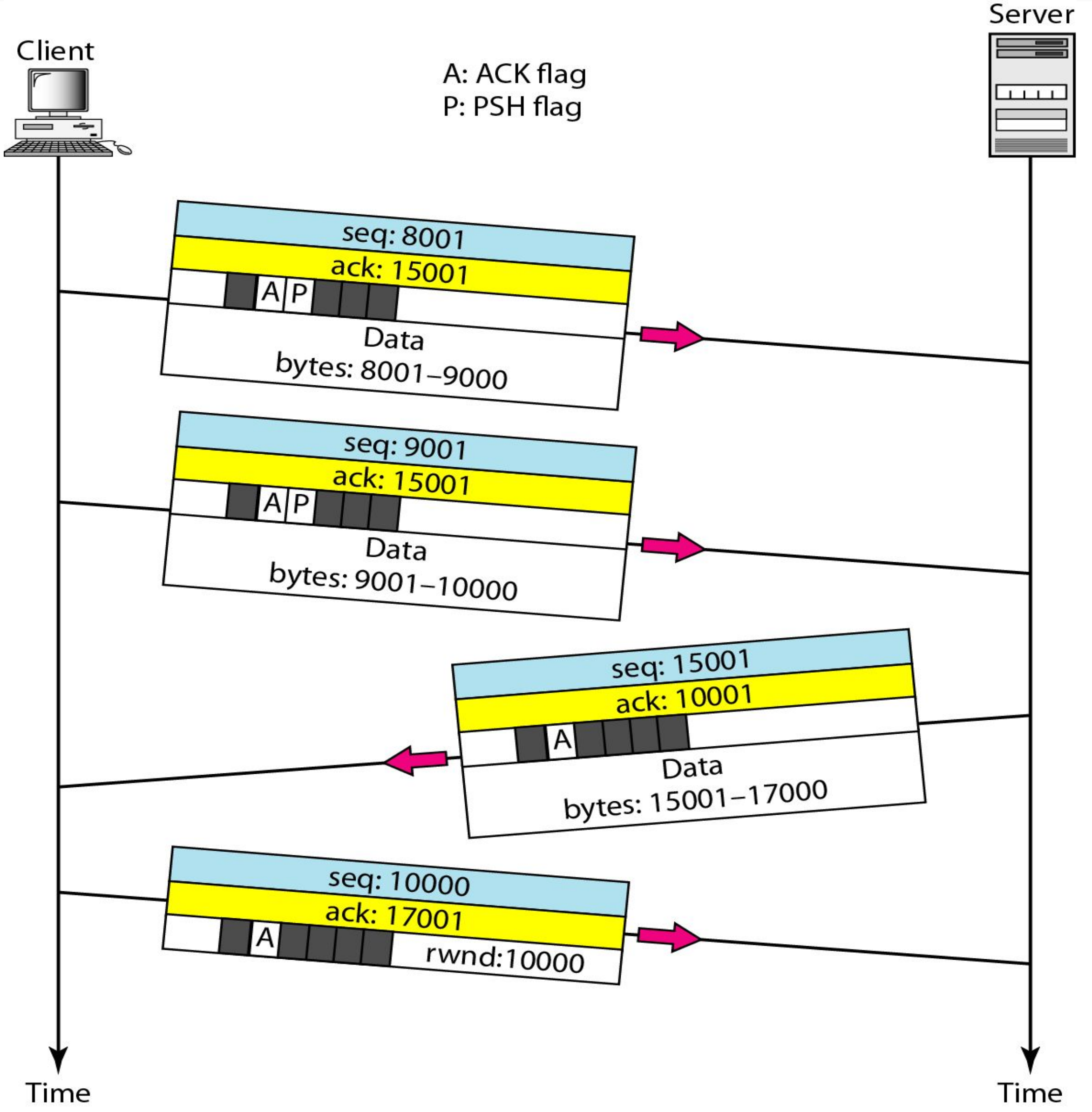
Flag	*Description*
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

Connection establishment using 3-way handshaking

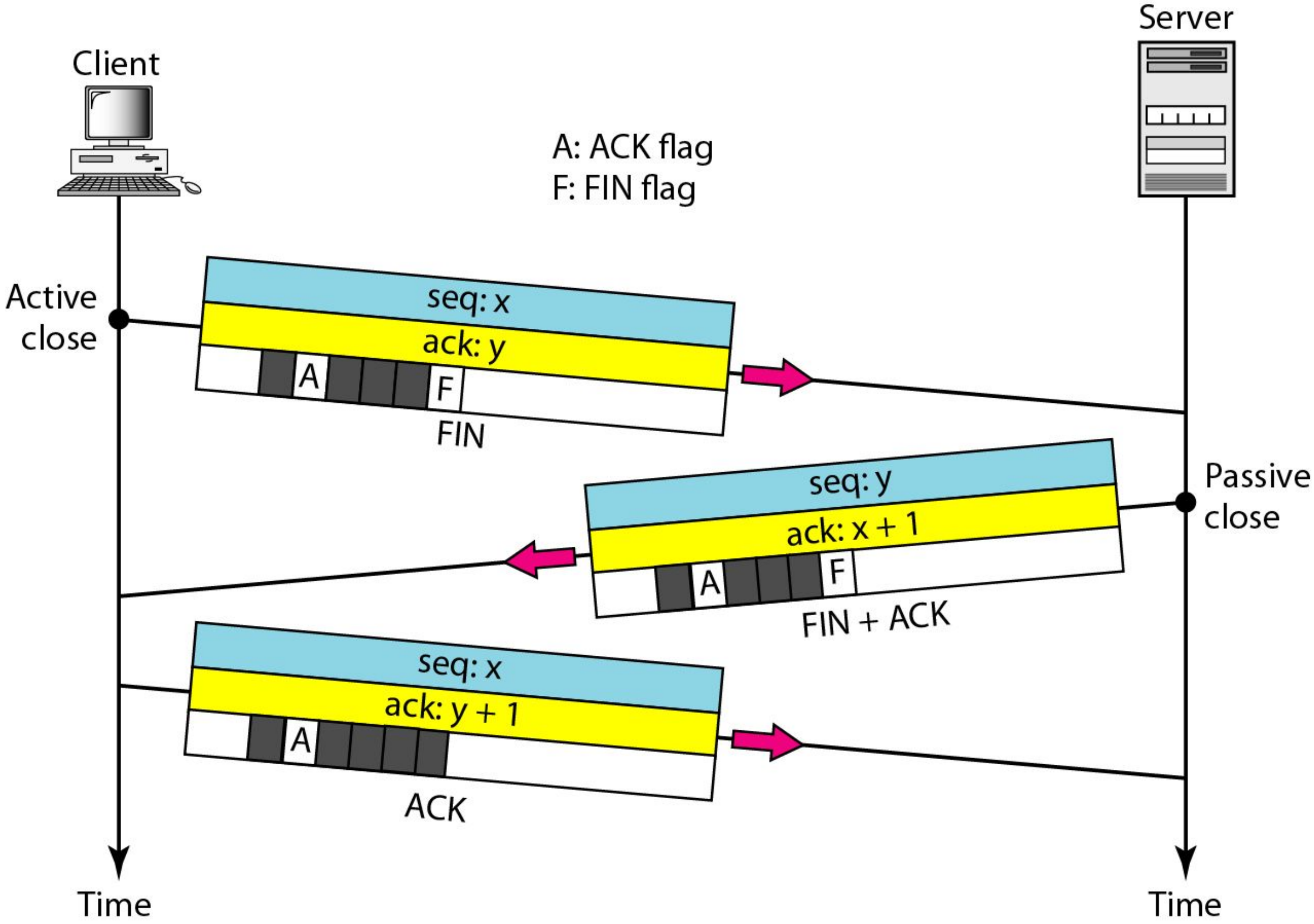


A SYN segment cannot carry data, but it consumes one sequence number.

Data transfer

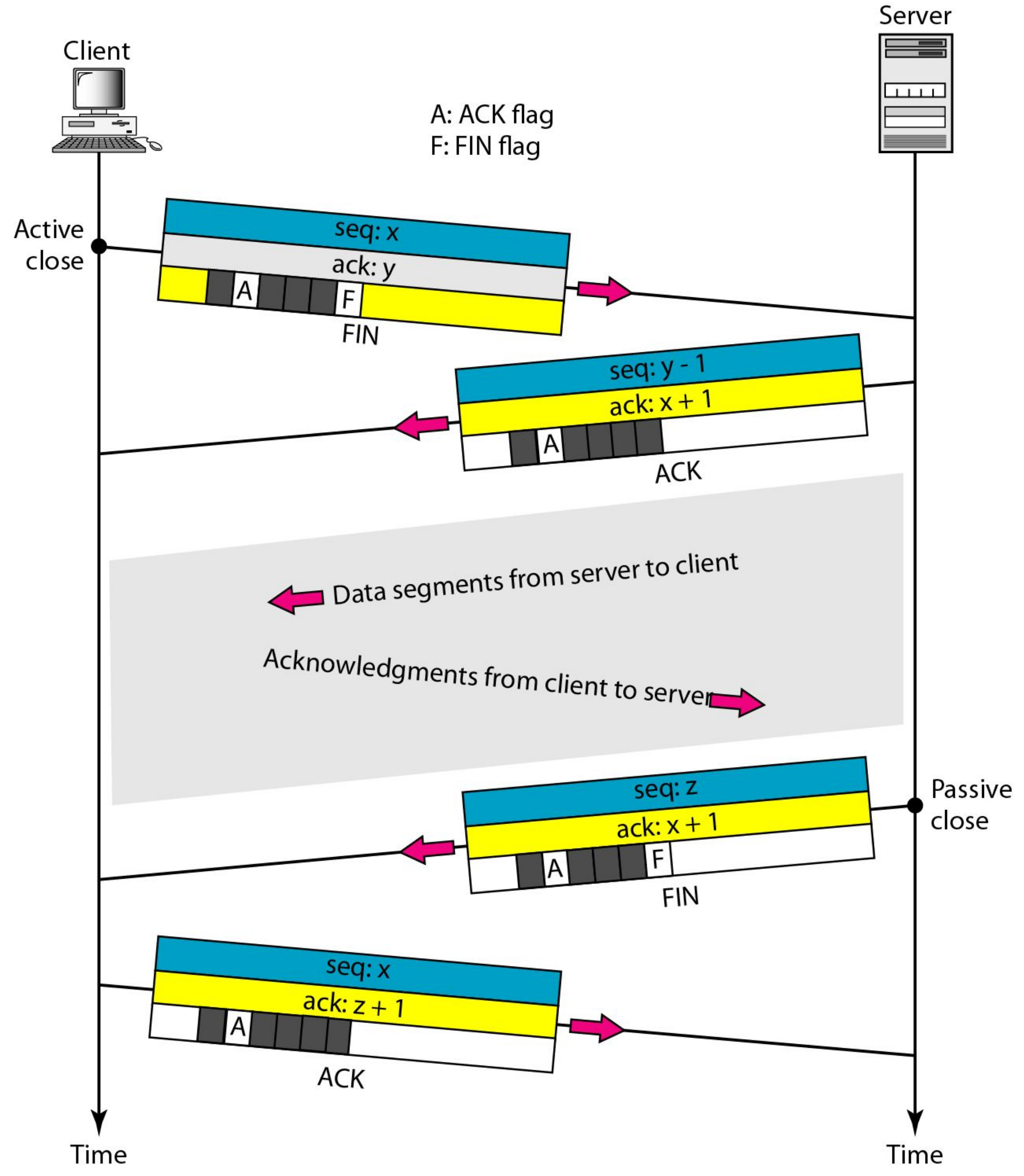


Connection termination using 3-way handshaking



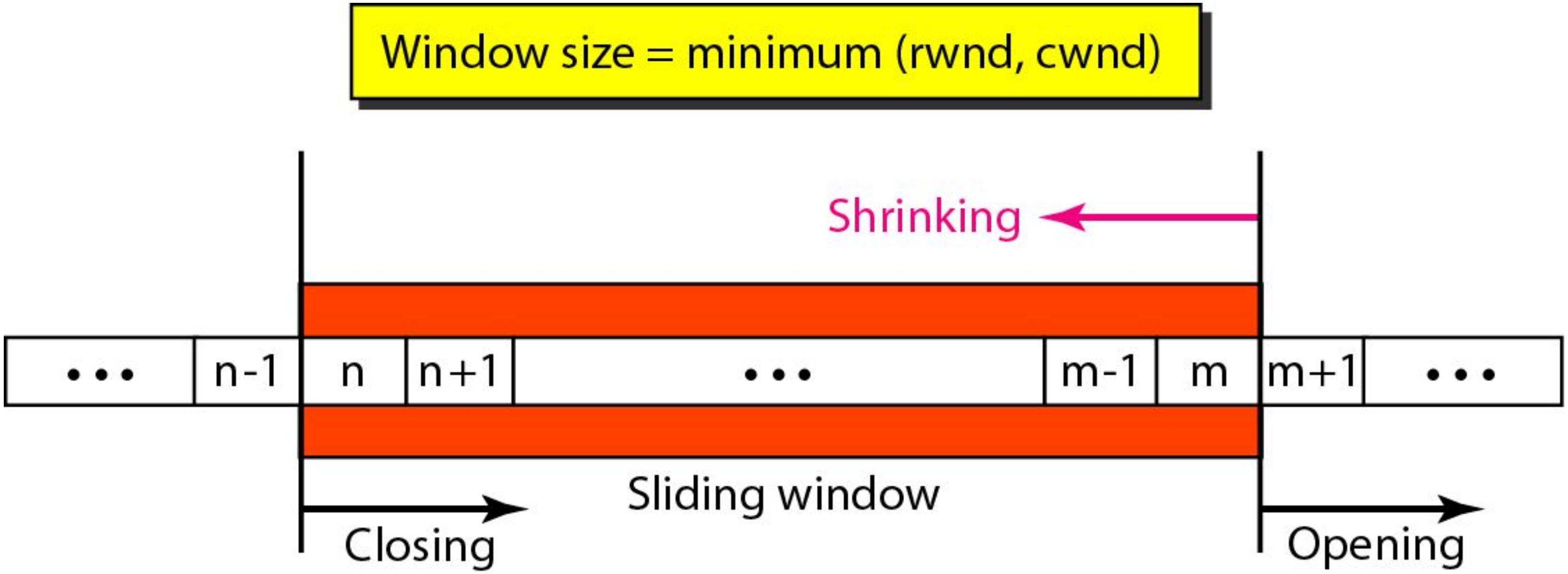
The FIN segment consumes one sequence number if it does not carry data.

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
Passive close
seq: z
ack: x + 1
A
F
FIN
seq: x
ack: z + 1
A
ACK
Time
Time

Sliding window



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.

Some points about TCP sliding windows:

- ❑ The size of the window is the lesser of rwnd and cwnd.
- ❑ The source does not have to send a full window’s worth of data
- ❑ The window can be opened or closed by the receiver, but should not be shrunk.
- ❑ The destination can send an acknowledgment at any time as long as it does not result in a shrinking window.
- ❑ The receiver can temporarily shut down the window; the sender, however, can always send a segment of 1 byte after the window is shut down.

Some points about TCP sliding windows(Contd..)

- ❑ ACK segments do not consume sequence numbers and are not acknowledged.
- ❑ In modern implementations, a retransmission occurs if the retransmission timer expires or three duplicate ACK segments have arrived.
- ❑ No retransmission timer is set for an ACK segment.
- ❑ Data may arrive out of order and be temporarily stored by the receiving TCP, but TCP guarantees that no out-of-order segment is delivered to the process.

Thank You