Transport Layer Protocols

Process-to-Process Delivery:
UDP, TCP

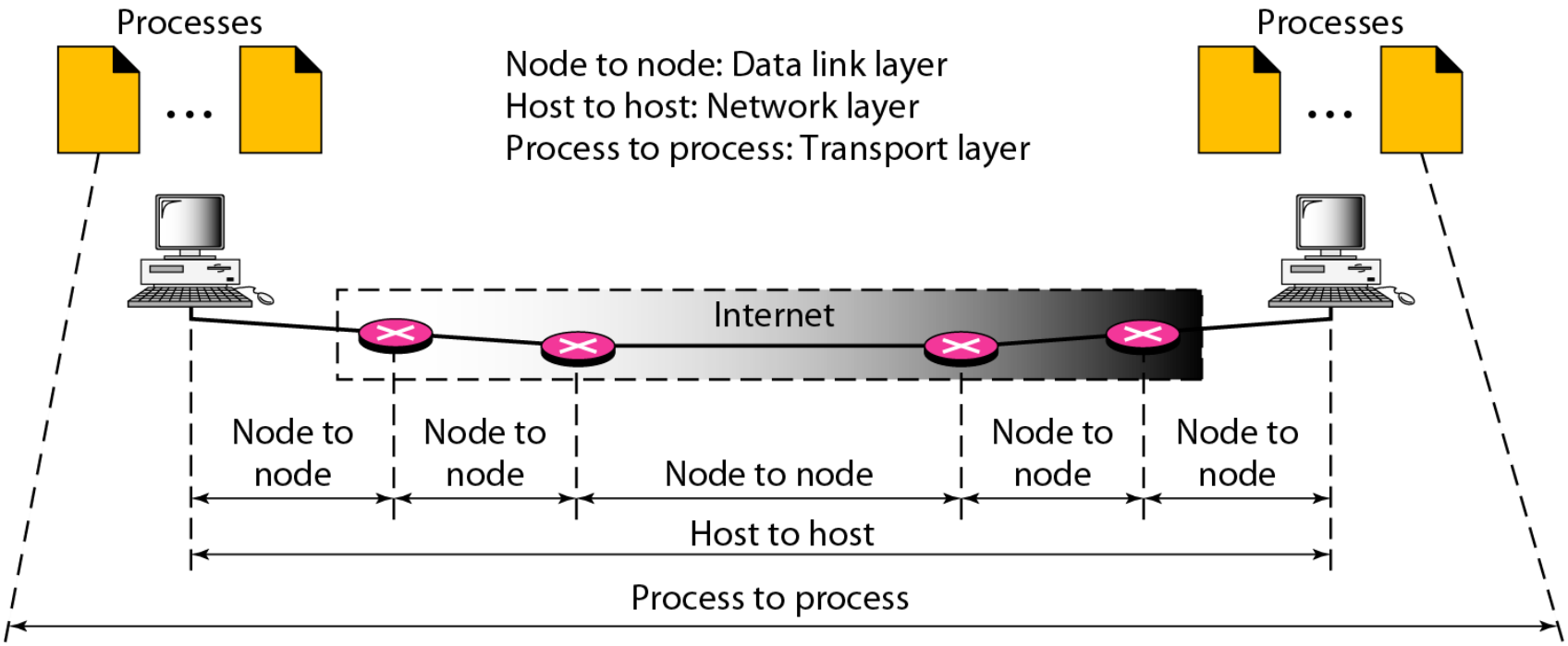
Process-to-Process Delivery

- ***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.***

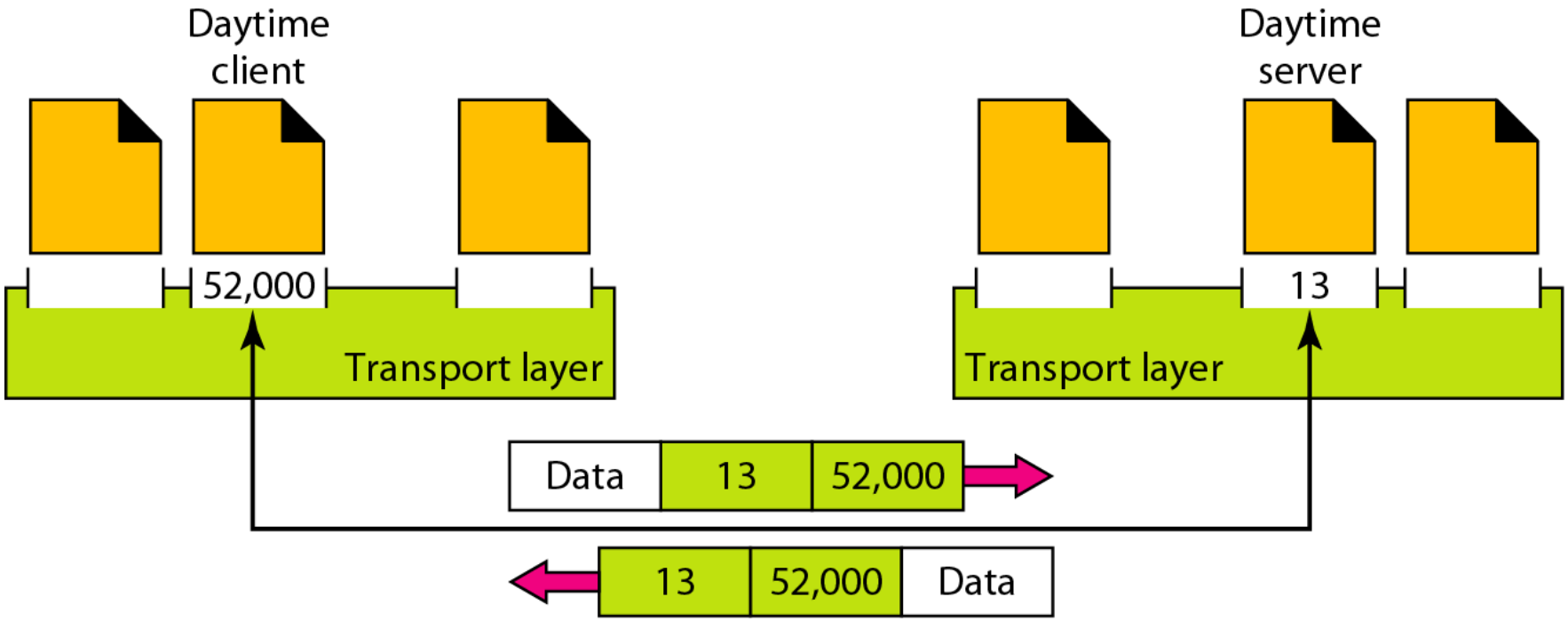
Note

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

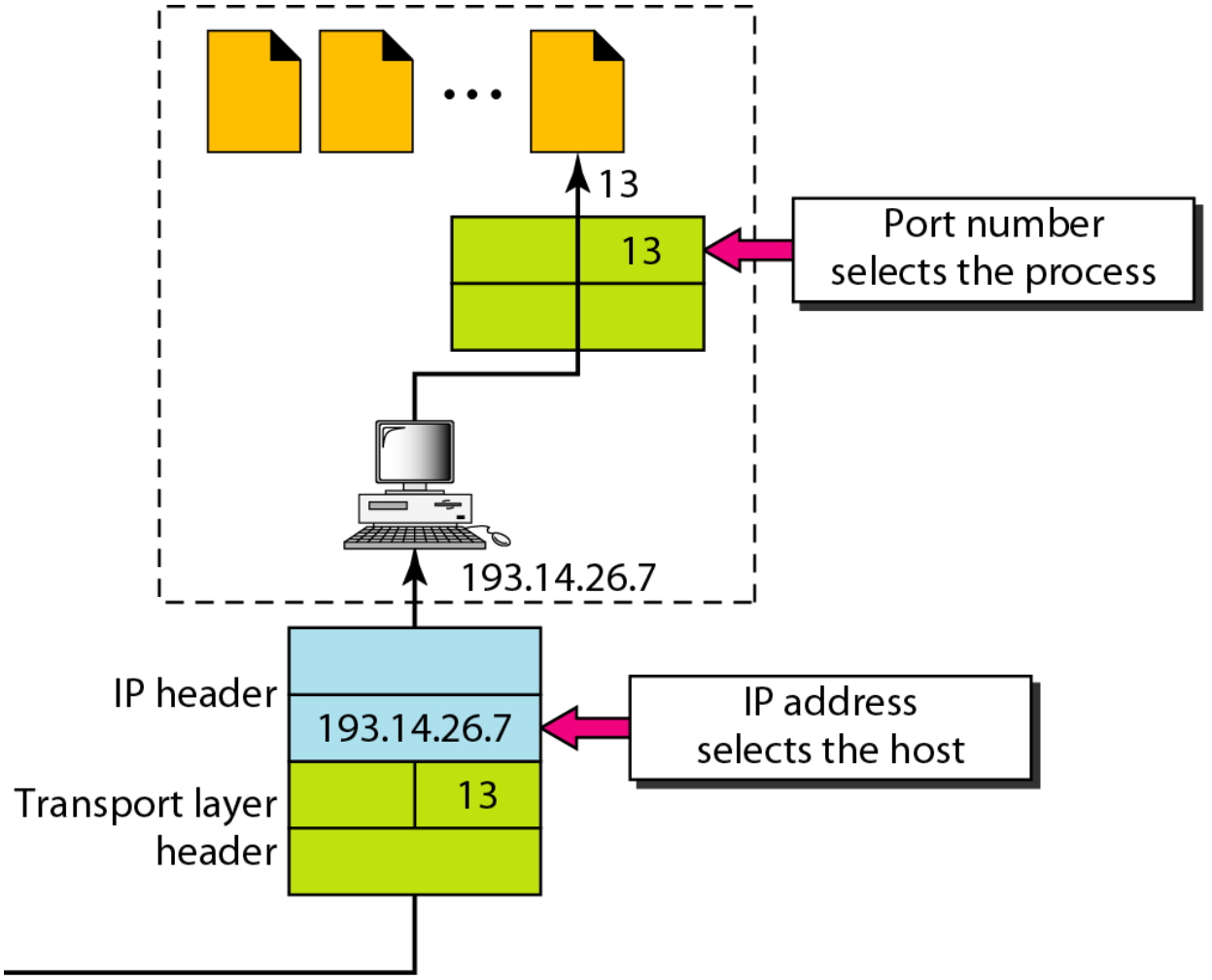
Types of data deliveries



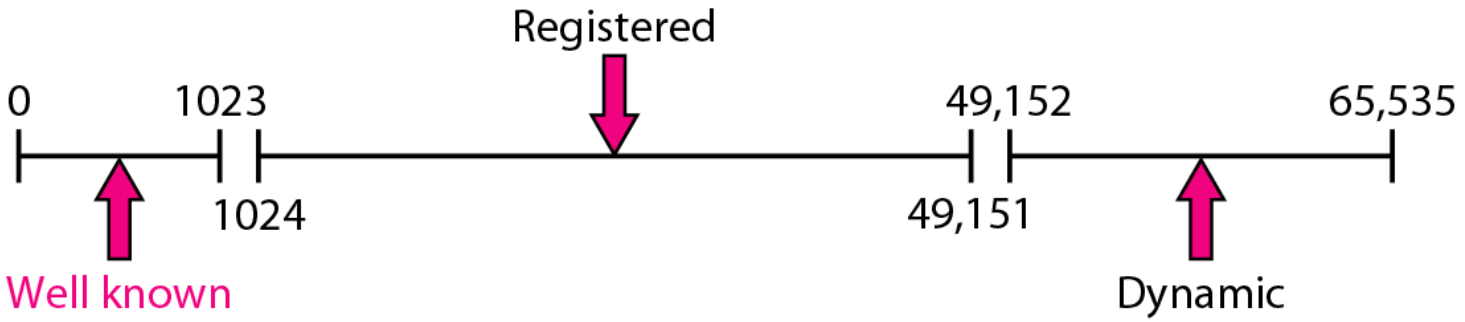
Port numbers



IP addresses versus port numbers



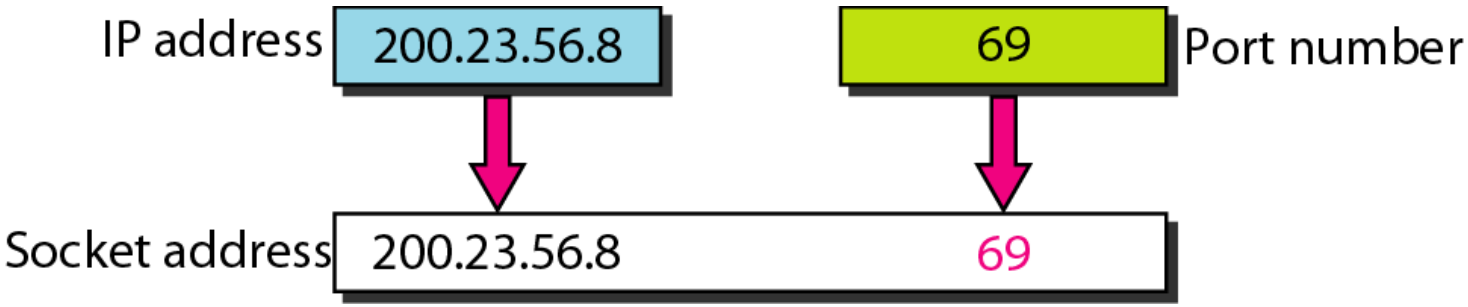
Internet Assigned Number Authority (IANA) Port Numbers



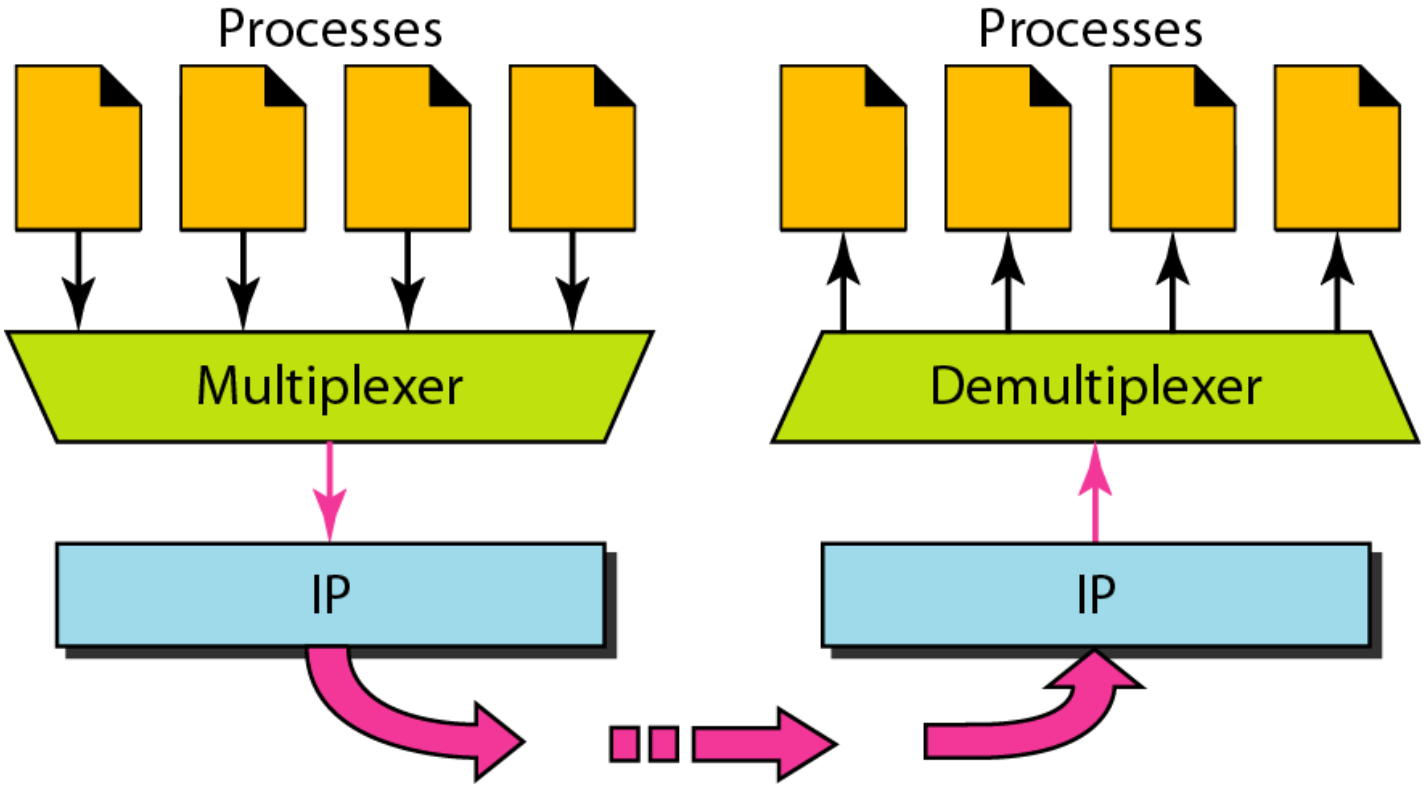
- **Well-known ports:**
  - The ports ranging from 0 to 1023 are assigned and controlled by lANA.
  - These are the well-known ports.
- **Registered ports:**
  - The ports ranging from 1024 to 49,151 are not assigned or controlled by lANA.
  - They can only be registered with lANA to prevent duplication.
- **Dynamic ports:**
  - The ports ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used by any process.
  - These are the ephemeral ports

Socket address

- Process-to-process delivery needs two identifiers, IP address and the port number, at each end to make a connection.
- The combination of an IP address and a port number is called a socket address.
- A transport layer protocol needs a pair of socket addresses:
  - *The client socket address and the server socket address.*



Multiplexing and demultiplexing



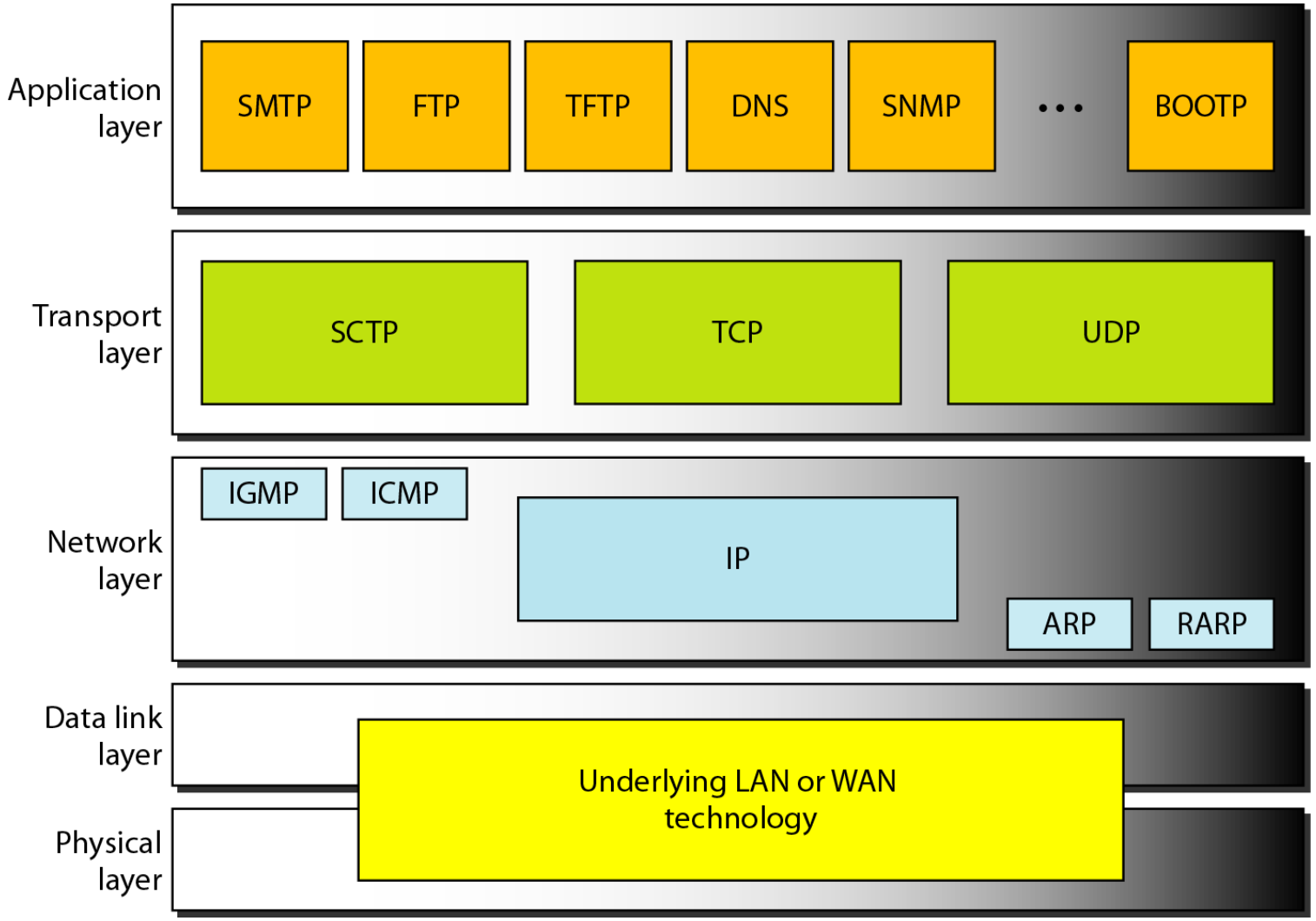
Connectionless Vs. Connection Oriented Service

- A transport layer protocol can either be connectionless or connection-oriented.
- Connectionless Service:
  - In a connectionless service, the packets are sent from one party to another with no need for connection establishment or connection release.
  - **Example:** UDP is connectionless.

Connectionless Vs. Connection Oriented Service

- Connection Oriented *Service*
  - A connection is first established between the sender and the receiver.
  - Data are transferred.
  - At the end, the connection is released.
  - **Example:** TCP and SCTP are connection-oriented protocols.

Position of UDP, TCP, and SCTP 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.*

- *There is no flow control and hence no window mechanism. The receiver may overflow with incoming messages.*

- *There is no error control mechanism in UDP except for the checksum.*

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)

Example

- ***In UNIX, the well-known ports are stored in a file called /etc/services. Each line in this file gives the name of the server and the well-known port number.***
- ***We can use the grep utility to extract the line corresponding to the desired application.***
- ***The following shows the port for FTP.***
- ***Note that FTP can use port 21 with either UDP or TCP.***

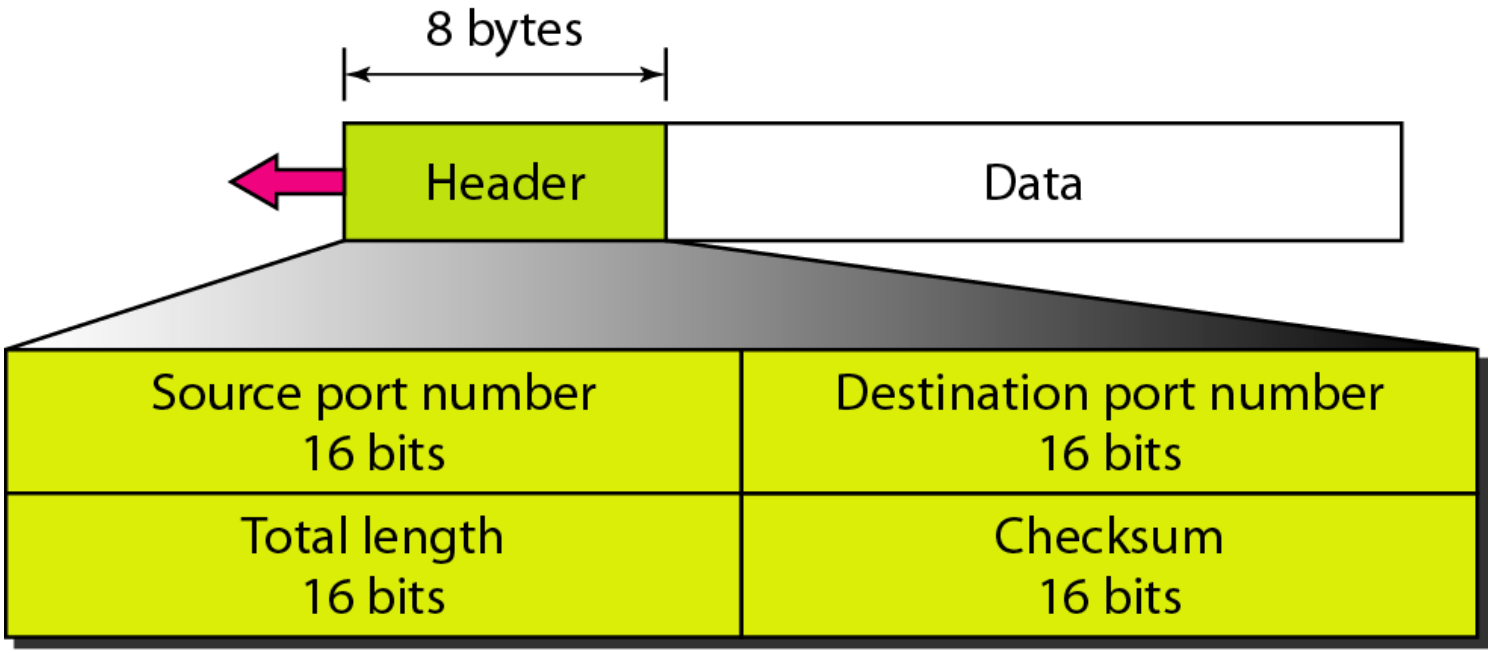
```
$ grep    ftp   /etc/services
ftp          21/tcp
ftp          21/udp
```

Example (continued)

SNMP uses two port numbers (161 and 162), each for a different purpose.

```
$ grep      snmp /etc/services
snmp            161/tcp         #Simple Net  Mgmt Proto
snmp            161/udp         #Simple Net  Mgmt Proto
snmptrap        162/udp         #Traps for SNMP
```

User datagram format



User datagram format

- **Source port number:**
  - This is the port number used by the process running on the source host.
  - It is 16 bits long → port number can range from 0 to 65,535.
  - If the source host **is the client**
    - Port number is **an ephemeral port number**
  - If the source host is **a server**
    - Port number is the **well known port number**

User datagram format

- **Destination port number.**
  - This is the port number used by the process running on the destination host.
  - If the source host is the client
    - Port number is an ephemeral port number
  - If the source host is a server
    - Port number is the well known port number
- **Length:**
  - This is a 16-bit field that defines the total length of the user datagram(header plus data)
  - The total length needs to be much less because a UDP user datagram is stored in an IP datagram with a total length of 65,535 bytes.

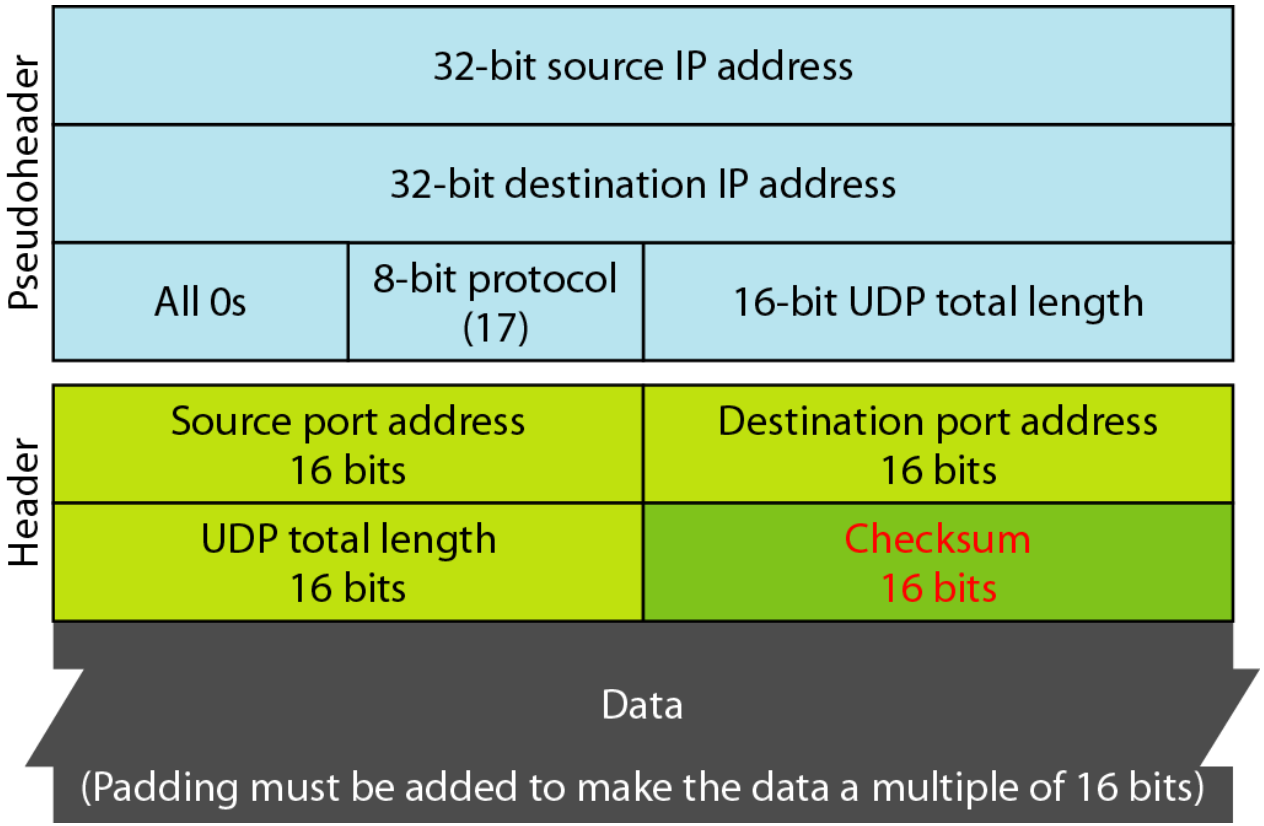
Note

UDP length
= IP length – IP header's length

Checksum

- The UDP checksum calculation is different from the one for IP and ICMP.
- Here the checksum includes three sections:
  - A pseudoheader,
  - the UDP header,
  - and the data coming from the application layer.
- The pseudoheader is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with 0s

Pseudoheader for checksum calculation

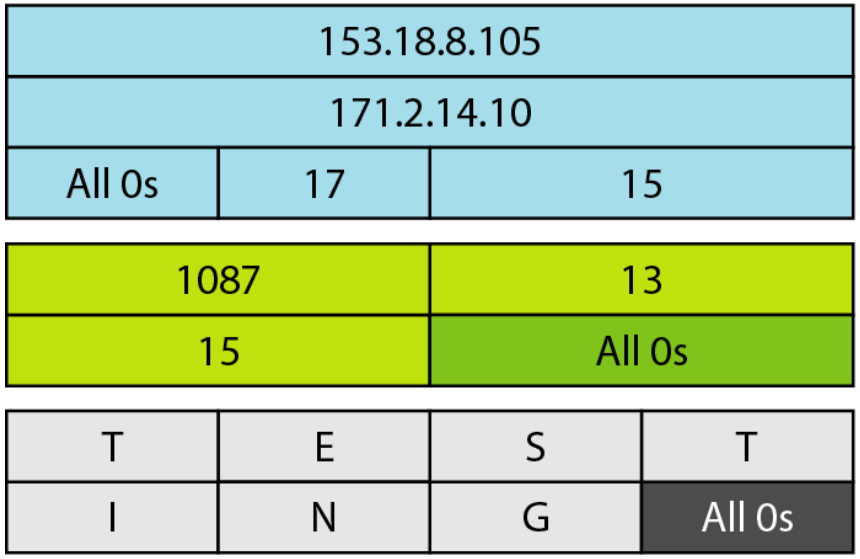


Checksum

- Optional Use of the Checksum
  - The calculation of the checksum and its inclusion in a user datagram are optional.
  - If the checksum is not calculated, the field is filled with 1s.
  - Note that a calculated checksum can never be all 1s because this implies that the sum is all 0s.
    - which is impossible because it requires that the value of fields to be 0s.

Checksum calculation of a simple UDP user datagram

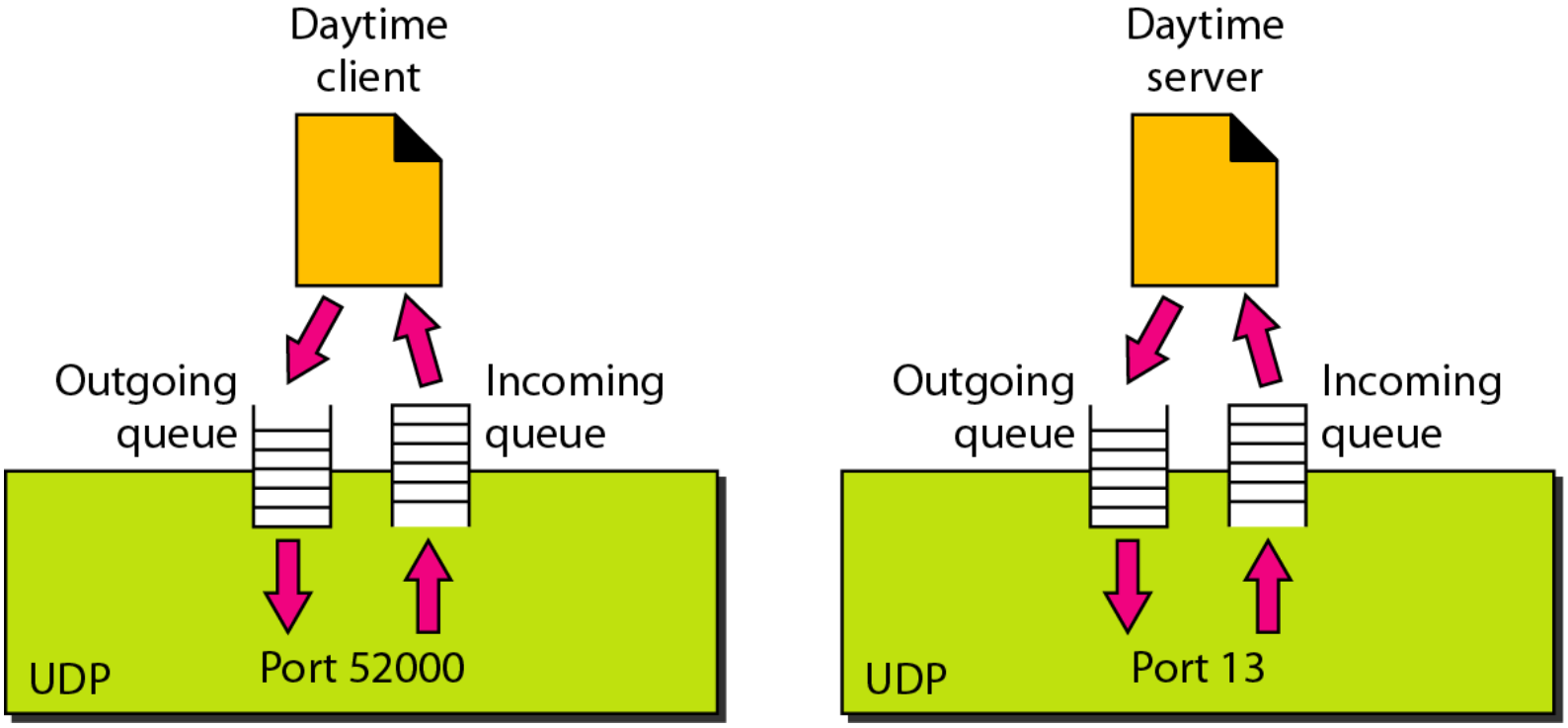
- Figure shows the checksum calculation for a very small user datagram with only 7 bytes of data.
- Because the number of bytes of data is odd, padding is added for checksum calculation.
- The pseudoheader as well as the padding will be dropped when the user datagram is delivered to IP.



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



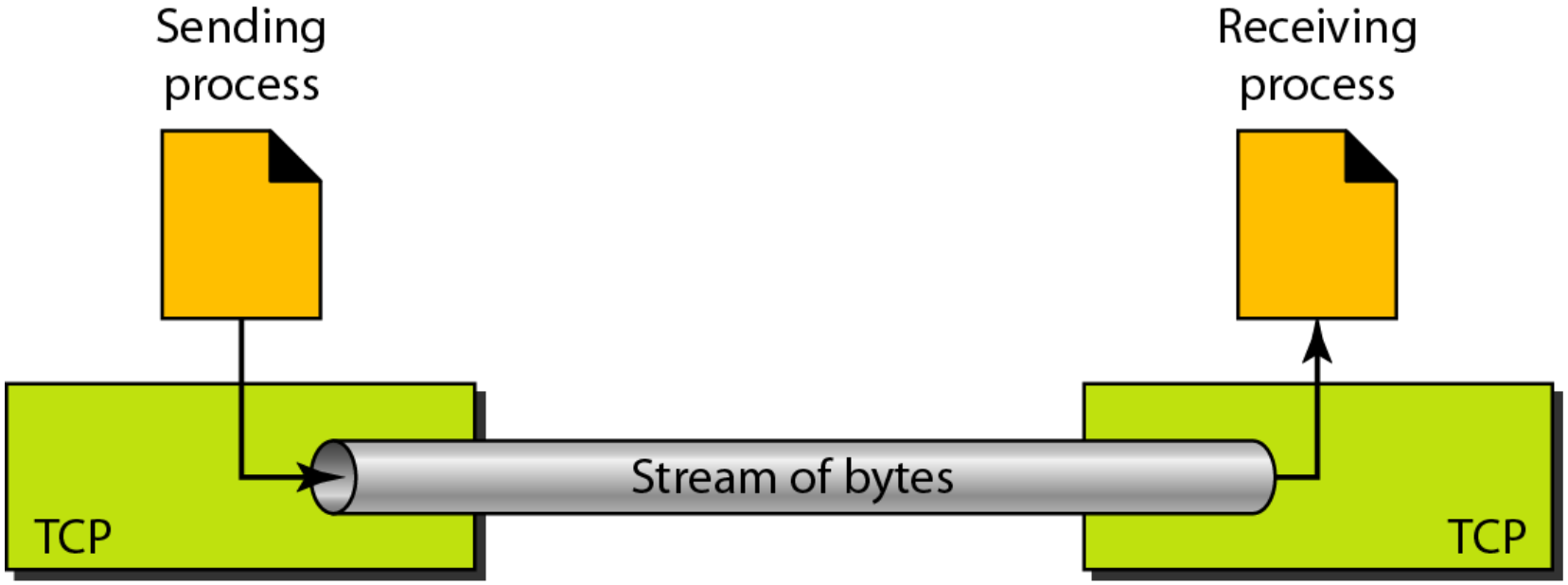
Transmission Control Protocol (TCP)

- ***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.***

Well-known ports used by TCP

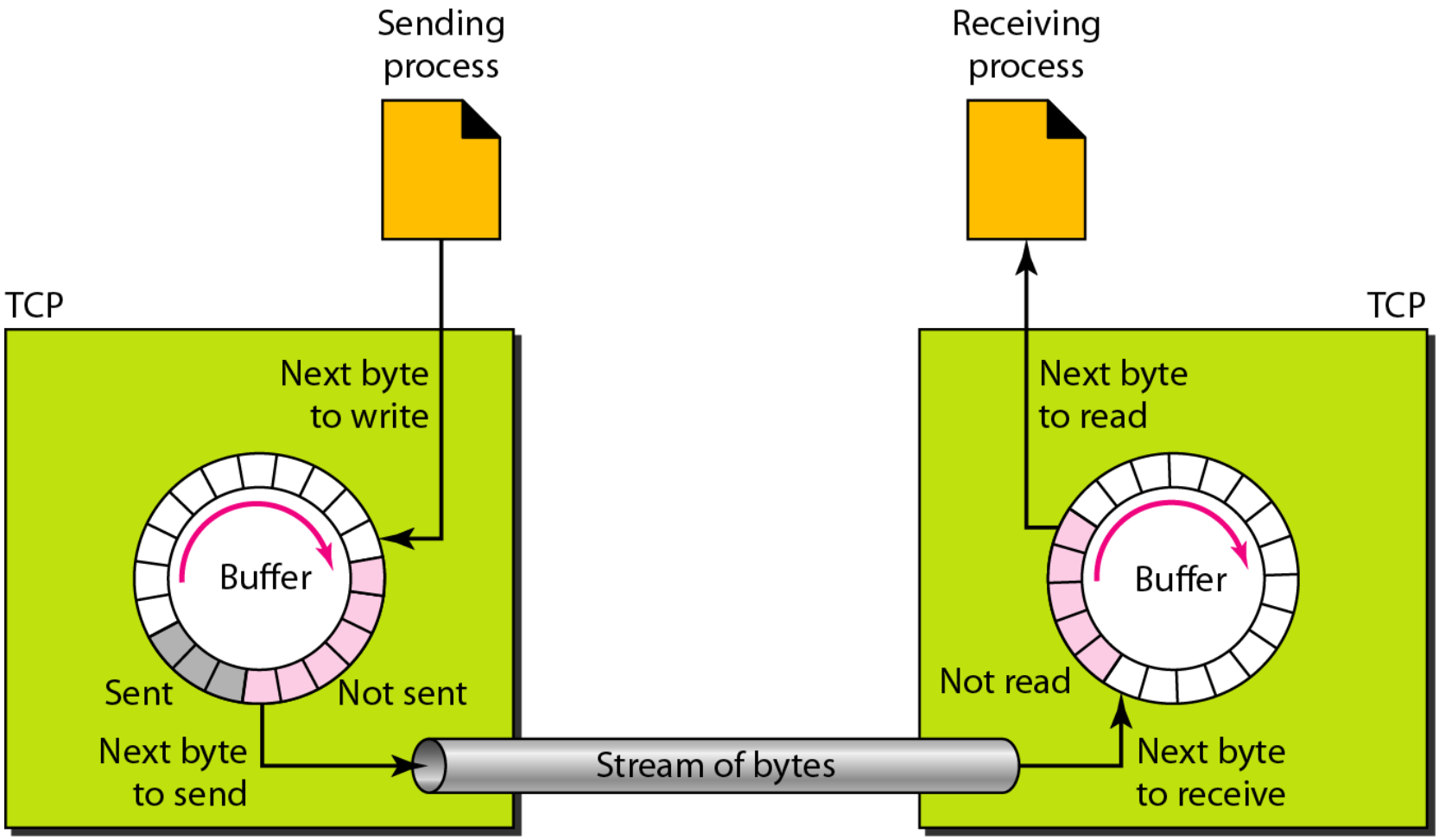
Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
20	FTP, Data	File Transfer Protocol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

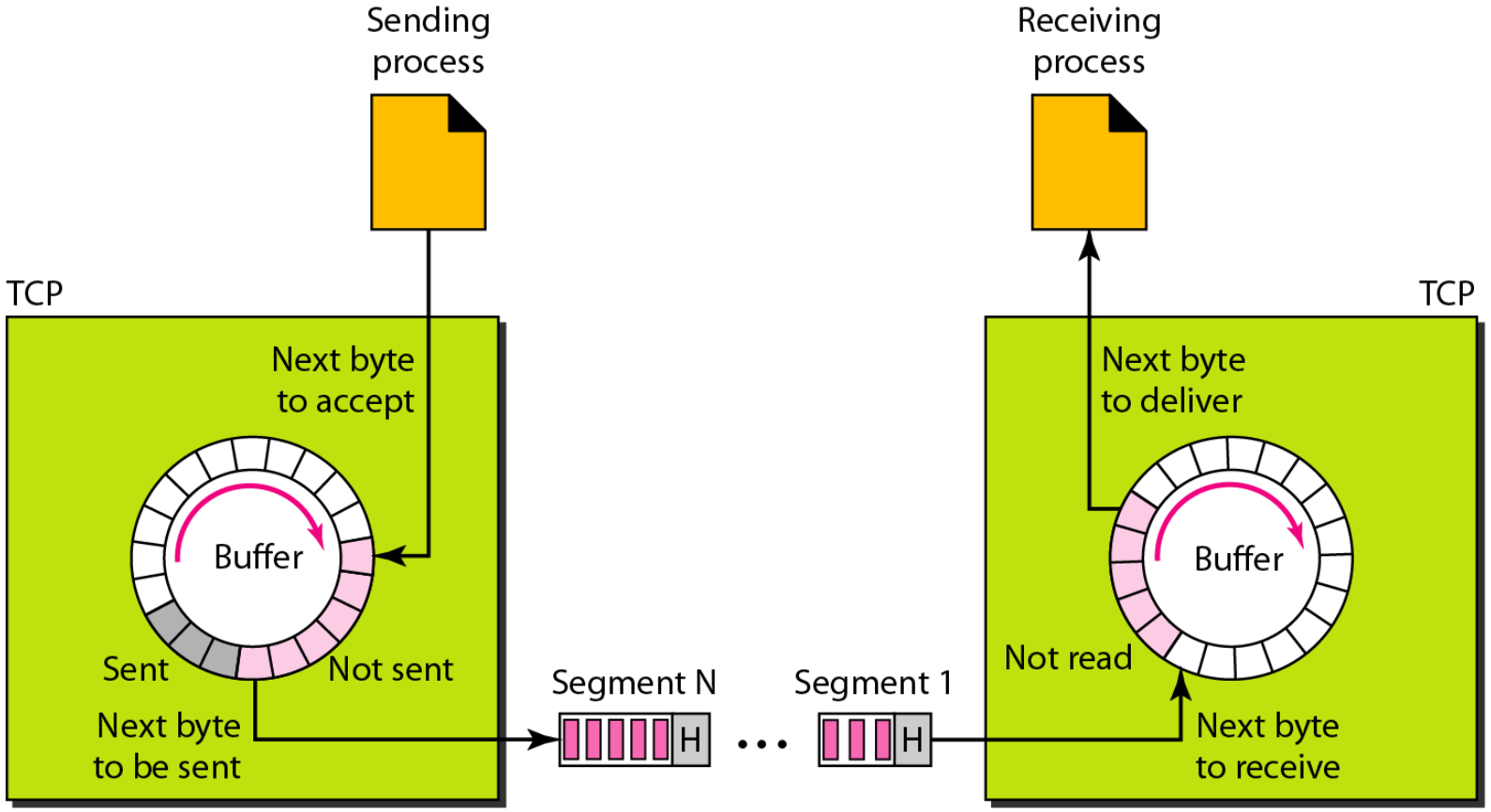


- TCP allows the sending process to deliver data as a stream of bytes
- Allows the receiving process to obtain data as a stream of bytes.
- TCP creates an environment in which the two processes seem to be connected by an imaginary "tube" that carries their data across the Internet.

Sending and receiving buffers



TCP 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.

Note

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

Example

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)**

Note

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

TCP

- *Full-Duplex Communication*
  - TCP offers full-duplex service, in which data can flow in both directions at the same time.
  - Each TCP then has a sending and receiving buffer, and segments move in both directions.

- *Reliable Service*
  - TCP is a reliable transport protocol.
  - It uses an acknowledgment mechanism to check the safe and sound arrival of data.

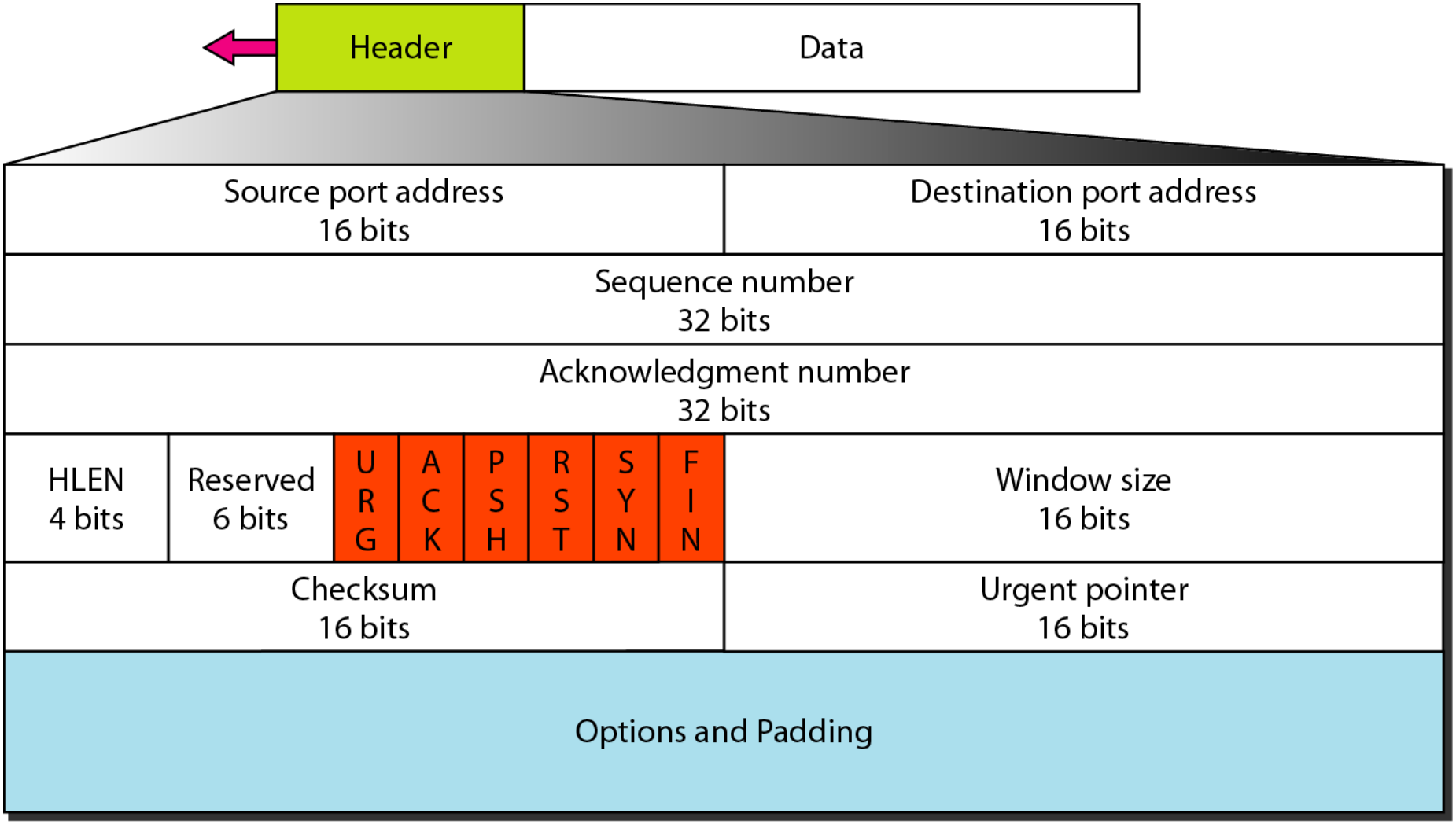
TCP

- *Connection-Oriented Service*
  - TCP, unlike UDP, is a connection-oriented protocol.
  - When a process at site A wants to send and receive data from another process at site B, the following occurs:
    - The two TCPs establish a connection between them.
    - Data are exchanged in both directions.
    - The connection is terminated.

Note

- **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



Sequence number.

- This 32-bit field defines the number assigned to the *first byte of data contained in this segment.*
- To ensure connectivity, each byte to be transmitted is numbered.
- The sequence number tells the destination which byte in this sequence comprises the first byte in the segment.
- During connection establishment, each party uses a random number generator to create an initial sequence number (ISN), which is usually different in each direction.

Acknowledgment number

- This 32-bit field defines the byte number that the receiver of the segment is *expecting to receive from the other party*.
- If the receiver of the segment has successfully received byte number x from the other party, it defines $x + 1$ as the acknowledgment number.
- Acknowledgment and data can be *piggybacked together.*

Header length

- This 4-bit field indicates the number of 4-byte words in the TCP header.
- The length of the header can be between 20 and 60 bytes.
- Therefore, the value of this field can be between 5 (5 x 4 =20) and 15 (15 x 4 =60).

Control fields

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

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. **Push data immediately to upper layer**
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

Window size

- This field defines the size of the window, in bytes, that the other party must maintain.
- Length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes.
- This value is normally referred to as the receiving window (rwnd) and is determined by the receiver.
- The sender must obey the dictation of the receiver in this case.

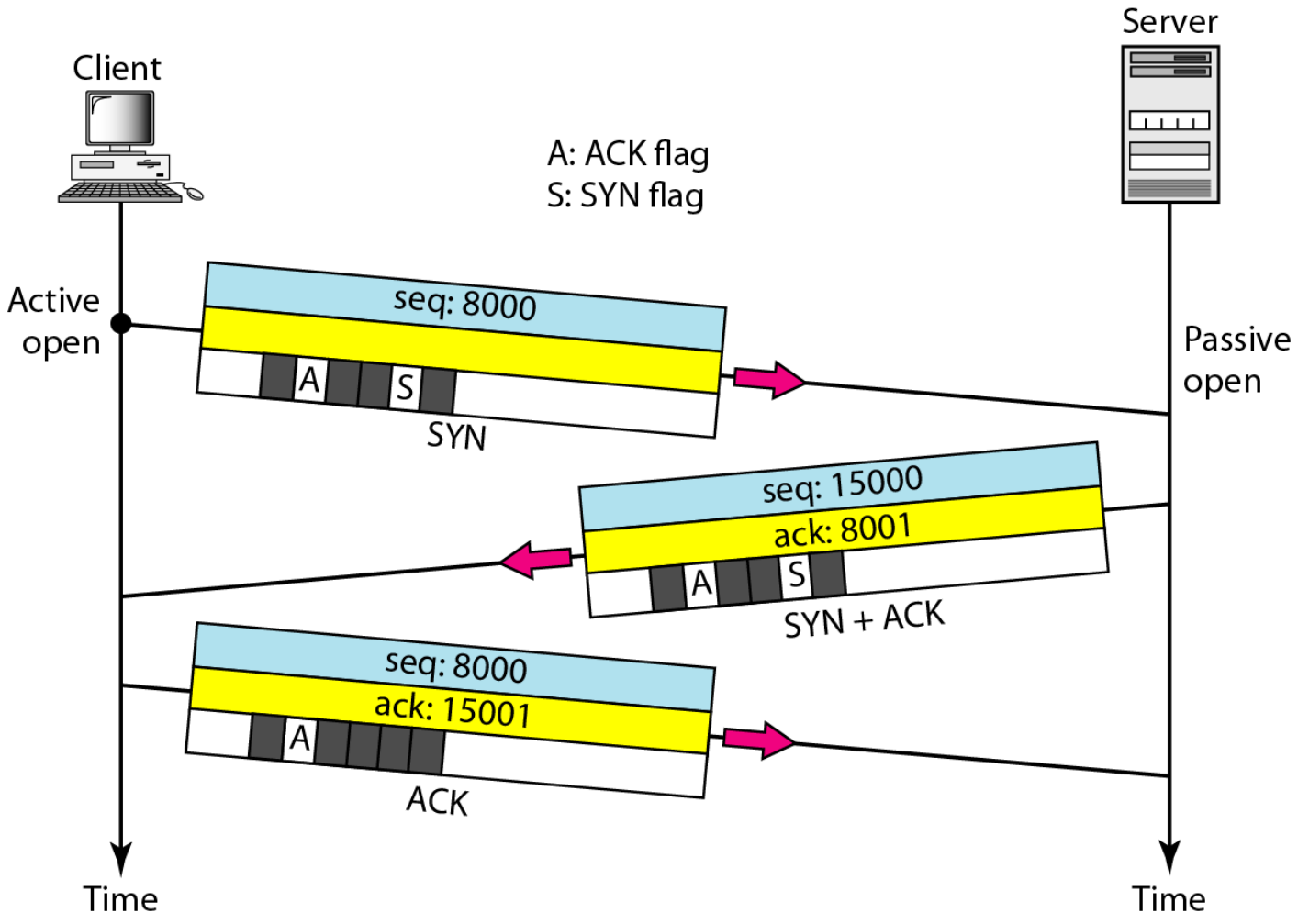
Urgent pointer

- This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data.
- It defines the number 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

- This normally specifies the acceptable ***maximum Segment Size*(MSS)**
- This defines maximum number of user data octets that a TCP segment can carry
- Some other QoS parameters like Precedence, Delay, Throughput can also be forwarded during connection establishment

Connection establishment using three-way handshaking



Note

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

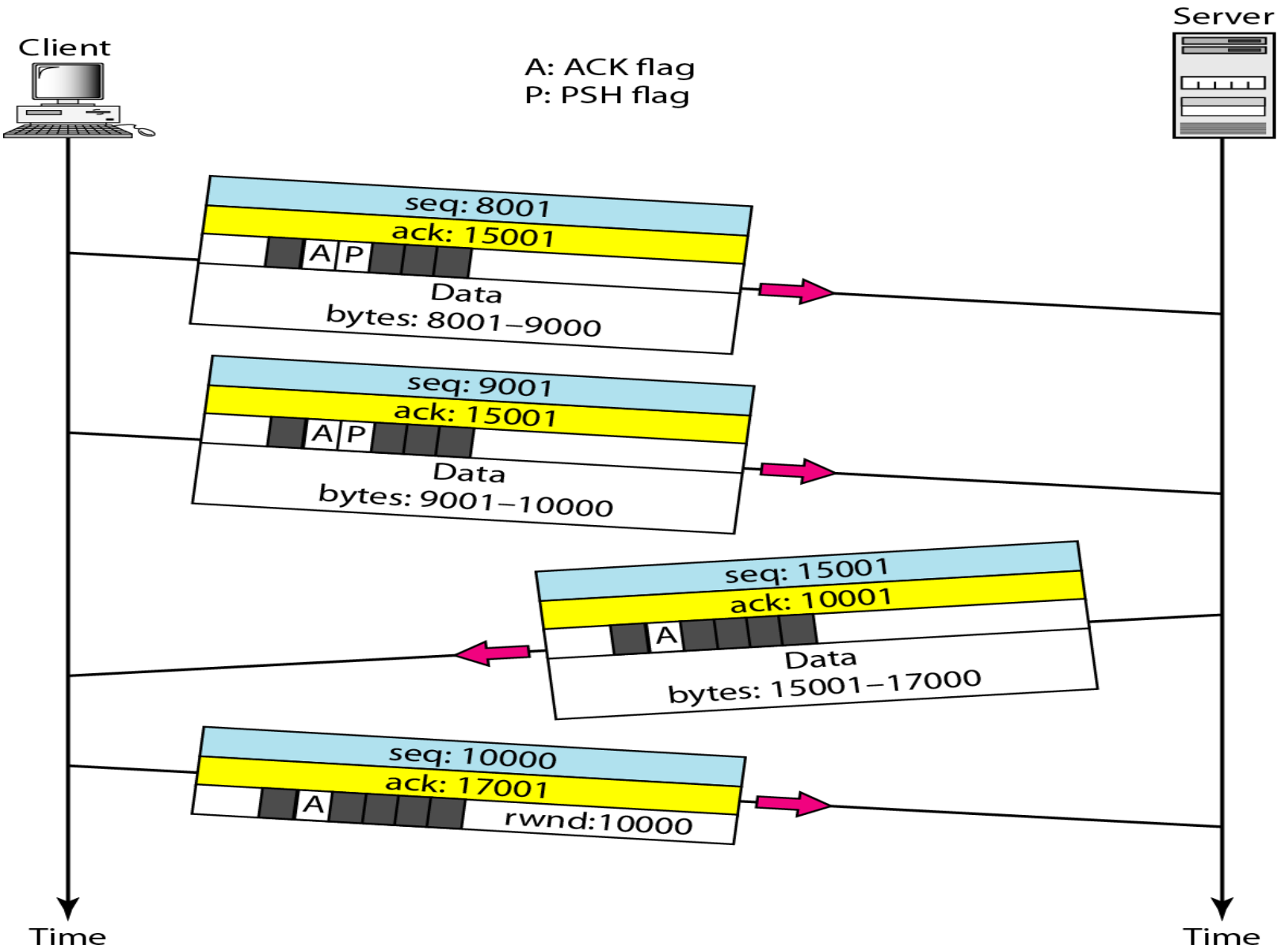
Note

A SYN + ACK segment cannot carry data, but does consume one sequence number.

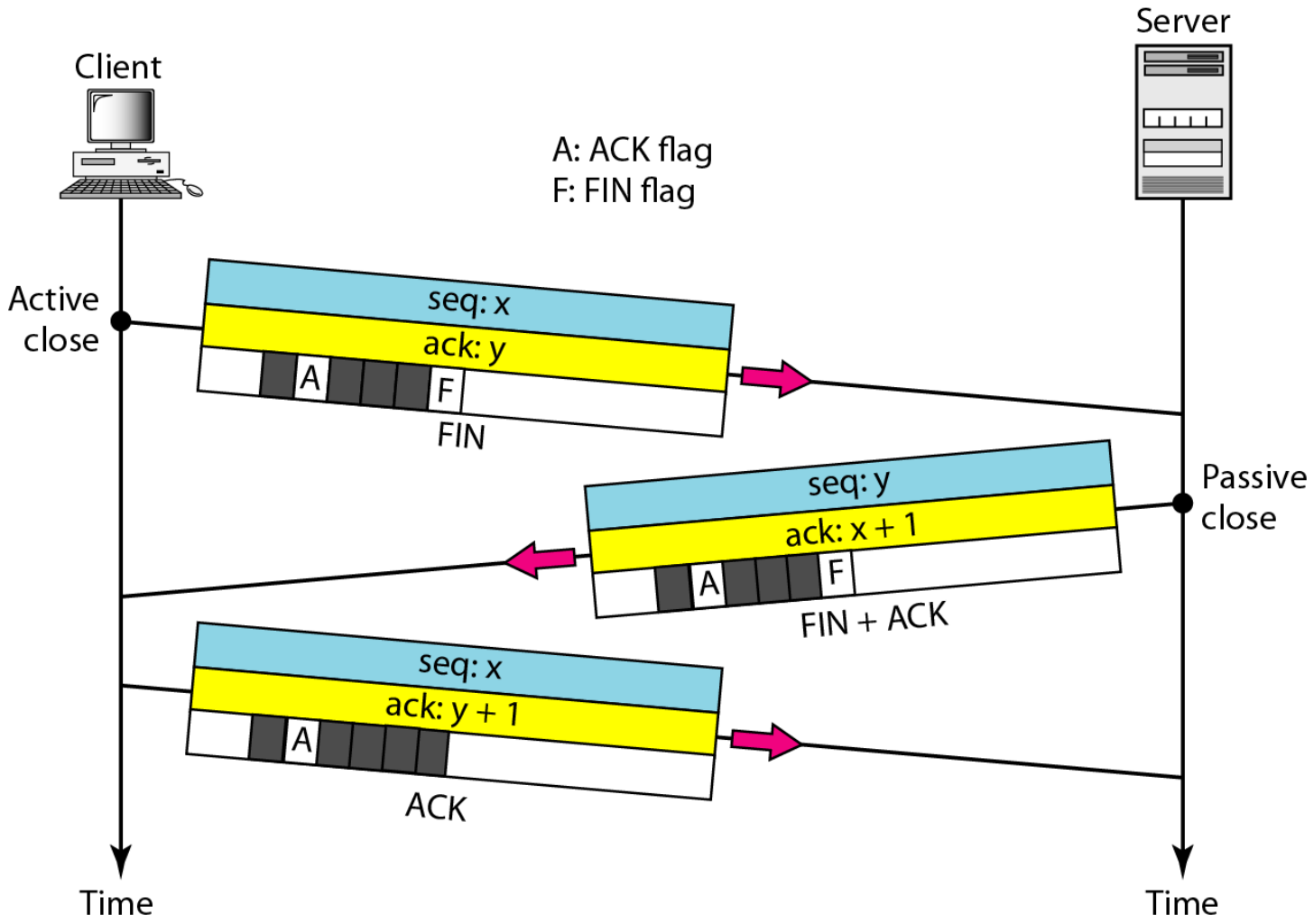
Note

An ACK segment, if carrying no data, consumes no sequence number.

Data transfer



Connection termination using three-way handshaking



Note

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

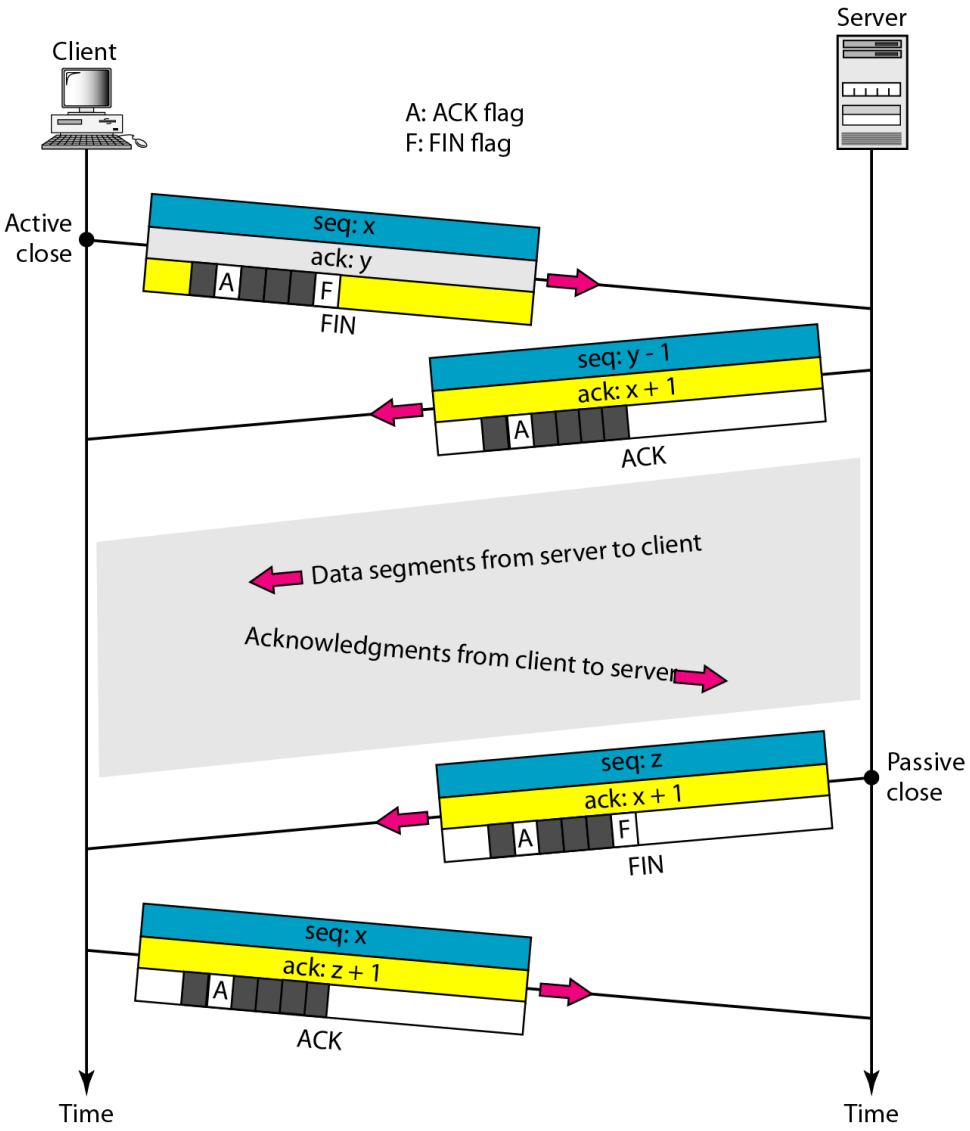
Note

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

Half-Close

- In TCP, one end can stop sending data while still receiving data.
- This is called a half-close. Although either end can issue a half-close, it is normally initiated by the client.
- It can occur when the server needs all the data before processing can begin.

Half-close

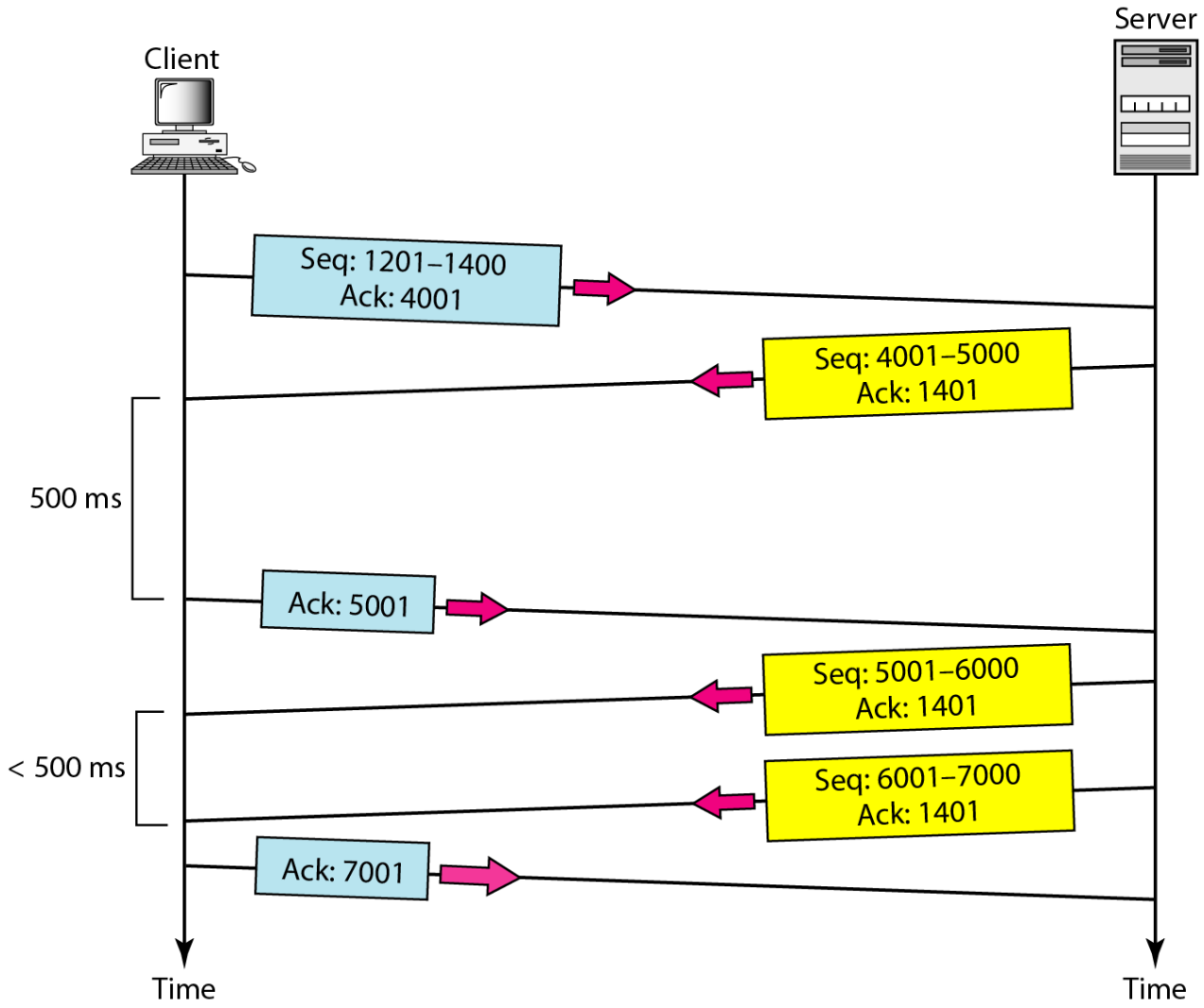

Client
Server
A: ACK flag
F: FIN flag
Active close
seq: x
ack: y
A
F
FIN
seq: y - 1
ack: x + 1
A
ACK
Data segments from server to client
Acknowledgments from client to server
seq: z
ack: x + 1
A
F
FIN
Passive close
seq: x
ack: z + 1
A
ACK
Time
Time

Note

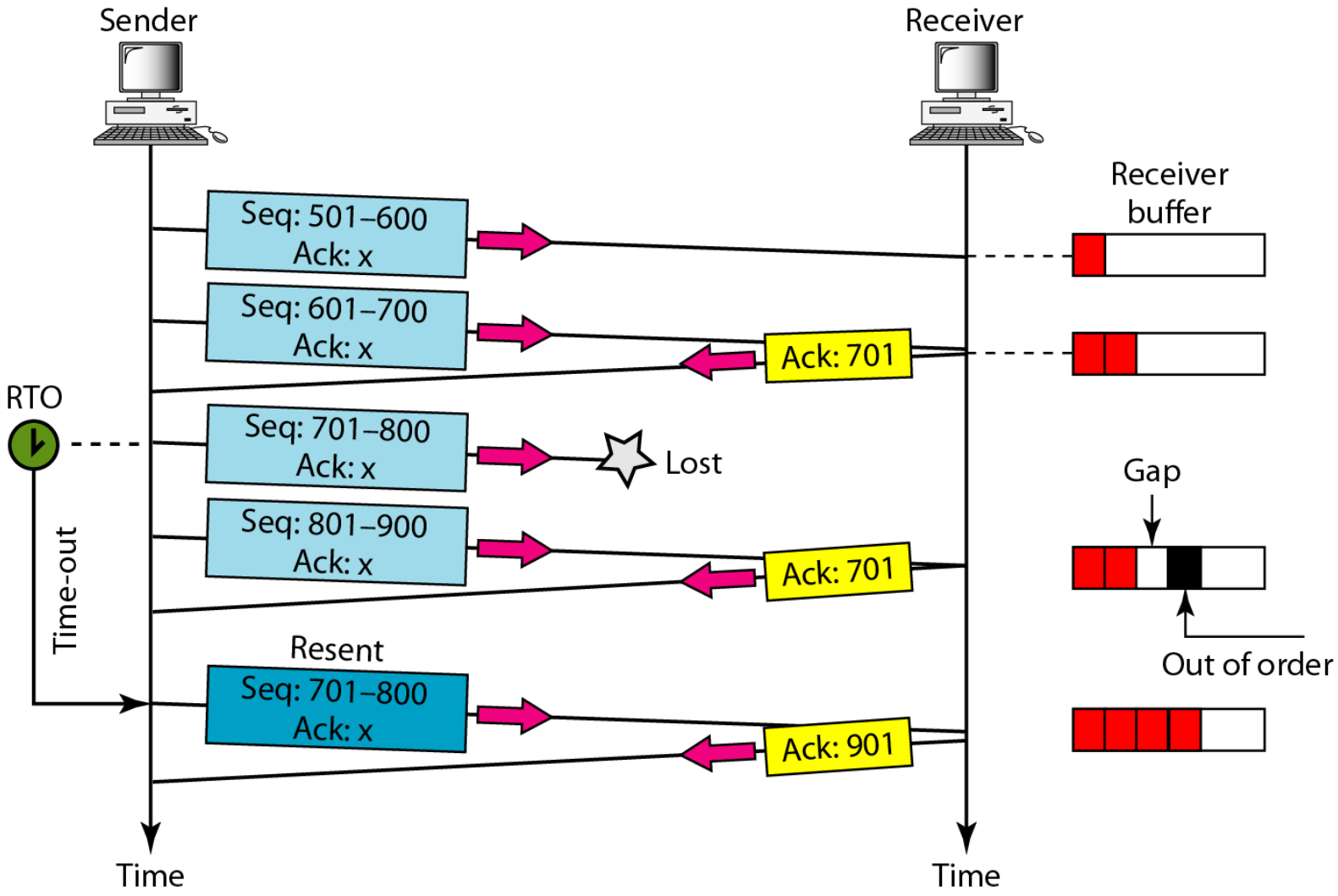
- **A sliding window is used to make transmission more efficient as well as to control the flow of data so that the destination does not become overwhelmed with data.**

- **TCP sliding windows are byte-oriented.**

Normal operation



Lost segment



Note

The receiver TCP delivers only ordered data to the process.

Fast retransmission