Process-to-Process Delivery: UDP, TCP, and SCTP

Chapter-23

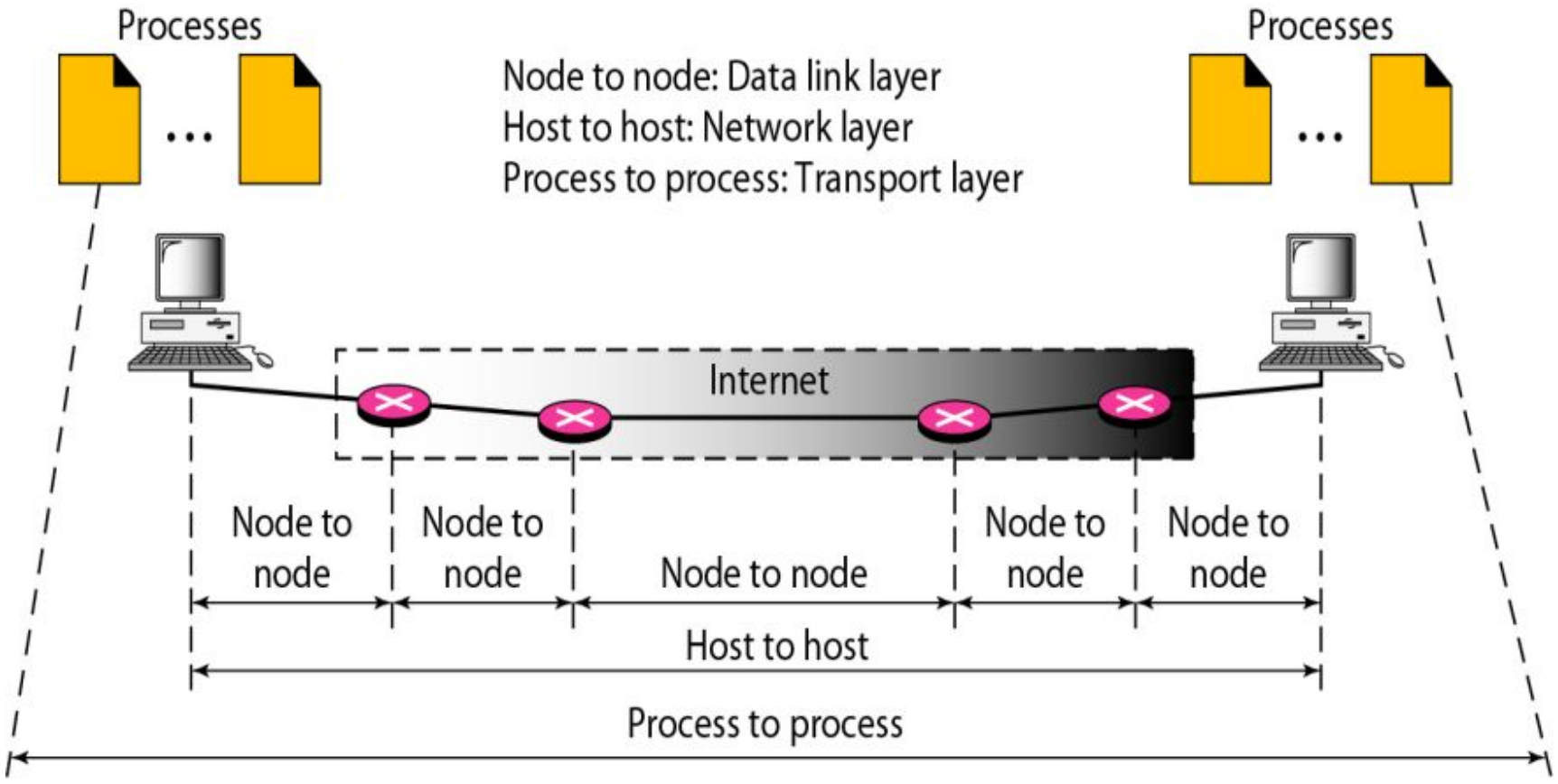
Introduction

- The transport layer is responsible for the delivery of a message from one process to another
- The transport layer header must include a service – point –address in the OSI model and port number in the TCP/IP (internet model)
- The Internet model has three protocols at the transport layer: **UDP, TCP, and SCTP**.
  - **UDP(User Datagram Protocol)**: Is the simplest of the three.
  - **TCP(Transmission Control Protocol)**: A complex transport layer protocol.
  - **SCTP(Stream Control Transmission Protocol):** The new transport layer protocol that is designed for specific applications such as multimedia.

Process-to-process Delivery

- The Data link layer is responsible for delivery of frames between nodes over a link node to node delivery
- The network layer is responsible for delivery of datagrams between two hosts host to host delivery
- Real communication takes place between two processes (application programs). We need process-to-process delivery.
- We need a mechanism to deliver data from one of processes running on the source host to the corresponding process running on the destination host.
- The transport layer is responsible for process-to-process delivery

Types of data deliveries



Client/Server Paradigm

- Process-to-process communication can be achieved through client/server
- A process on the local host, called a client, needs services from a process usually on the remote host, called a server.
- Both processes (client and server) have the same name.
  - For example, to get the day and time from a remote machine, we need a Daytime client process running on the local host and a Daytime server process running on a remote machine.
- A remote computer can run several server programs at the same time, just as local computers can run one or more client programs at the same time.
- For communication, we must define :
  1. Local host
  2. Local process
  3. Remote host
  4. Remote process

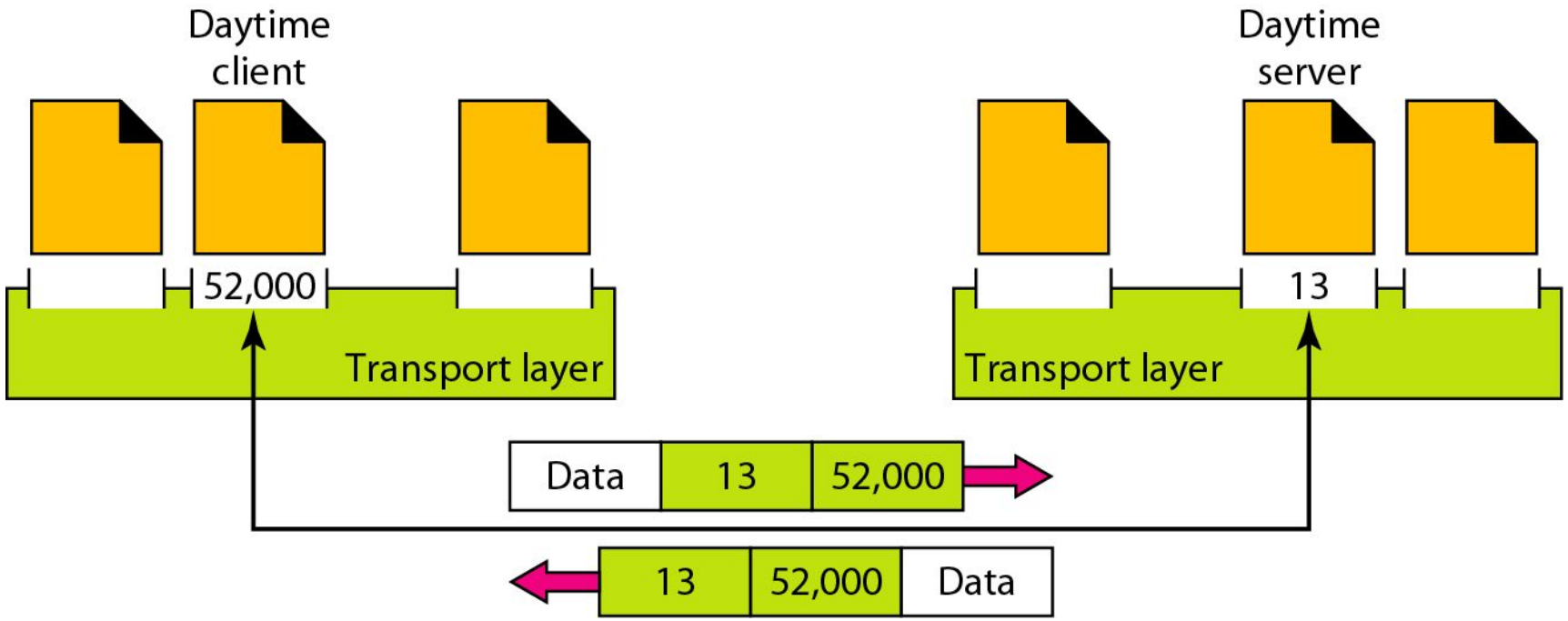
Addressing

- At the data link layer, we need a MAC address to choose one node among several
- At the network layer, we need an IP address to choose one host among millions.
- At the transport layer, we need a port number, to choose among multiple processes running on the destination host.
- The destination port number is needed for delivery; the source port number is needed for the reply.
- In the Internet model, the port numbers are 16-bit integers between 0 and 65,535.

Port number

- The client program defines itself with a port number, chosen randomly by the transport layer software running on the client host
- The server process must also define itself with a port number This port number, however, cannot be chosen randomly
- The Internet uses port numbers for servers called well- known port numbers.
- Every client process knows the well-known port number of the corresponding server process
  - For example, while the Daytime client process, can use an ephemeral (temporary) port number 52,000 to identify itself, the Daytime server process must use the well-known (permanent) port number 13.

Port number

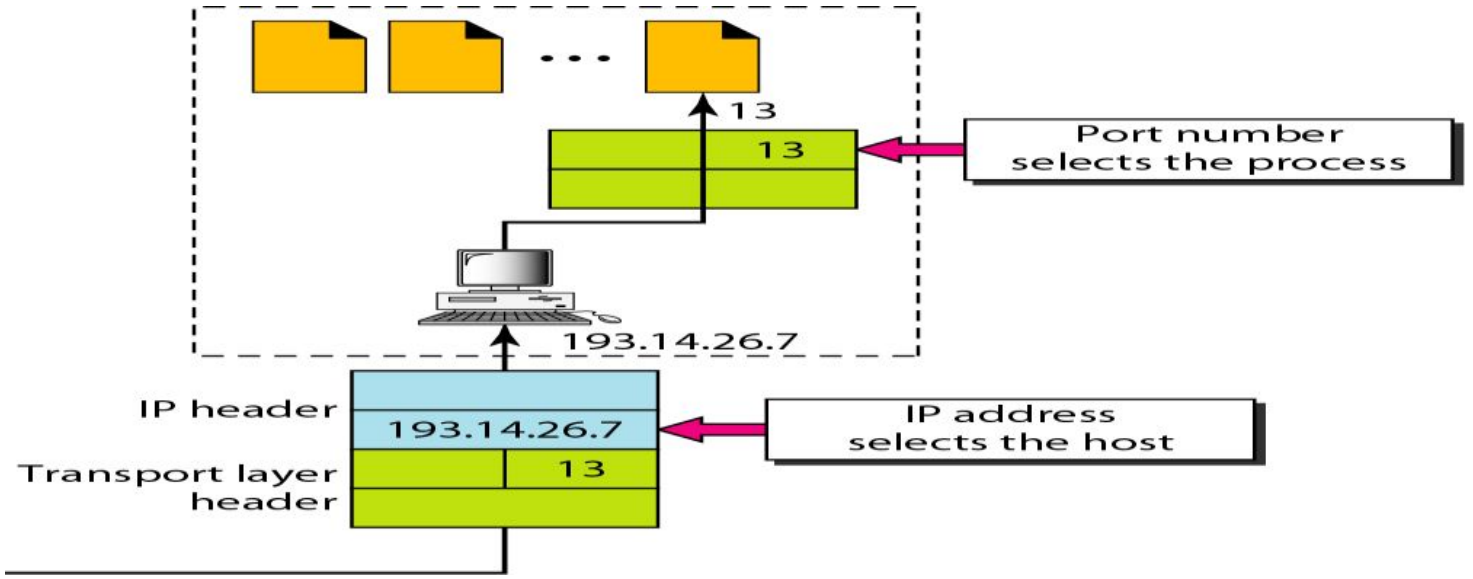

Daytime
client
52,000
Transport layer
Daytime
server
13
Transport layer
Data
13
52,000
13
52,000
Data

lANA Ranges

- **The lANA (Internet Assigned Number Authority) has divided the port numbers into three ranges:**
  - **Well- known ports:**
    - **The ports ranging from 0 to 1023 are assigned and controlled by IANA**
  - **Registered 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 (or private):**
    - **ranging from 49,152 to 65,535 are neither controlled nor registered. They can be used by any process. These are the ephemeral port**

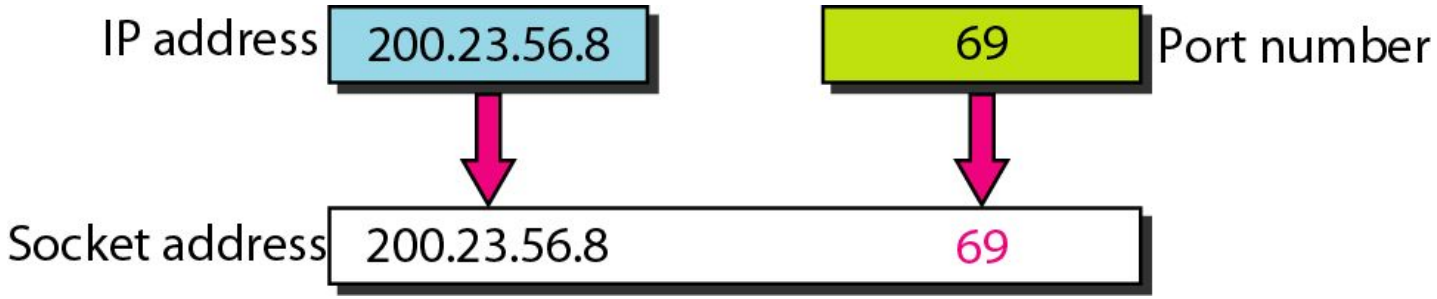
IP addresses versus port numbers

- **IP addresses and port numbers play different roles in selecting the final destination of data.**
- The destination IP address defines the host among the different hosts
- **After the host has been selected, the port number defines one of the processes on this particular host**

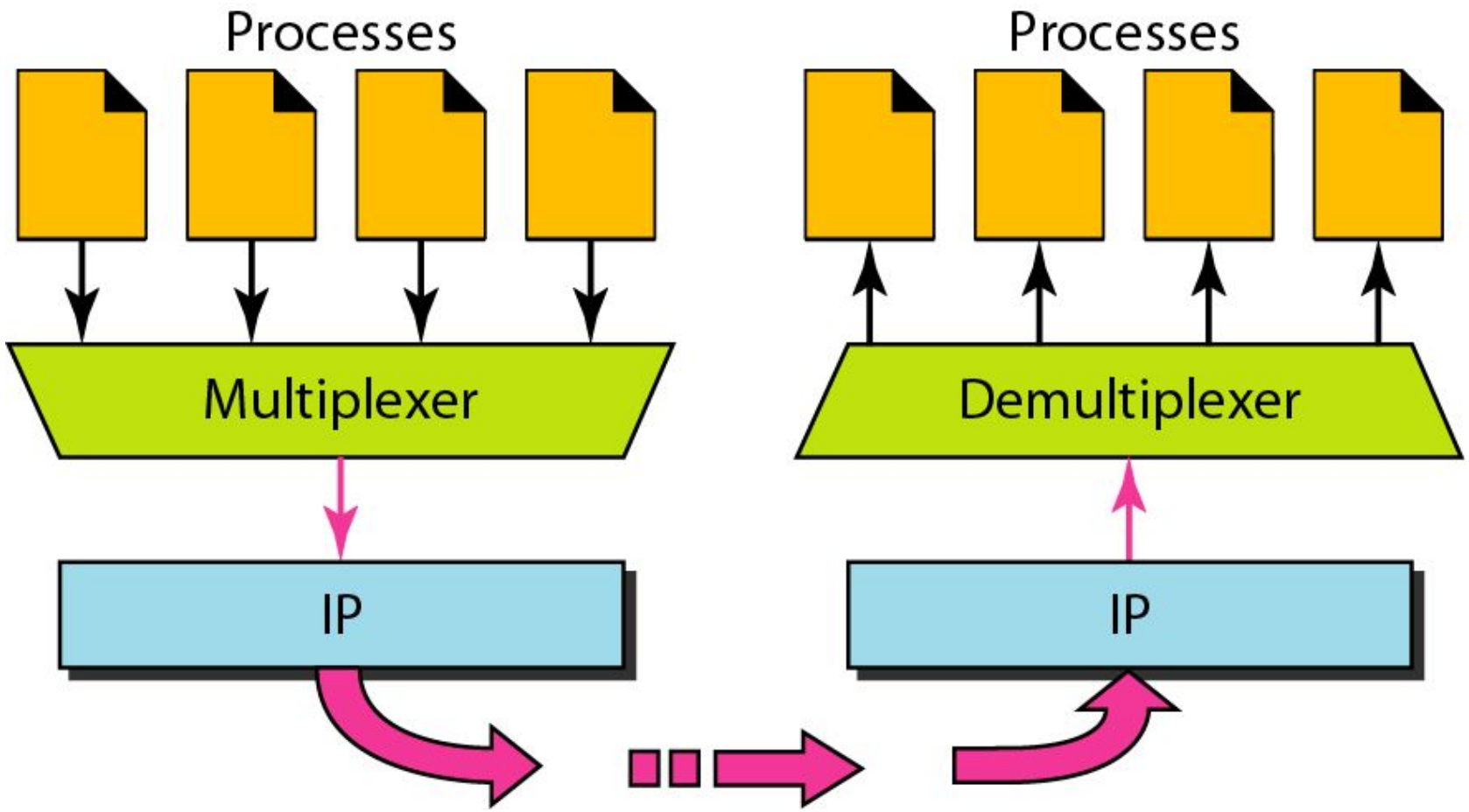


Socket Addresses

- 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.
- These four pieces of information are part of the IP header and the transport layer protocol header.
  - The IP header contains the IP addresses; the UDP or TCP header contains the port numbers.



Multiplexing and Demultiplexing



Connectionless Versus 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.
  - The packets are not numbered; they may be delayed or lost or may arrive out of sequence.
  - There is no acknowledgment .
- UDP is a connectionless transport layer protocols.

Connectionless Versus Connection-Oriented Service

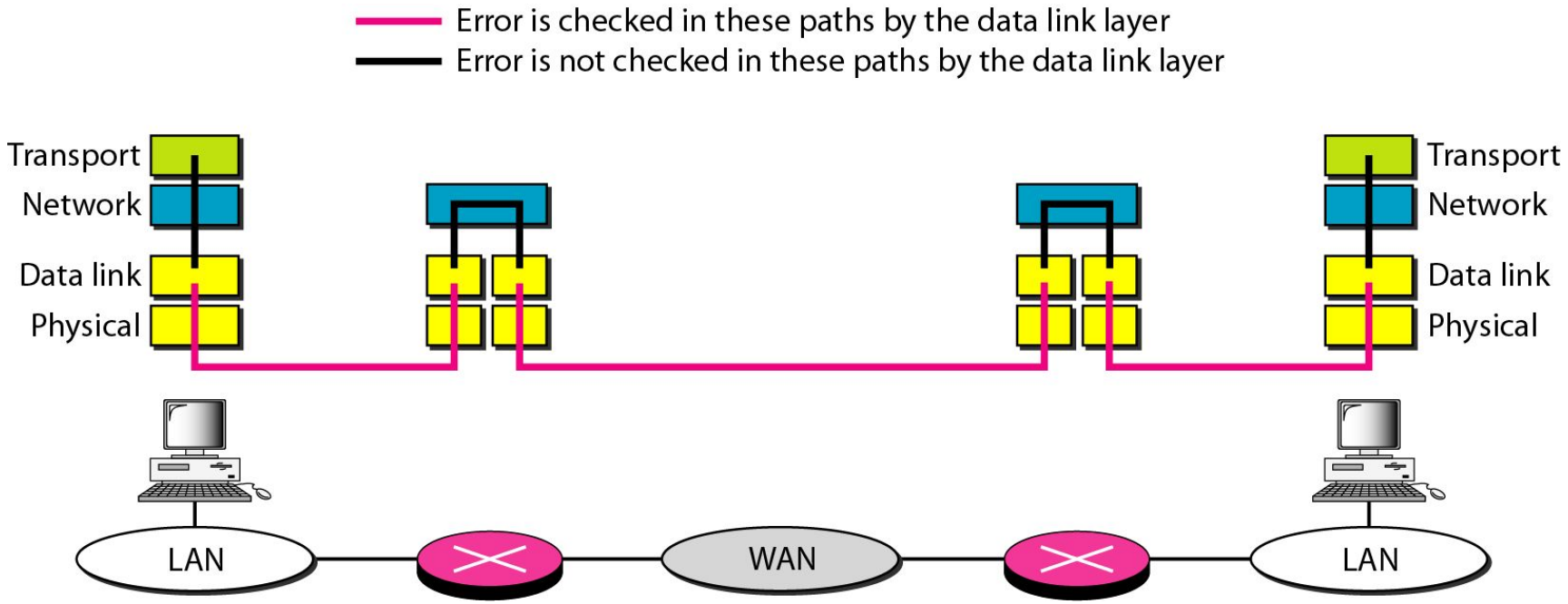
- Connection Oriented Service
  - In a connection-oriented service, a connection is first established between the sender and the receiver.
  - Data are transferred.
  - At the end, the connection is released. TCP and SCTP are connection-oriented protocols.

Reliable Versus Unreliable

- The transport layer service can be reliable or unreliable.
- If the application layer program needs reliability, we use a reliable transport layer protocol by implementing flow and error control at the transport layer. This means a slower and more complex service.
- On the other hand, if the application program does not need reliability then an unreliable protocol can be used.
- UDP is connectionless and unreliable;
- TCP and SCTP are connection oriented and reliable.
- These three protocols can respond to the demands of the application layer programs.

Error control

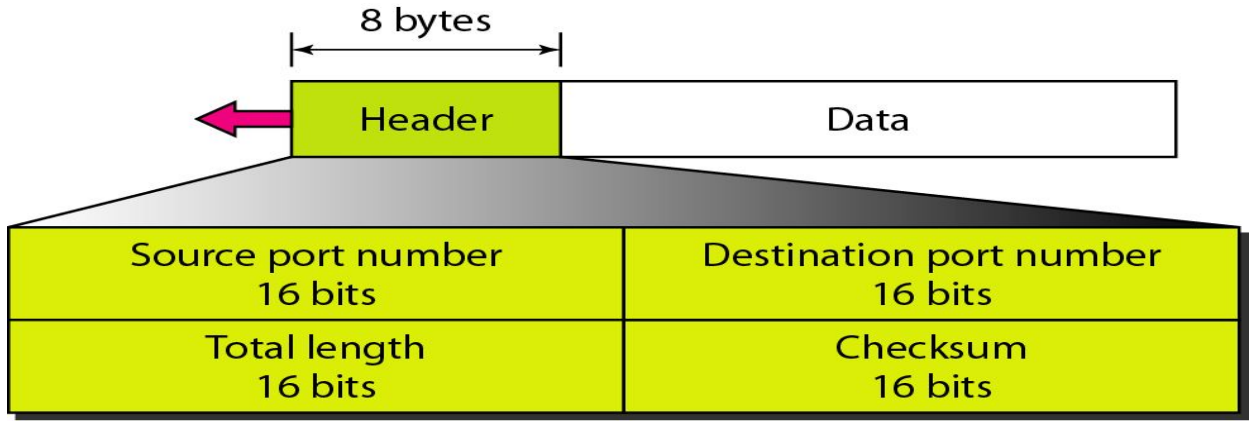
If the data link layer is reliable and has flow and error control, do we need this at the transport layer, too? Yes



User Datagram Protocol (UDP)

- UDP is 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.
- UDP is a 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.
- Sending a small message by using UDP takes much less interaction between the sender and receiver than using TCP or SCTP.

User Datagram



- UDP packets, called user datagrams, have a fixed size header of 8 bytes.
- Source port number: This is the port number used by the process running on the source host.
- Destination port number: This is the port number used by the process running on the destination host.
- Length: This is a 16-bit field that defines the total length of the user datagram.
- Checksum: This field is used to detect errors over the entire user datagram (header plus data). The inclusion of the checksum in the UDP datagram is optional

UDP Operation

UDP provides a connectionless service

- no relationship between the different user datagram even if they are coming from the same source process and going to the same destination program.
- Also, there is no connection establishment and no connection termination.
- Each user datagram can travel on a different path.
- The user datagrams are **not numbered.**
- Each UDP user datagram request must be small enough to fit into one user datagram. Only those processes sending short messages should use UDP.

UDP Operation

Flow and Error Control

- UDP is a very simple, unreliable transport protocol.
- There is no flow control: The receiver may overflow with incoming messages.
- There is no error control mechanism in UDP except for the checksum.
- The sender does not know if a message has been lost or duplicated.
- When the receiver detects an error through the checksum, the user datagram is **discarded.**

Use of UDP

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.
- UDP is suitable for a process with internal flow and error control mechanisms. For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control..
- UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.
- UDP is used for management processes such as SNMP
- UDP is used for some route updating protocols such as Routing Information Protocol (RIP)

Transmission Control Protocol(TCP)

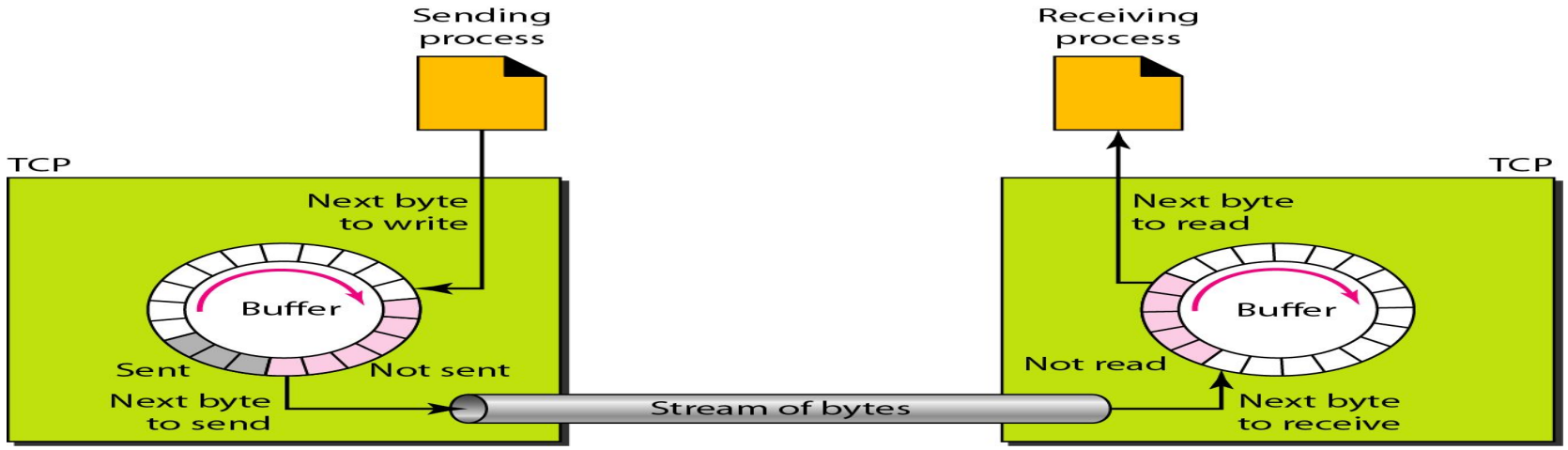
- TCP, like UDP, is a process-to-process (program-to-program) protocol uses port numbers.
- Unlike UDP, TCP is a connection oriented protocol; it creates a virtual connection between two TCPs to send data.
- TCP uses flow and error control mechanisms at the transport level.

Stream Delivery Service

- TCP allows the sending process to deliver data as a stream of bytes and 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. The sending process produces (writes to) the stream of bytes, and the receiving process consumes (reads from) them.

Sending and receiving buffers

- The sending and the receiving processes may not write or read data at the same speed, TCP needs buffers for storage.
- Two buffers, the sending buffer and the receiving buffer, one for each direction.
- These buffers are also necessary for flow and error control mechanisms used by TCP.
- One way to implement a buffer is to use a circular array of 1-byte locations



Sending and receiving buffers

The sending buffer has three types of chambers:

- **The white section :**

  Contains empty chambers that can be filled by the sending process

- **The gray section :**
  - holds bytes that have been sent but not yet acknowledged.
  - TCP keeps these bytes in the buffer until it receives an acknowledgment.
- **The colored section :**
  - Contains bytes to be sent by the sending TCP.
  - TCP may be able to send only part of this colored section.
  - This could be due to the slowness of the receiving process or to congestion in the network.
  - After the bytes in the gray chambers are acknowledged, the chambers are recycled and available for use by the sending process.

Sending and receiving buffers

The circular buffer at the receiver is divided into two areas:

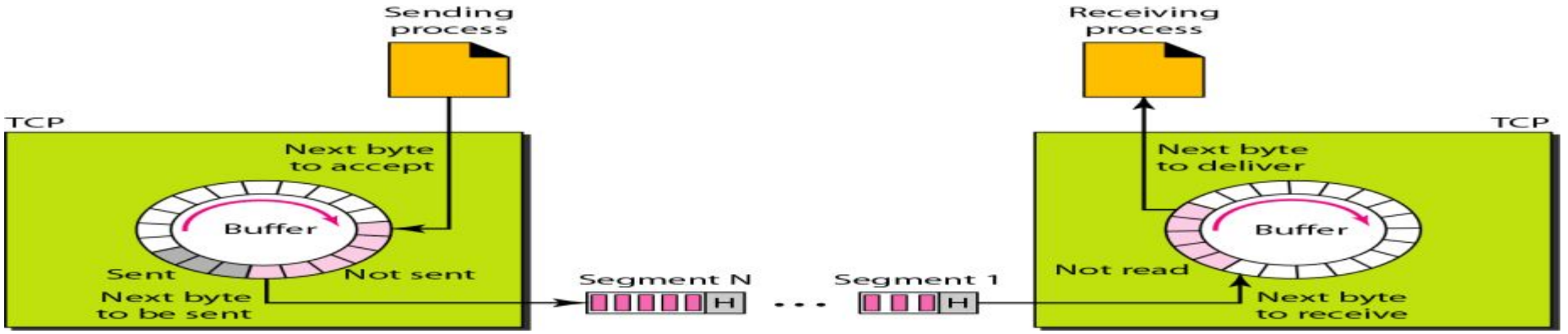
- **The white area :**

  Contains empty chambers to be filled by bytes received from the network.

- **The colored sections:**
  - Contain received bytes that can be read by the receiving process.
  - When a byte is read by the receiving process, the chamber is recycled and added to the pool of empty chambers

Segments

- The IP layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes.
- the transport layer, TCP groups a number of bytes together into a packet called a segment.
- TCP adds a header to each segment and delivers the segment to the IP layer for transmission.
- 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



Connection-oriented Services

When a process at site A wants to send and receive data from another process at site B, the following occurs:

1. The two processes establish a connection between them.(virtual connection , not a physical connection)
2. Data are exchanged in both directions.
3. The connection is terminated.

Reliable Transport Protocol

- **TCP is a reliable transport protocol.**

  It uses an acknowledgment mechanism to check the safe arrival of data.

- **Flow Control:**

  The receiver of the data controls the amount of data that are to be sent by the sender.

- **Error Control.**

  To provide reliable service, TCP implements an error control mechanism.

- **Congestion Control:**

  The amount of data sent by a sender is controlled by the level of congestion in the network.

TCP Features

Numbering System:

- There are two fields called the sequence number and the acknowledgment number.
- These two fields refer to the byte number and not the segment number.
- TCP numbers all data bytes that are transmitted in a connection.
- Numbering is independent in each direction.
- When TCP receives bytes of data from a process, it stores them in the sending buffer and numbers them.
- The numbering does not necessarily start from 0.
- TCP generates a random number between 0 and 232 - 1 for the number of the first byte.
- Byte numbering is used for flow and error control. For example: if the random number happens to be 1057 and the total data to be sent are 6000 bytes, the bytes are numbered from 1057 to 7056.

Notes

- The bytes of data being transferred in each connection are numbered by TCP.
- The numbering starts with a randomly generated number.
- After the bytes have been numbered, TCP assigns a sequence number to each segment that is being sent.
- The sequence number for each segment is the number of the first byte carried in that segment

Example 23.3

- **Suppose a TCP connection is transferring a file of 5000 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 1000 bytes?**

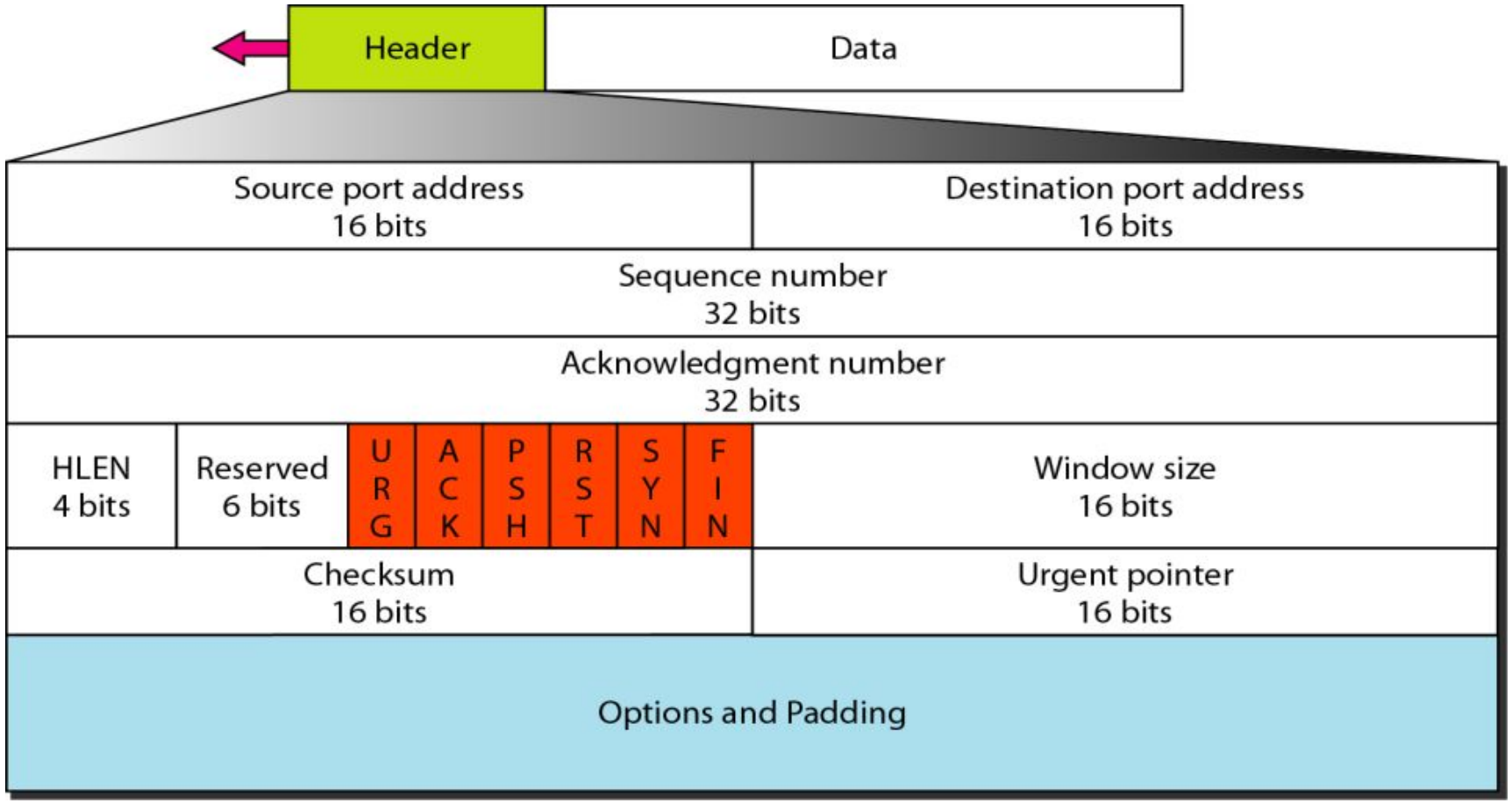
The following shows the sequence number for each segment:

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)**

Acknowledgment Number

- Each party uses an acknowledgment number to confirm the bytes it has received.
- The acknowledgment number defines the number of the next byte that the party expects to receive.
- The acknowledgment number is cumulative, which means that the party takes the number of the last byte that it has received, safe adds 1 to it. This sum is the acknowledgment number. For example :If the receiver of the segment has successfully received byte number x from the other party, it defines x + 1 as the acknowledgment number.
- The term cumulative here means that if a party uses 5643 as an acknowledgment number, it has received all bytes from the beginning up to 5642.
- Note that this does not mean that the party has received 5642 bytes because the first byte number does not have to start from 0.

TCP Segment Format



TCP Segment Format

- The segment consists of a 20-60-byte header.
- Source port address:

  This is a 16-bit field , it defines the port number of the application program in the host that is sending the segment.
- Destination port address:

  This is a 16-bit field, it defines the port number of the application program in the host that is receiving the segment.

TCP Segment Format

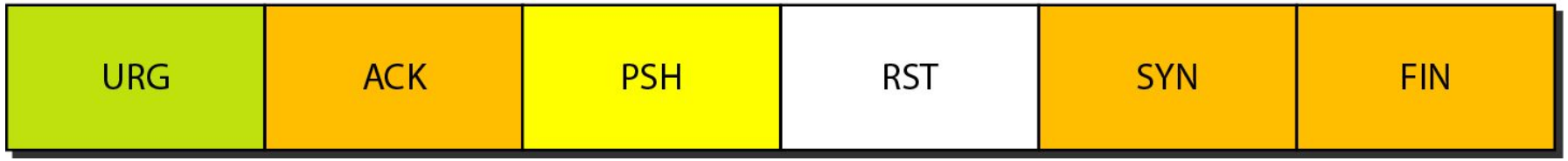
- Sequence number: This 32-bit field defines the number assigned to the first byte of data contained in this segment.
- Acknowledgment number: This 32 bit field defines the number of the next byte a party expects to receive.
- Header length: A 4-bit field that indicates the number of 4-byte words in 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).
- Reserved. This is a 6-bit field reserved for future use.

TCP Segment Format

- Control: This field defines 6 different control bits or flags. One or more of these bits can be set at a time. These bits enable flow control, connection establishment and termination, connection abortion, and the 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 Segment Format

- Window size: Defines the size of the window, in bytes, that the other party must maintain. the 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.
- Checksum: This 16-bit field contains the checksum. The inclusion of the checksum for TCP is mandatory.
- Options: There can be up to 40 bytes of optional information in the TCP header.

TCP Segment Format

- Urgent data :
- The data are presented from the application program to TCP as a stream of bytes.
- Each byte of data has a position in the stream.
- on occasion an application program needs to send urgent bytes.
- This means that the sending application program wants a piece of data to be read out of order by the receiving application program.
- The sending TCP creates a segment and inserts the urgent data at the beginning of the segment.
- The rest of the segment can contain normal data from the buffer
- The urgent pointer field in the header defines the end of the urgent data and the start of normal data.

TCP Segment Format

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.

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 urgent pointer, and delivers them, out of order, to the receiving application program.

TCP Connection

- A Connection-oriented transport protocol establishes a virtual path between the source and destination.
- In TCP, connection-oriented transmission requires three phases:
    1. connection establishment
    2. data transfer
    3. connection termination

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.
- Each party must initialize communication and get approval from the other party before any data are transferred.
- The connection establishment in TCP is called three way handshaking.

Example: Client-server communication using TCP as the transport layer protocol.

1.The server issues a request for a passive open: The server program tells its TCP that it is ready to accept a connection.

2.The client program issues a request for an active open: A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server. TCP can now start the three-way handshaking process

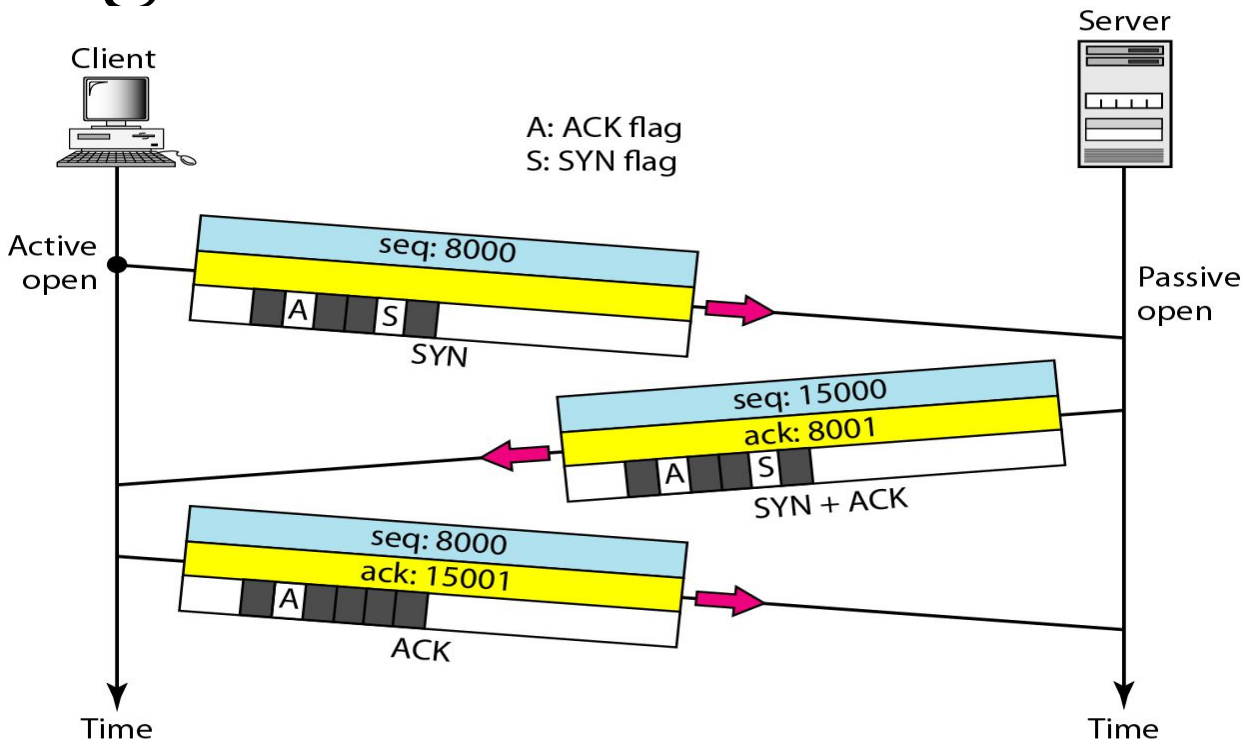
Three-way handshaking process

1.The client sends the first segment, a SYN segment, in which only the SYN flag is set.
   - This segment is for synchronization of sequence numbers. It consumes one sequence number.
   - When the data transfer starts, the sequence number is incremented by 1.
   - The SYN segment carries no real data

2. The server sends the second segment, a SYN +ACK segment, with 2 flag bits set: SYN and ACK.
   - This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment.
   - It consumes one sequence number.

3. The client sends the third segment. This is just an ACK segment.
   - It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field.
   - The sequence number in this segment is the same as the one in the SYN segment.
   - The ACK segment does not consume any sequence numbers.

Connection establishment using three-way handshaking



Practice Question: *TCP opens a connections using an initial sequence number (ISN) of 14534. The other party opens the connection with an ISN of 21732. Show the three TCP segment during the connection establishment.*

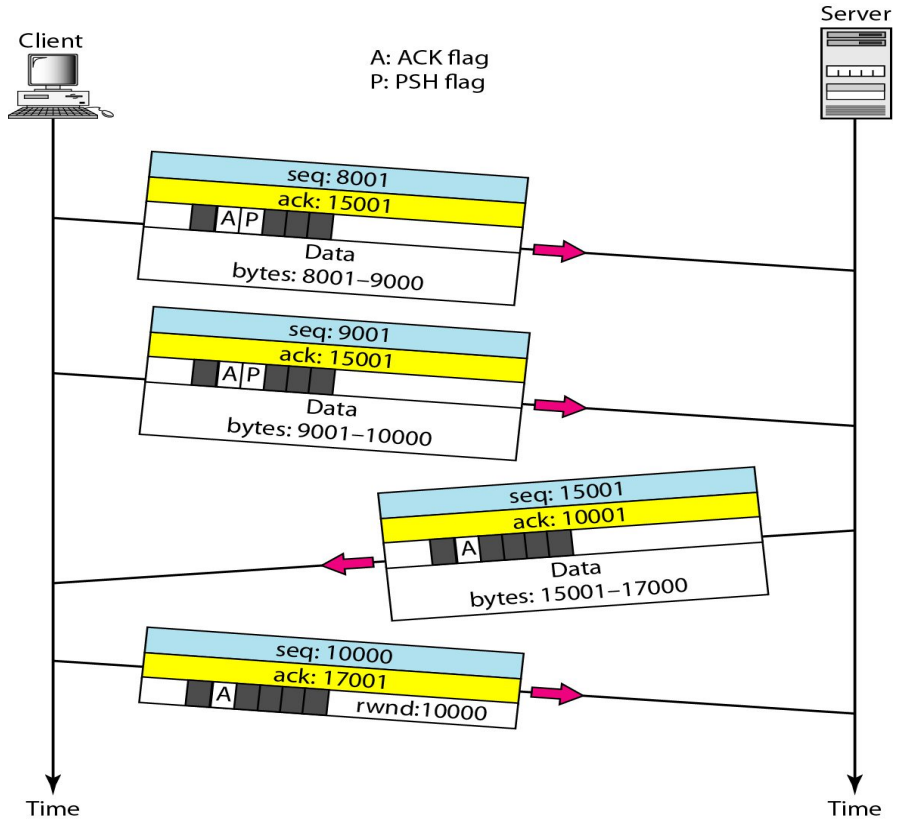
Data transfer

- After connection is established, bidirectional data transfer can take place. The client and server can both send data and acknowledgments
- The acknowledgment is piggybacked with the data.

Example:

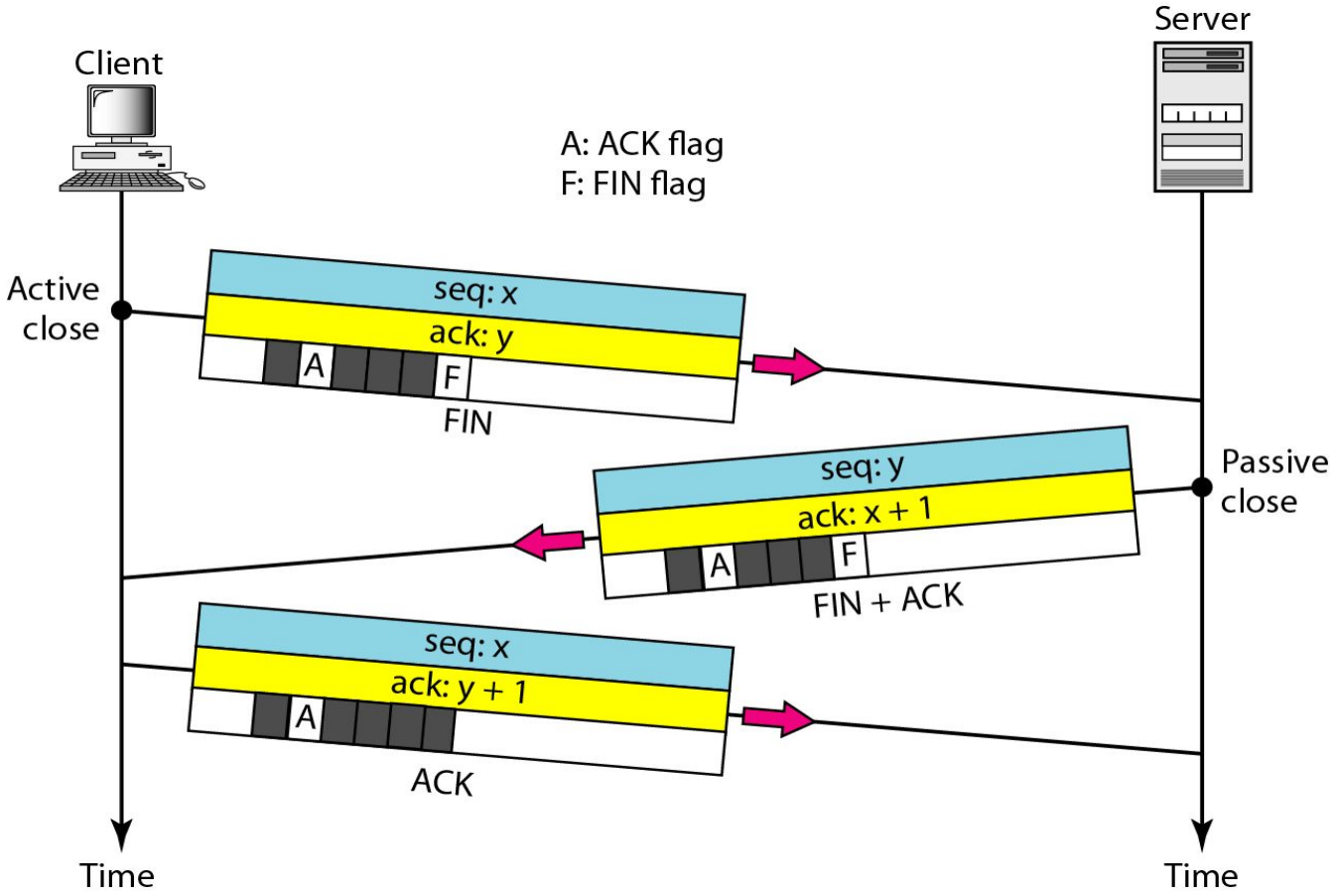
- After connection is established, the client sends 2000 bytes of data in two segments.
- The server then sends 2000 bytes in one segment.
- The client sends one more acknowledgment segment.
- The first three segments carry both data and acknowledgment, but the last segment carries only an acknowledgment because there are no more data to be sent.

Data transfer



Connection Termination Using Three-way Handshaking

Any of the two parties involved in exchanging data (client or server) can close the connection using three-Way Handshaking



Three-way Handshaking steps

1. The client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set.

- A FIN segment can include the last chunk of data sent by the client, or it can be just a control segment
- If it is only a control segment, it consumes only one sequence number.

2. The server TCP, after receiving the FIN segment, sends the second segment, a FIN +ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction.

- This segment can also contain the last chunk of data from the server.
- If it does not carry data, it consumes only one sequence number.

3. The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server.

- This segment contains the acknowledgment number, which is 1 plus the sequence number received in the FIN segment from the server.
- This segment cannot carry data and consumes no sequence numbers.

NOTE

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

Flow Control

- TCP uses a sliding window to handle flow control.
- TCP sliding window is of variable size.
- The window is *opened, closed, or shrunk.*
  - Opening a window means moving the right wall to the right.
  - Closing the window means moving the left wall to the right.
  - Shrinking the window means moving the right wall to the left.

- The size of the window at one end is determined by the lesser of two values: *receiver window (rwnd) or congestion window (cwnd).*

Example 23.4

- ***What is the value of the receiver window (rwnd) for host A if the receiver, host B, has a buffer size of 5000 bytes and 1000 bytes of received and unprocessed data?***
- ***Solution***
- *The value of rwnd = 5000 − 1000 = 4000. Host B can receive only 4000 bytes of data before overflowing its buffer. Host B advertises this value in its next segment to A.*

Example 23.5

- ***What is the size of the window for host A if the value of rwnd is 3000 bytes and the value of cwnd is 3500 bytes?***
- ***Solution***
- *The size of the window is the smaller of rwnd and cwnd, which is 3000 bytes.*

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.

Error Control

- TCP is a reliable transport layer protocol. This means that TCP delivers the entire stream to the application program without error, and without any part lost or duplicated.
- TCP provides reliability using error control.
- Error control includes mechanisms for detecting corrupted segments, lost segments, out-of-order segments, and duplicated segments.
- Error control also includes a mechanism for correcting errors after they are detected.
- Error detection and correction in TCP is achieved through the use of three simple tools:
    - checksum
    - acknowledgment
    - time-out.

Error Control

Checksum:

- Each segment includes a checksum field which is used to check for a corrupted segment.
- If the segment is corrupted, it is discarded by the destination TCP and is considered as lost.
- TCP uses a 16-bit checksum that is mandatory in every segment

Acknowledgment:

- TCP uses acknowledgments to confirm the receipt of data segments.
- ACK segments do not consume sequence
- ACK segments are not acknowledged.

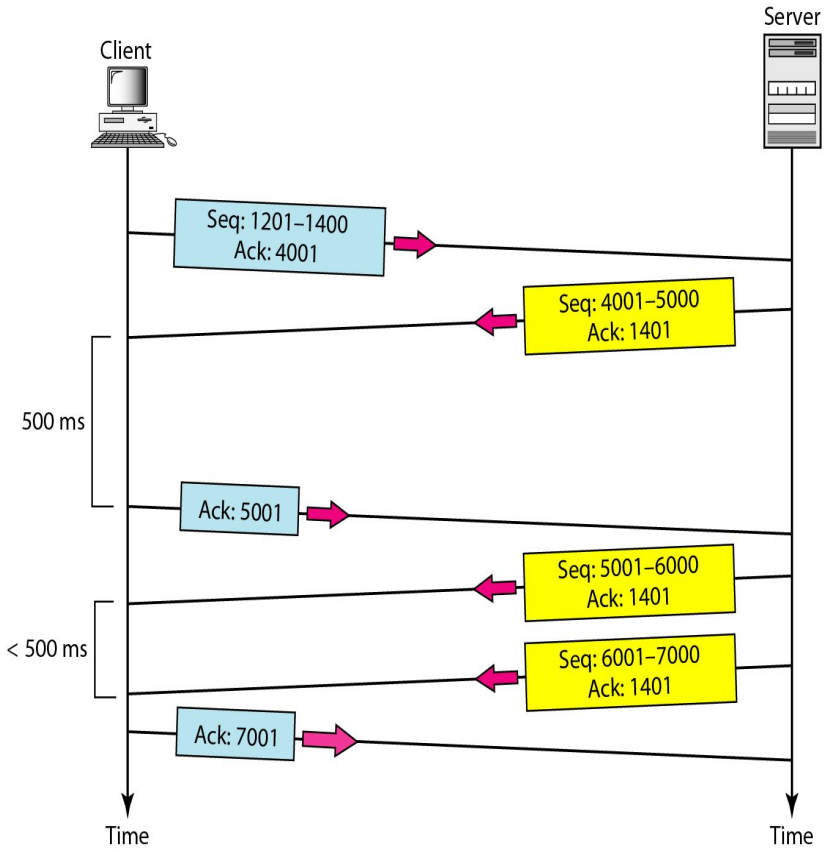
Error Control

Retransmission :

- When a segment is corrupted, lost, or delayed, it is retransmitted.
- the segments following that segment arrive out of order.
- Most TCP implementations today do not discard the out-of-order segments. They store them temporarily and flag them as out-of-order segments until the missing segment arrives through retransmission .
- The out-of-order segments are not delivered to the process. TCP guarantees that data are delivered to the process in order.
- In modern implementations, a segment is retransmitted on two occasions:
    1. **When a retransmission timer expires RTO or**
    2. **when the sender receives three duplicate ACKs.**

Note: No retransmission occurs for segments that do not consume sequence numbers. There is no transmission for an ACK segment. No retransmission timer is set for an ACK segment.

Normal operation



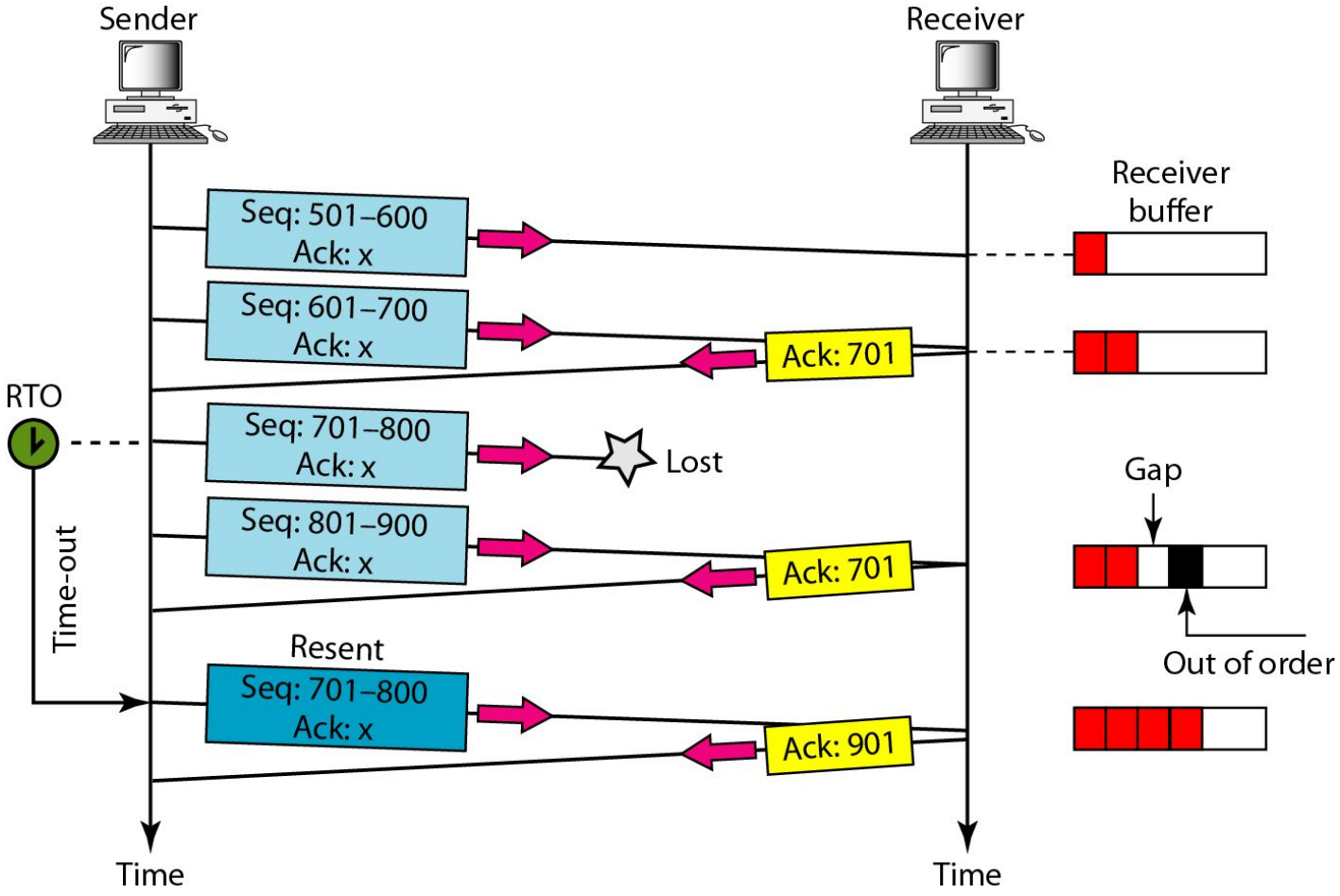
- The client TCP sends one segment; the server TCP sends three.
- When the client receives the first segment from the server, it does not have any more data to send; it sends only an ACK segment
- the acknowledgment needs to be delayed for 500 ms to see if any more segments arrive.

Lost segment: Retransmission after RTO

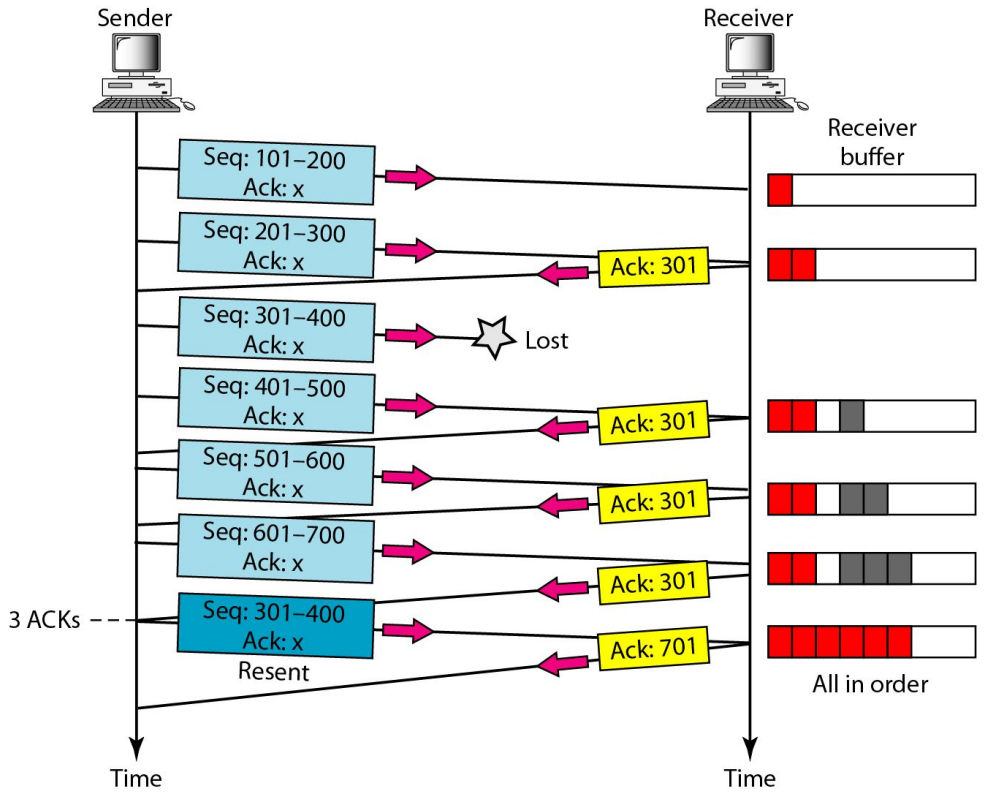
- Assume that data transfer is unidirectional: one site is sending, the other is receiving.
- In our scenario, the sender sends segments 1 and 2, which are acknowledged immediately by an ACK.
- Segment 3, however, is lost. The receiver receives segment 4, which is out of order.
- The receiver stores the data in the segment in its buffer but leaves a gap to indicate that there is no continuity in the data.
- The receiver immediately sends an acknowledgment to the sender, displaying the next byte it expects.
- The receiver stores bytes 801 to 900, but never delivers these bytes to the application until the gap is filled.

Note: The receiver TCP delivers only ordered data to the process.

Lost segment: Retransmission after RTO



Lost segment: Fast retransmission



Retransmission after three duplicated Ack segments

The scenario is the same as the RTO except that the RTO has a higher value

Lost segment: Fast retransmission

- When the receiver receives the fourth, fifth, and sixth segments, it triggers an acknowledgment.
- The sender receives four acknowledgments with the same value (three duplicates).
- Although the timer for segment 3 has not matured yet, the fast transmission requires that segment 3, the segment that is expected by all these acknowledgments, be resent immediately.

- Note that only one segment is retransmitted although four segments are not acknowledged. When the sender receives the retransmitted ACK, it knows that the four segments are safe and sound because acknowledgment is cumulative.

Book Reference

- Data Communications and Networking 5th Edition- Behrouz A. Forouzan