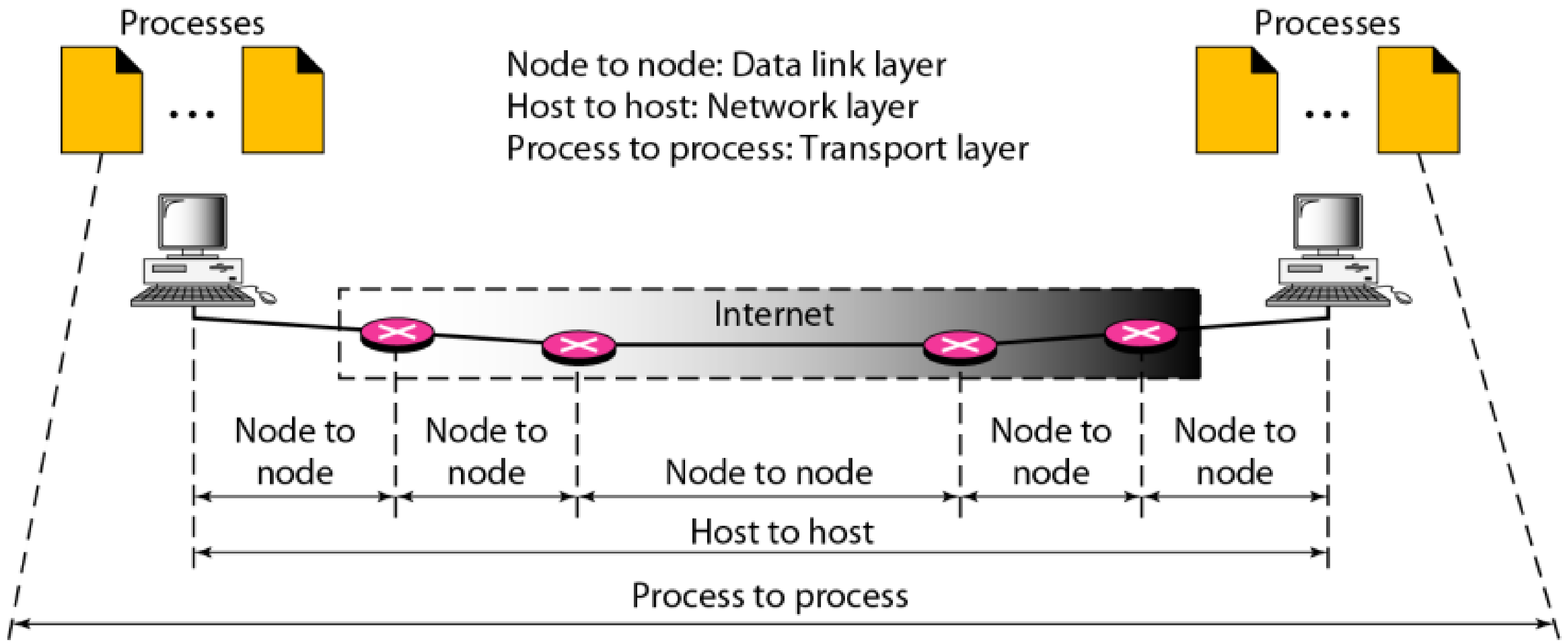
Unit 5: Transport Layer

Part A: Introduction

- The transport layer is an end-to-end layer – this means that nodes within the subnet do not participate in transport layer protocols – only the end hosts.
- As with other layers, transport layer protocols send data as a sequence of packets (segments).
- The transport layer is responsible for process-to process delivery—the delivery of a packet, part of a message, from one process to another.

Introduction



Introduction

- The **network layer** provides communication between two hosts.
- The **transport layer** provides communication between two ***processes*** running on different hosts.
- A process is an instance of a program that is running on a host.
- Example: There may be multiple processes communicating between two hosts – for example, there could be a FTP session and a Telnet session between the same two hosts.

Transport Layer Services

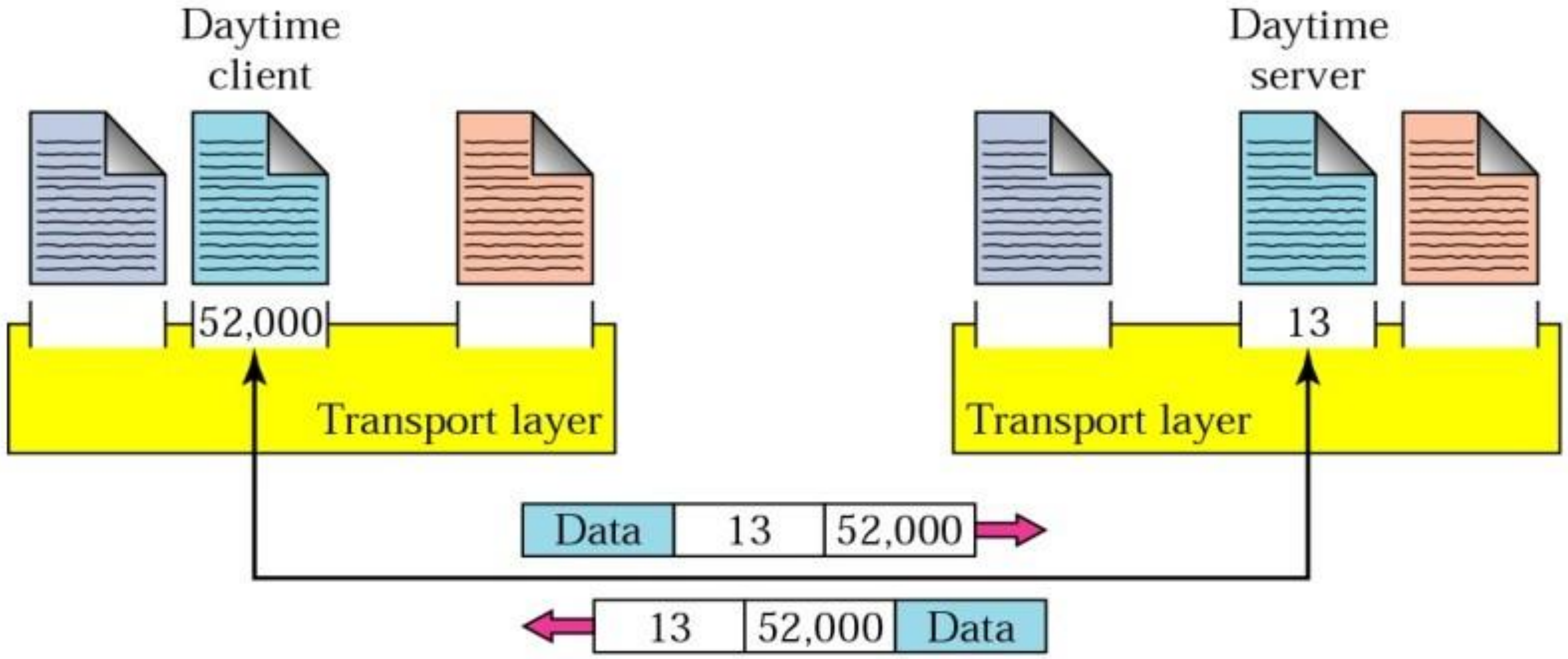
- Process to Process delivery
- Multiplexing/Demultiplexing
- Connectionless/Connection-oriented service
- Reliability

Process to Process delivery: Port Addresses

Client/Server Paradigm: Addressing

- A process on the local host, called a client, needs services from a process usually on the remote host, called a server.
- At the transport layer, we need a transport layer address, called 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. The client program defines itself with a port number, chosen randomly by the transport layer software running on the client host. This is the ephemeral port number.
- The Internet has decided to use universal port numbers for servers; these are called well-known port numbers.

Port Addresses

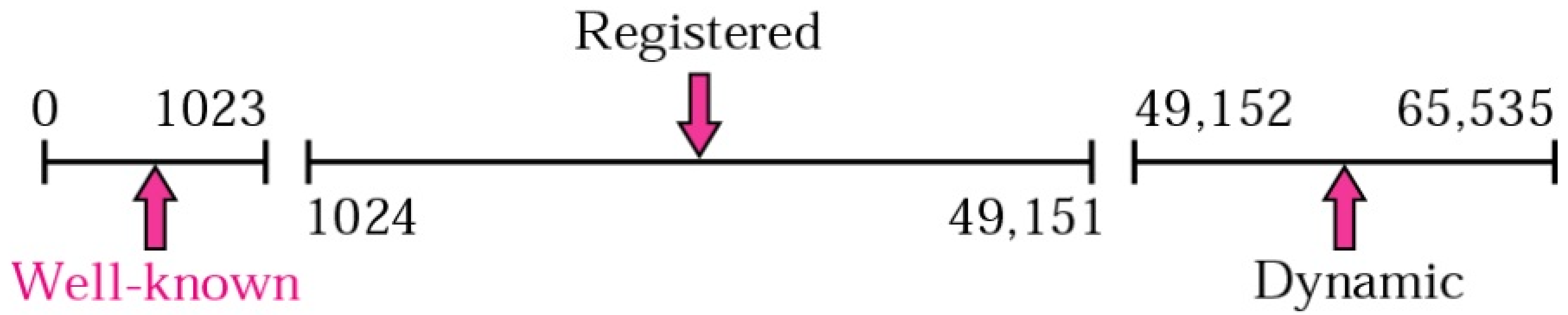


IANA Ranges: Port Addresses

IANA Ranges :The IANA (Internet Assigned Number Authority) has divided the port numbers into three ranges:

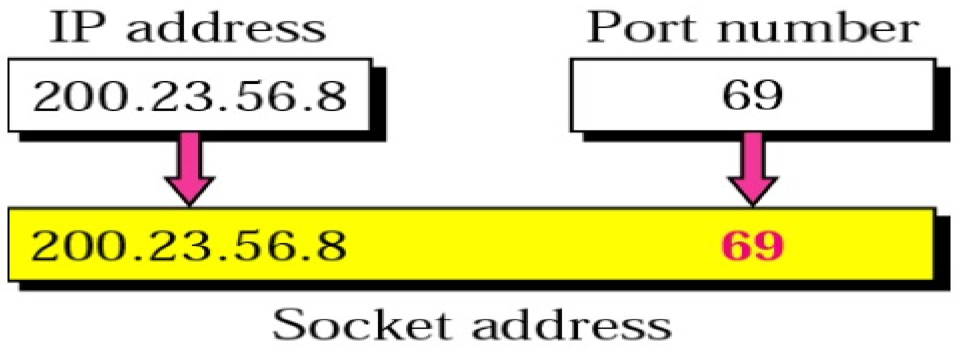
- **Well-known ports**. The ports ranging from 0 to 1023 are assigned and controlled by lANA. These are the well-known ports.
- **Registered ports**. The ports ranging from 1024 to 49,151 are not assigned or controlled by IANA. They can only be registered with IANA to prevent duplication.
- **Dynamic 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.

IANA Ranges: Port Addresses



Process to Process delivery: 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.



Multiplexing and Demultiplexing

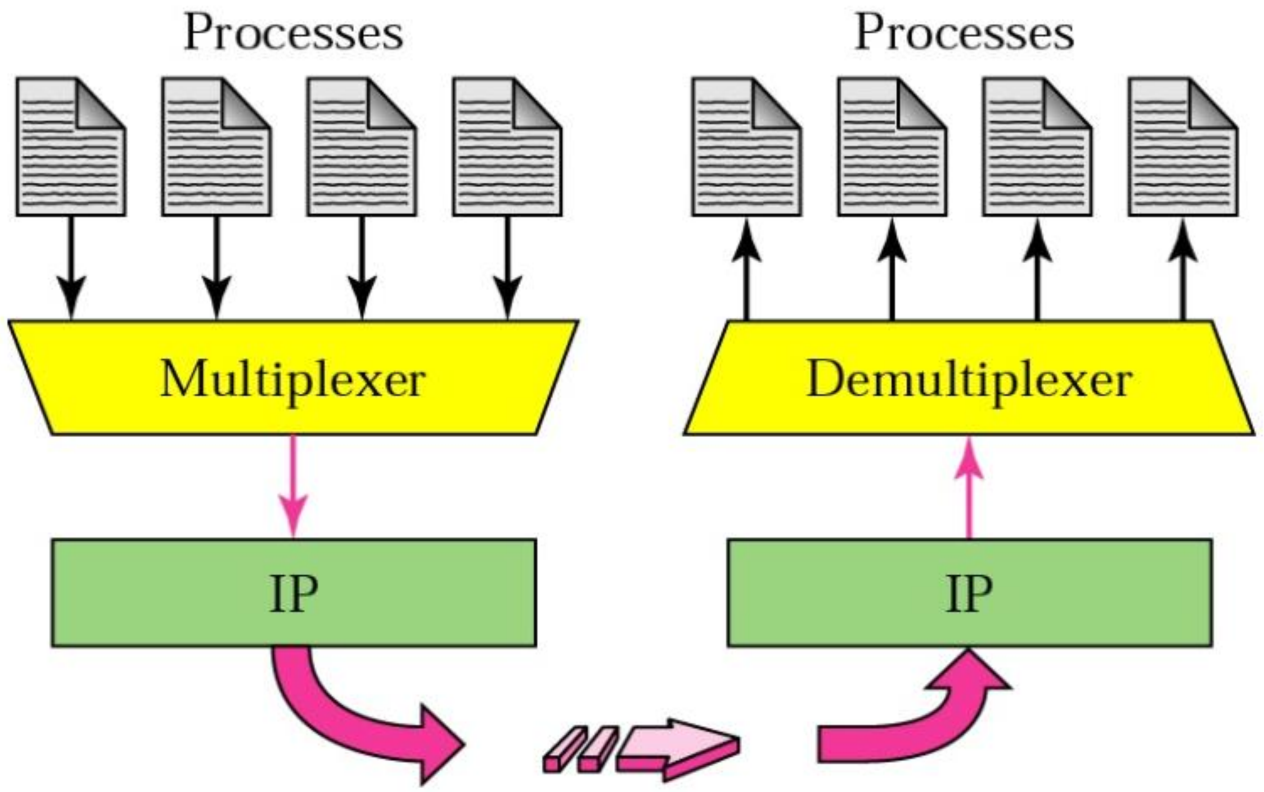
Multiplexing:

- At the sender site, there may be several processes that need to send packets. However, there is only one transport layer protocol at any time. This is a many-to-one relationship and requires multiplexing.
- The protocol accepts messages from different processes, differentiated by their assigned port numbers. After adding the header, the transport layer passes the packet to the network layer.

Demultiplexing:

- At the receiver site, the relationship is one-to-many and requires demultiplexing.
- The transport layer receives datagrams from the network layer. After error checking and dropping of the header, the transport layer delivers each message to the appropriate process based on the port number.

Multiplexing and Demultiplexing



Connectionless Versus Connection-Oriented Service

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 either.
- Example: UDP.

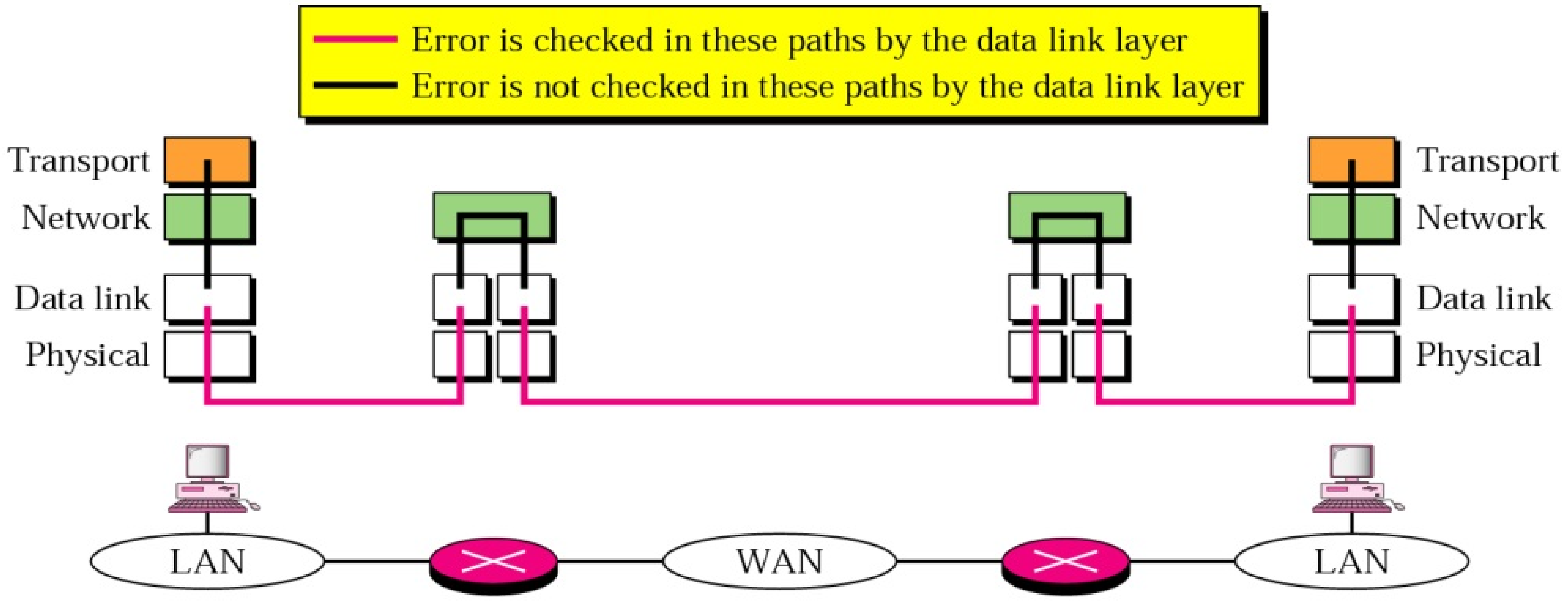
Connection-Oriented Service:

- 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.
- Example: TCP and SCTP

Reliable Versus 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.
- UDP is connectionless and unreliable; TCP and SCTP are connection-oriented and reliable.
- If the data link layer is reliable and has flow and error control, do we need this at the transport layer, too? The answer is yes. Reliability at the data link layer is between two nodes; we need reliability between two ends.

Reliable Versus Unreliable


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

NETWORK LAYER PROTOCOLS:

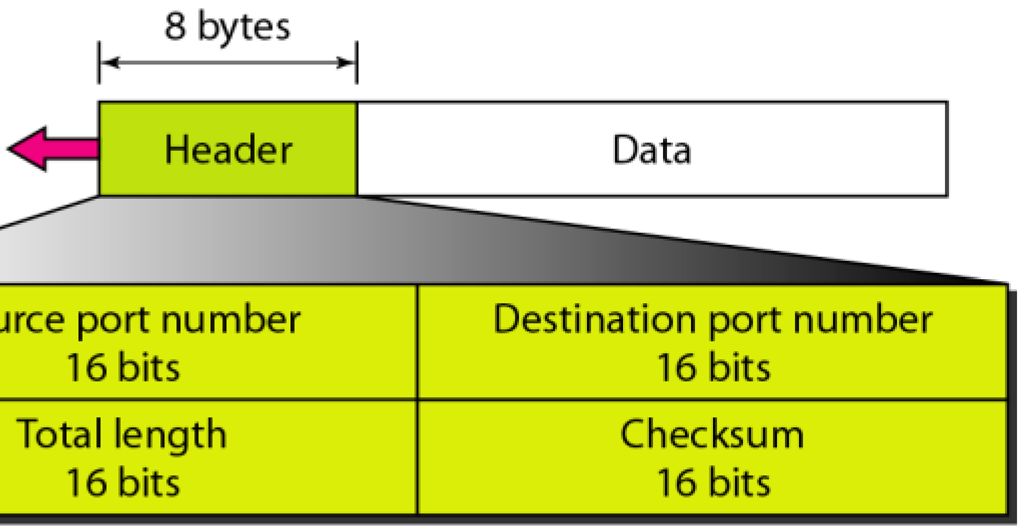
User Datagram Protocol (UDP)

- 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.
- UDP is a very simple protocol using a minimum overhead.
- If a process wants to send a small message and does not care much about reliability, it can use UDP.

Table 23.1 ***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/UDP Packet



User Datagram Format /UDP Packet

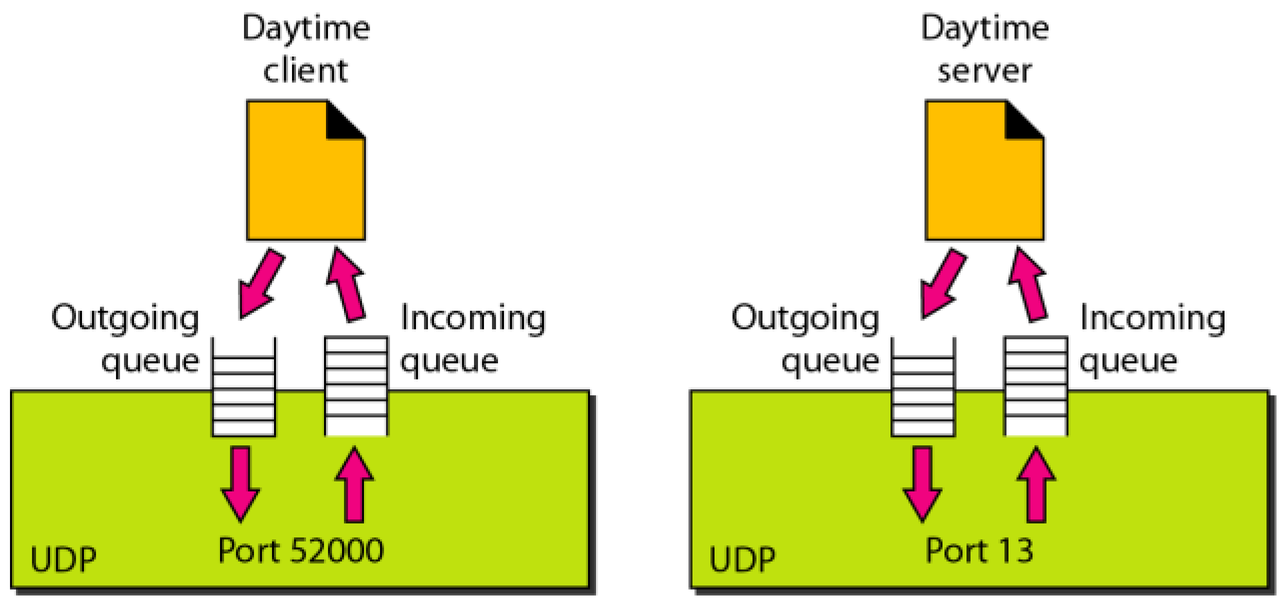
UDP packets, called user datagrams, have a fixed-size header of 8 bytes. The various fields are as follows:

- Source port number. This is the port number used by the process running on the source host. It is 16 bits long, which means that the port number can range from 0 to 65,535.
- Destination port number. This is the port number used by the process running on the destination host. It is also 16 bits long.
- Length. This is a 16-bit field that defines the total length of the user datagram, header plus data.
- Checksum. This field is used to detect errors over the entire user datagram (header plus data).

UDP Operation

- Connectionless Services: UDP provides a connectionless service. This means that each user datagram sent by UDP is an independent datagram.
- Flow and Error Control:
  - ➢ 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. This means that 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 silently discarded.
- Encapsulation and Decapsulation: To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages in an IP datagram
- Queuing: In UDP, queues are associated with ports

Queuing in UDP



Use of UDP

UDP is suitable for

- a process that requires simple request-response communication with little concern for flow and error control.
- a process with internal flow and error control mechanisms. For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control.
- multicasting.
- management processes such as SNMP.
- route updating protocols such as Routing Information Protocol (RIP).

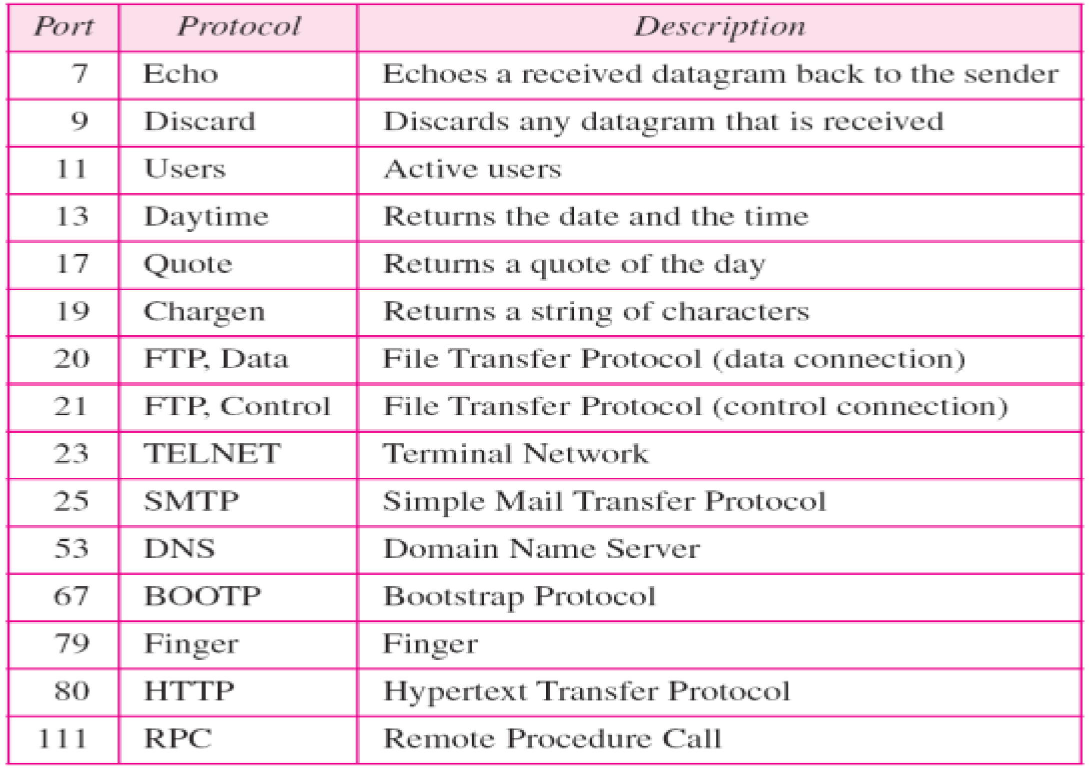
Example: Online Gaming, Video Streaming, Voice-over-IP (VoIP), etc

TCP (Transmission Control Protocol)

- TCP is called a connection-oriented, reliable transport protocol. It adds connection-oriented and reliability features to the services of IP.

TCP Services

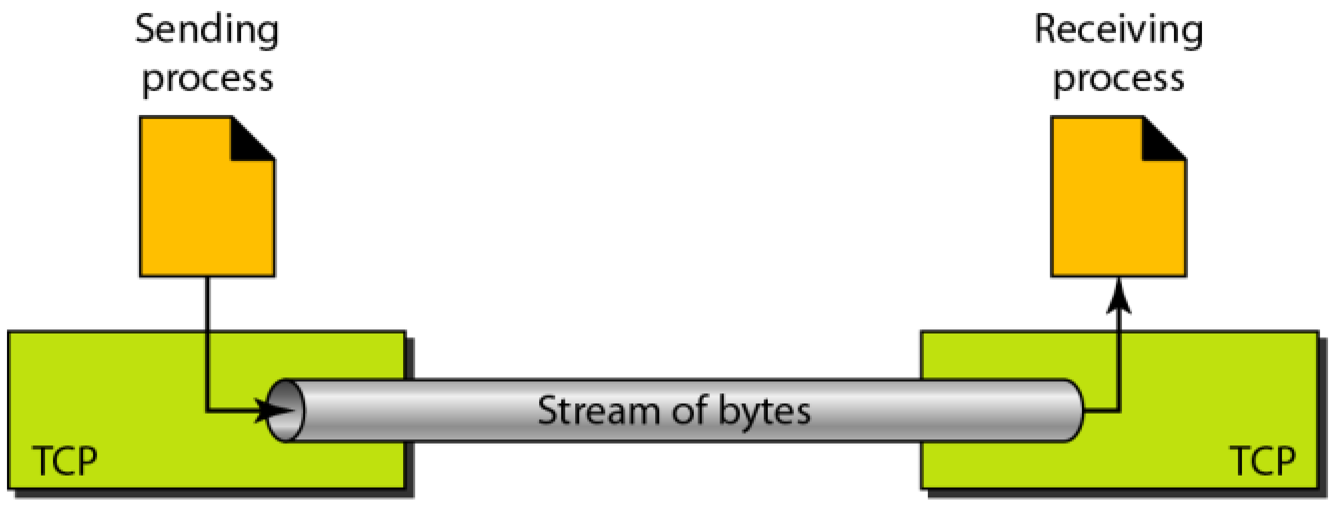
- Process-to-Process Communication: TCP provides process-to-process communication using port numbers. Some well-known port numbers used by TCP are



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

TCP Services

- 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, like an imaginary tube.

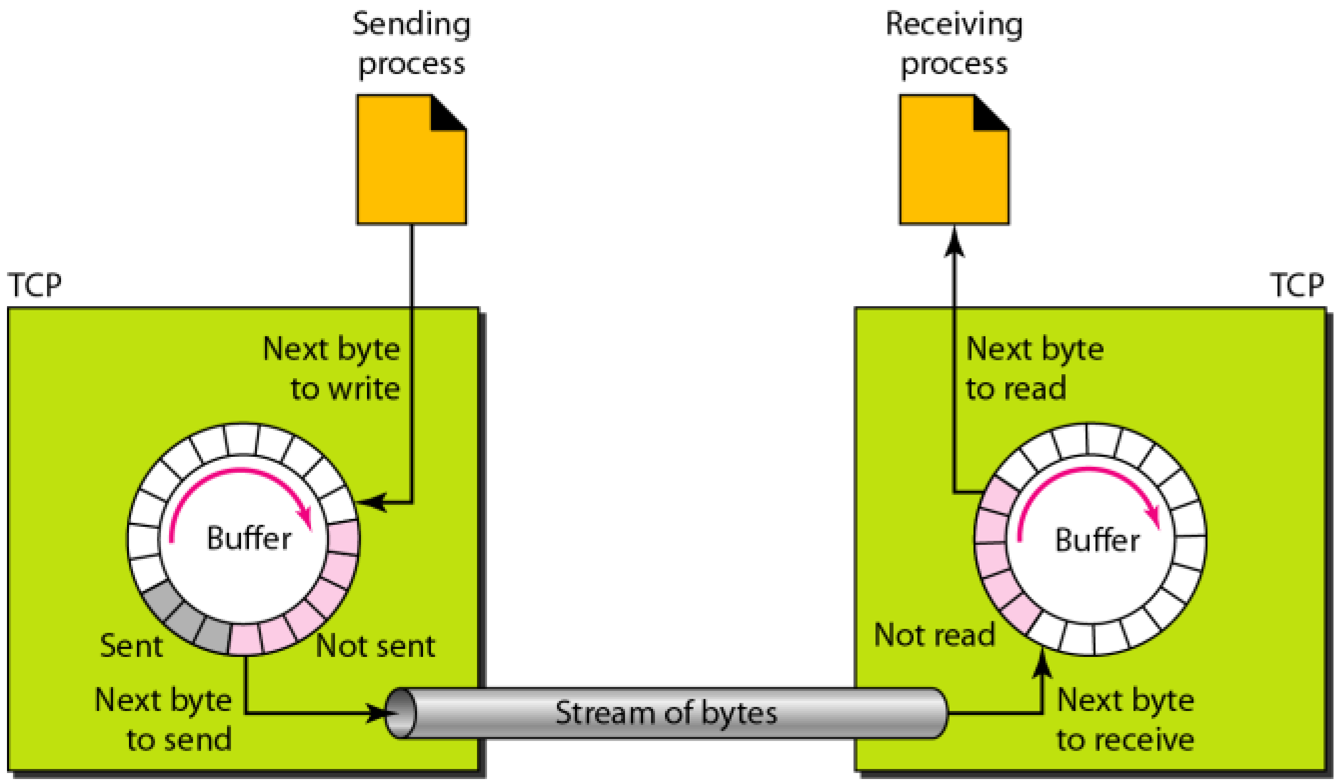


TCP Services

- Sending and Receiving Buffers:

  - Because the sending and the receiving processes may not write or read data at the same speed, TCP needs buffers for storage.
  - There are two buffers, the sending buffer and the receiving buffer, one for each direction.
  - One way to implement a buffer is to use a circular array of 1-byte locations as shown in Fig.

TCP Services



TCP Services

- Segments: 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.
- Full-Duplex Communication: TCP offers full-duplex service, in which data can flow in both directions at the same time.
- Connection-Oriented Service: TCP establishes a virtual connection between the two devices that need to communicate with each other.
- Reliable Service: TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data.

TCP Features

To provide the services discussed, TCP has several feature:

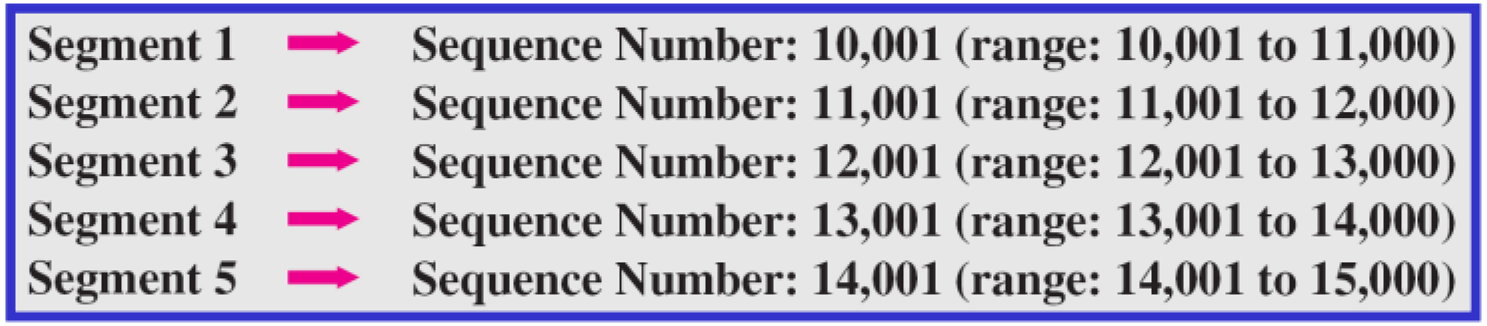
- Numbering System:
  - *Byte Number:*
    - TCP numbers all data bytes that are transmitted in a connection. Numbering is independent in each direction.
    - TCP generates a random number between 0 and $2^{32} - 1$ for the number of the first byte.
    - For example, if the random number happens to be 1057 and the total data to be sent are 6000 bytes, the bytes are numbered from 1057 to 7056.
  - *Sequence Number*:
    - 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

TCP Features

Example: Suppose a TCP connection is transferring a file of 5000 bytes. The first byte is numbered 1000l. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1000 bytes?

Sol:

The following shows the sequence number for each segment:



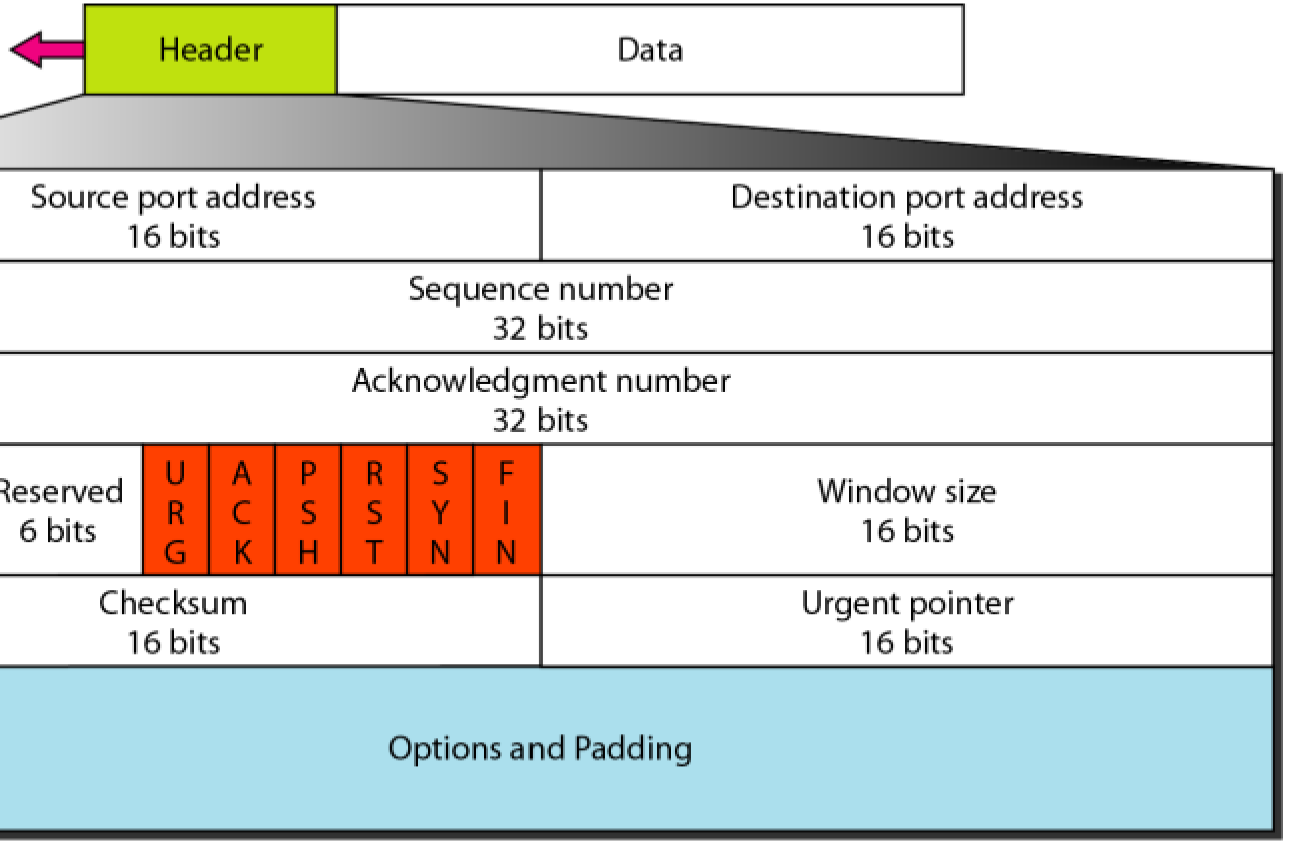
Segment 1	→	Sequence Number: 10,001 (range: 10,001 to 11,000)
Segment 2	→	Sequence Number: 11,001 (range: 11,001 to 12,000)
Segment 3	→	Sequence Number: 12,001 (range: 12,001 to 13,000)
Segment 4	→	Sequence Number: 13,001 (range: 13,001 to 14,000)
Segment 5	→	Sequence Number: 14,001 (range: 14,001 to 15,000)

TCP Features

- *Acknowledgement Number:*
  - The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive.
  - The acknowledgment number is cumulative, which means that the party takes the number of the last byte that it has received, safe and sound, adds 1 to it, and announces this sum as the acknowledgment number.
  - Example: A party announces 5643 as an acknowledgment number, it has received all bytes from the beginning up to 5642

- Flow Control: In TCP, the receiver of the data controls the amount of data that are to be sent by the sender. This is done to prevent the receiver from being overwhelmed with data.
- Error Control: To provide reliable service, TCP implements an error control mechanism.
- Congestion Control

TCP Segment Format



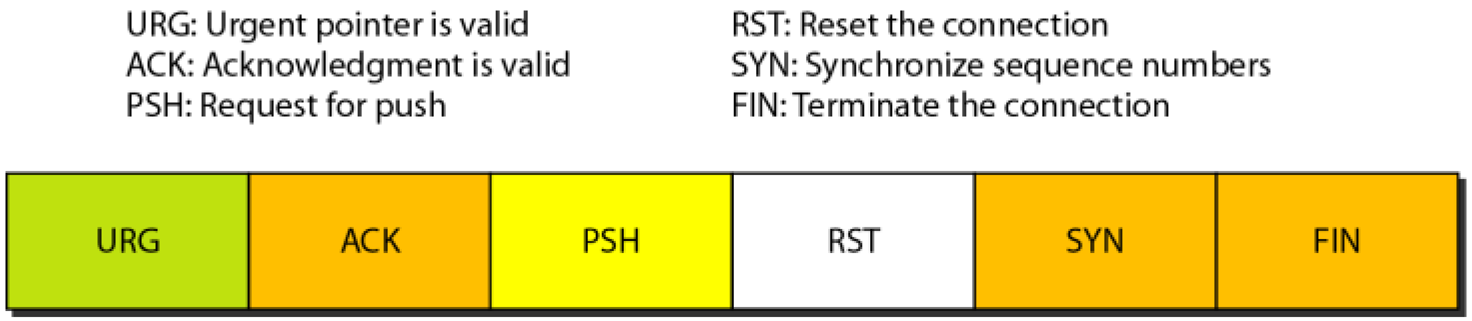
TCP Segment Format

The segment consists of a 20- to 60-byte header, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options.

- *Source port address:* This is a 16-bit field that 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 that defines the port number of the application program in the host that is receiving the segment.
- *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 byte number that the receiver of the segment is expecting to receive from the other party

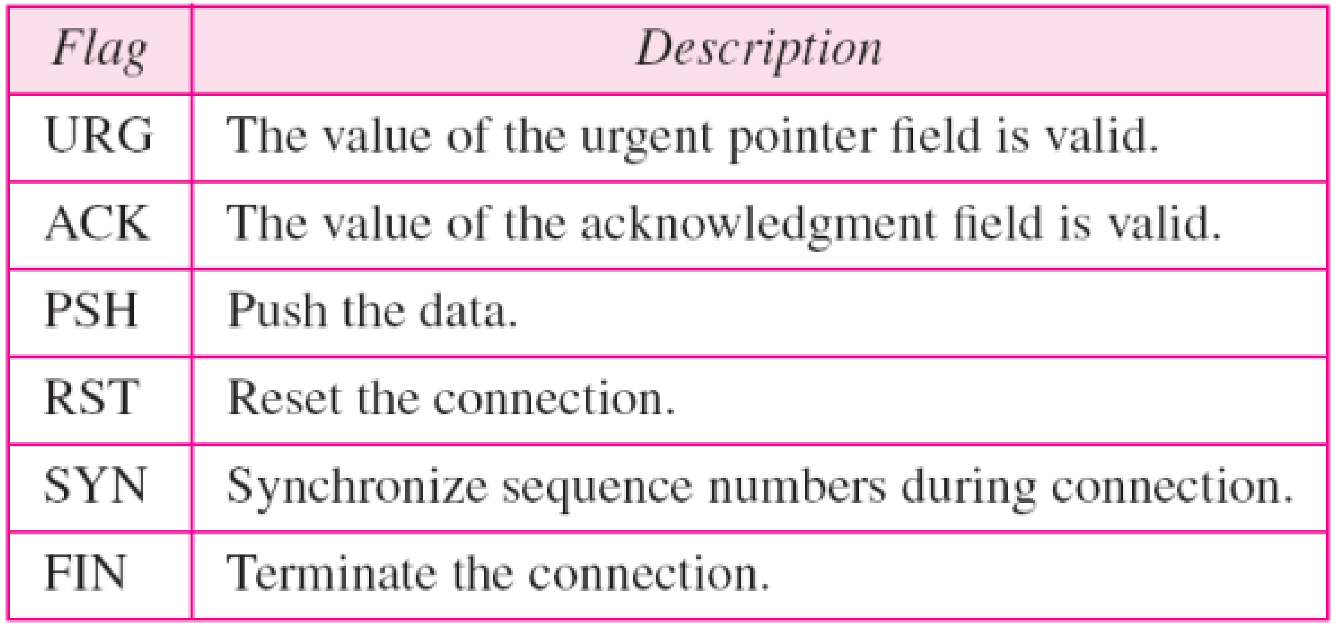
TCP Segment Format

- *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).
- *Reserved:* This is a 6-bit field reserved for future use.
- *Control bits:* This field defines 6 different control bits or flags as shown in Figure. One or more of these bits can be set at a time.


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

TCP Segment Format

- *Control Bits*: These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP.



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.

TCP Segment Format

- *Window size:* This field defines the size of the window, in bytes, that the other party must maintain. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes.
- *Checksum:* This 16-bit field contains the checksum.
- *Urgent pointer:* This l6-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data.
- *Options:* There can be up to 40 bytes of optional information in the TCP header.

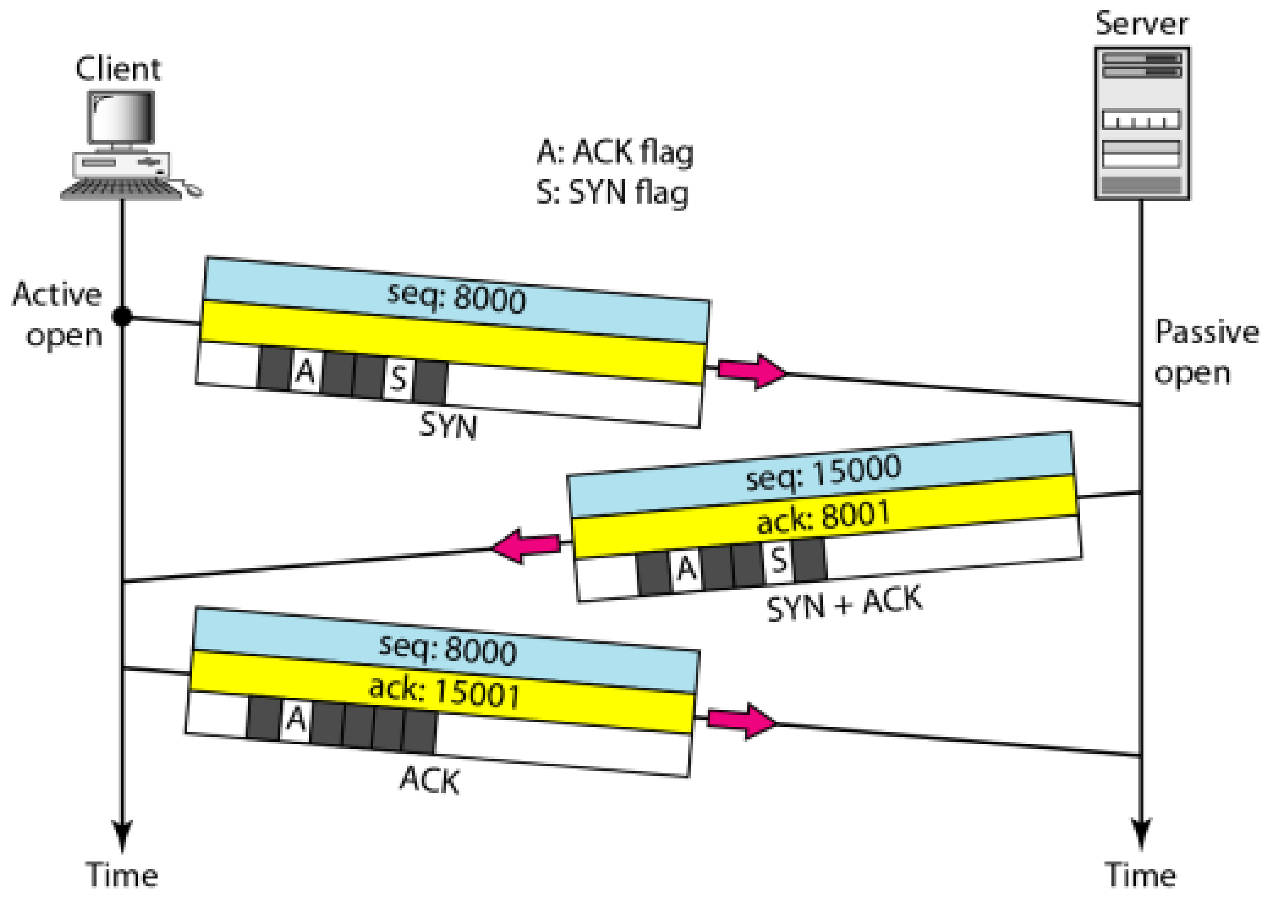
TCP Connection

- Using a single virtual pathway for the entire message facilitates the acknowledgment process as well as retransmission of damaged or lost frames.
- TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself. If a segment is lost or corrupted, it is retransmitted.
- If a segment arrives out of order, TCP holds it until the missing segments arrive; IP is unaware of this reordering and retransmission.
- In TCP, connection-oriented transmission requires three phases:
  - connection establishment,
  - data transfer, and
  - connection termination.

Connection Establishment: Three-Way Handshaking

- The connection establishment in TCP is called three-way handshaking.
- In our example, an application program, called the client, wants to make a connection with another application program, called the server, using TCP as the transport layer protocol.
- The server program tells its TCP that it is ready to accept a connection. This is called a request for a passive open.
- 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 as shown in Figure.

Connection Establishment: Three-Way Handshaking



Connection Establishment: Three-Way Handshaking

The three steps in this phase are as follows:

- 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.

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

- 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.

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

Connection Establishment: Three-Way Handshaking

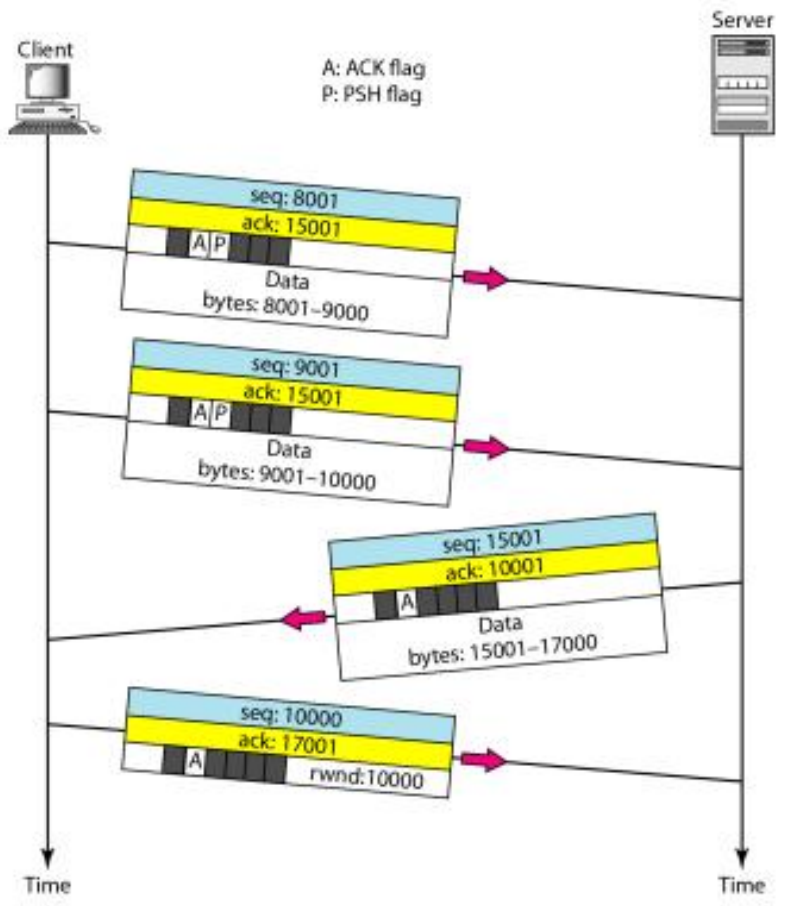
- The client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field.

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

Data Transfer

- After connection is established, bidirectional data transfer can take place.
- The client and server can both send data and acknowledgments.
- Data traveling in the same direction as an acknowledgment are carried on the same segment. The acknowledgment is piggybacked with the data.

Data Transfer



Data Transfer

- In this 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 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.
- The data segments sent by the client have the PSH (push) flag set so that the server TCP knows to deliver data to the server process as soon as they are received.

Data Transfer

- *Pushing Data*:
  - The application program at the sending site can request a push operation. This means that the sending TCP must not wait for the window to be filled. It must create a segment and send it immediately.
  - The sending TCP must set the push bit (PSH) to let the receiving TCP know that the segment includes data that must be delivered to the receiving application program as soon as possible and not to wait for more data to come.
- *Urgent Data:*
  - TCP is a stream-oriented protocol. Each byte of data has a position in the stream. However, on occasion an application program needs to send urgent bytes.
  - The solution is to send a segment with the URG bit set. The sending application program tells the sending TCP that the piece of data is urgent.
  - The urgent pointer field in the header defines the end of the urgent data and the start of normal data.

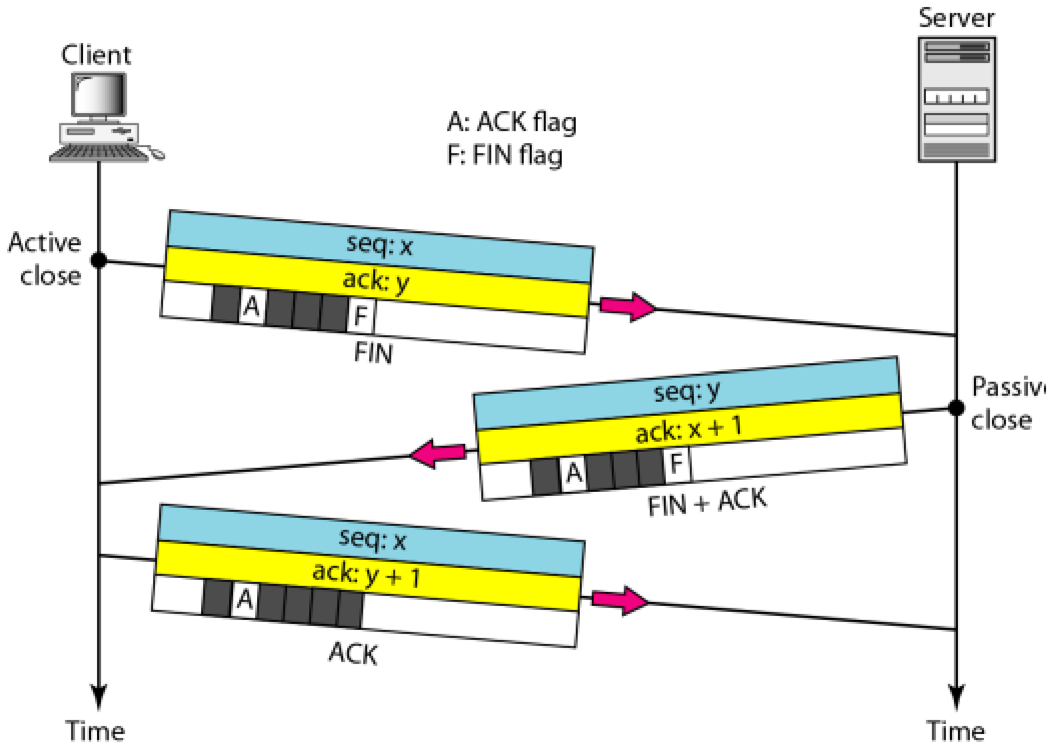
Connection Termination

- Any of the two parties involved in exchanging data (client or server) can close the connection, although it is usually initiated by the client.
- Most implementations today allow two options for connection termination:
  - three-way handshaking and
  - four-way handshaking with a half-close option.

Connection Termination: Three-Way Handshaking:

- In a normal situation, 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.
  - The FIN segment consumes one sequence number if it does not carry data.
- The server TCP, after receiving the FIN segment, informs its process of the situation and 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.
  - The FIN +ACK segment consumes one sequence number if it does not carry data.
- The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server.
  - This segment cannot carry data and consumes no sequence numbers.

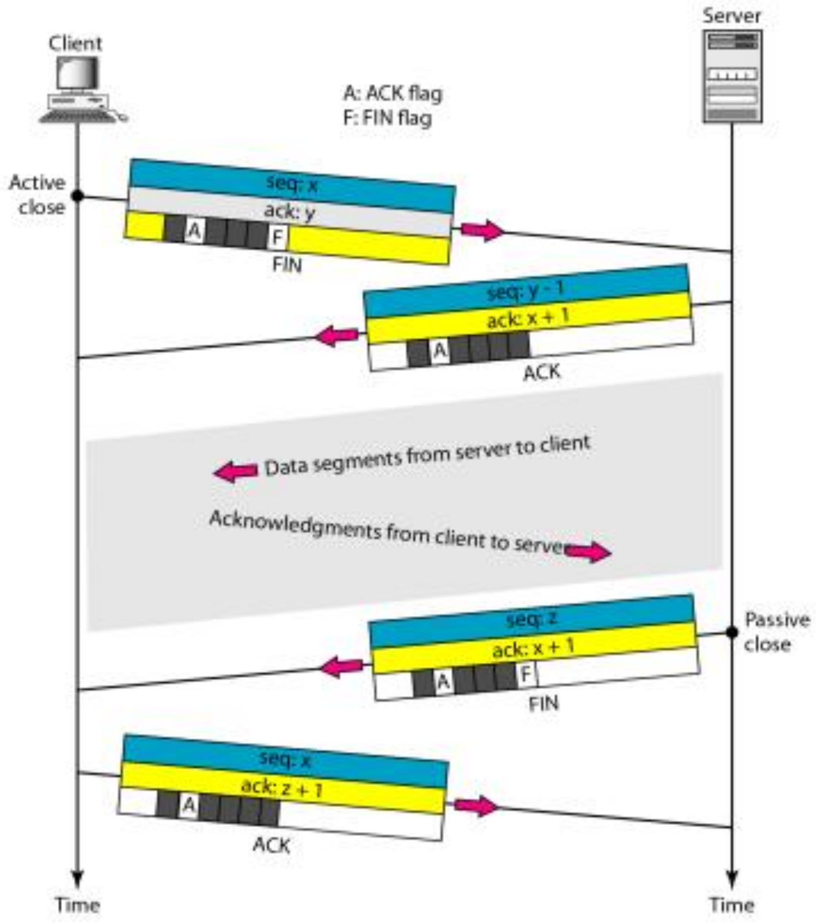
Connection Termination: Three-Way Handshaking



Connection Termination: 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.
- Figure shows an example of a half-close. The client half-closes the connection by sending a FIN segment.
- The server accepts the half-close by sending the ACK segment.
- The data transfer from the client to the server stops. The server, however, can still send data.
- When the server has sent all the processed data, it sends a FIN segment, which is acknowledged by an ACK from the client.

Connection Termination: Half-Close

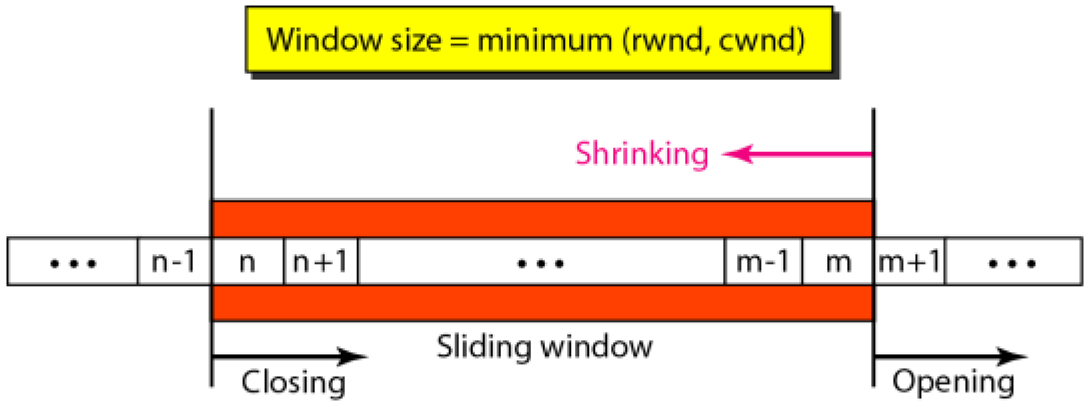


TCP Sliding Window: Flow Control

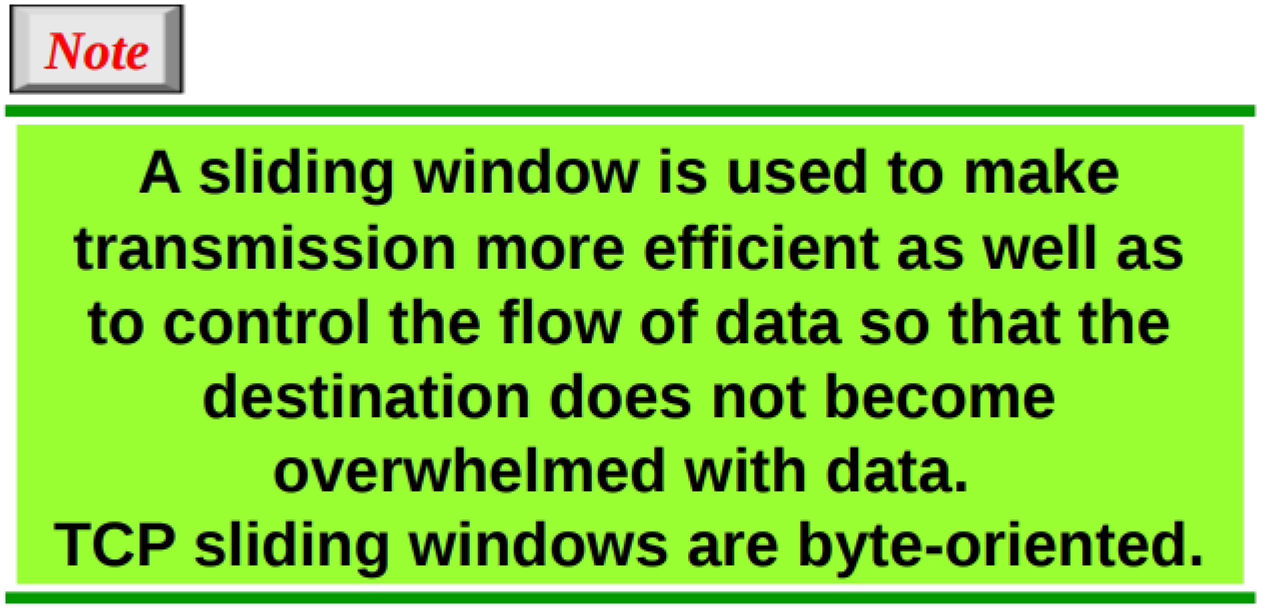
- TCP uses a sliding window to handle flow control.
- The sliding window protocol used by TCP is something between the Go-Back-N and Selective Repeat sliding window:
  - The sliding window protocol in TCP looks like the Go-Back-N protocol because it does not use NACKs
  - It looks like Selective Repeat because the receiver holds the out-of-order segments until the missing ones arrive.

TCP Sliding Window: Flow Control

- Figure shows the sliding window in TCP.
- The window spans a portion of the buffer containing bytes received from the process.
- The bytes inside the window are the bytes that can be in transit; they can be sent without worrying about acknowledgment.
- The imaginary window has two walls: one left and one right.



TCP Sliding Window: Flow Control



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.

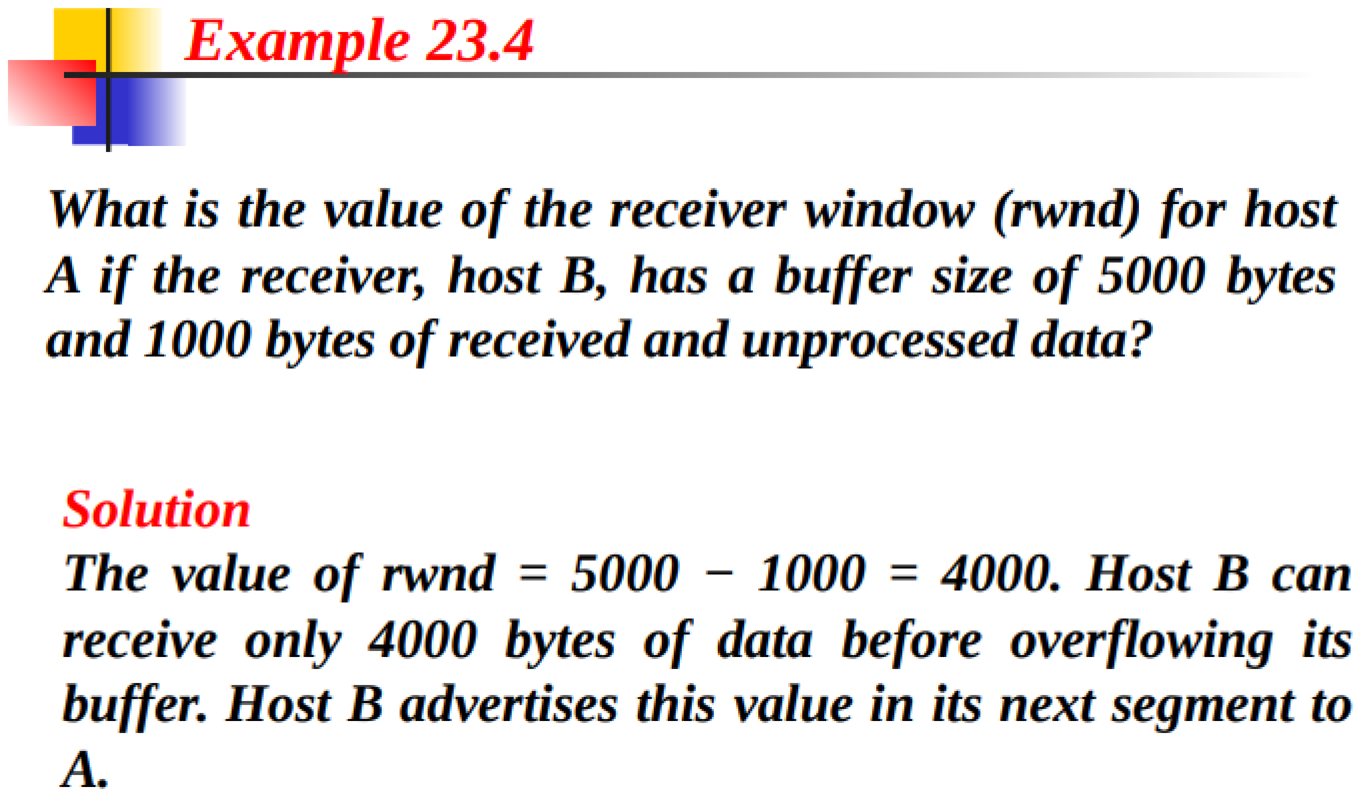
TCP Sliding Window: Flow Control

- The window is opened, closed, or shrunk. These three activities are in the control of the receiver.
- Opening a window means moving the right wall to the right. This allows more new bytes in the buffer that are eligible for sending.
- Closing the window means moving the left wall to the right. This means that some bytes have been acknowledged, and the sender need not worry about them anymore.
- Shrinking the window means moving the right wall to the left.

TCP Sliding Window: Flow Control

- The size of the window at one end is determined by the lesser of two values: receiver window (rwnd) or congestion window (cwnd).
- The receiver window is the number of bytes the other end can accept before its buffer overflows and data are discarded.
- The congestion window is a value determined by the network to avoid congestion.

TCP Sliding Window: Flow Control



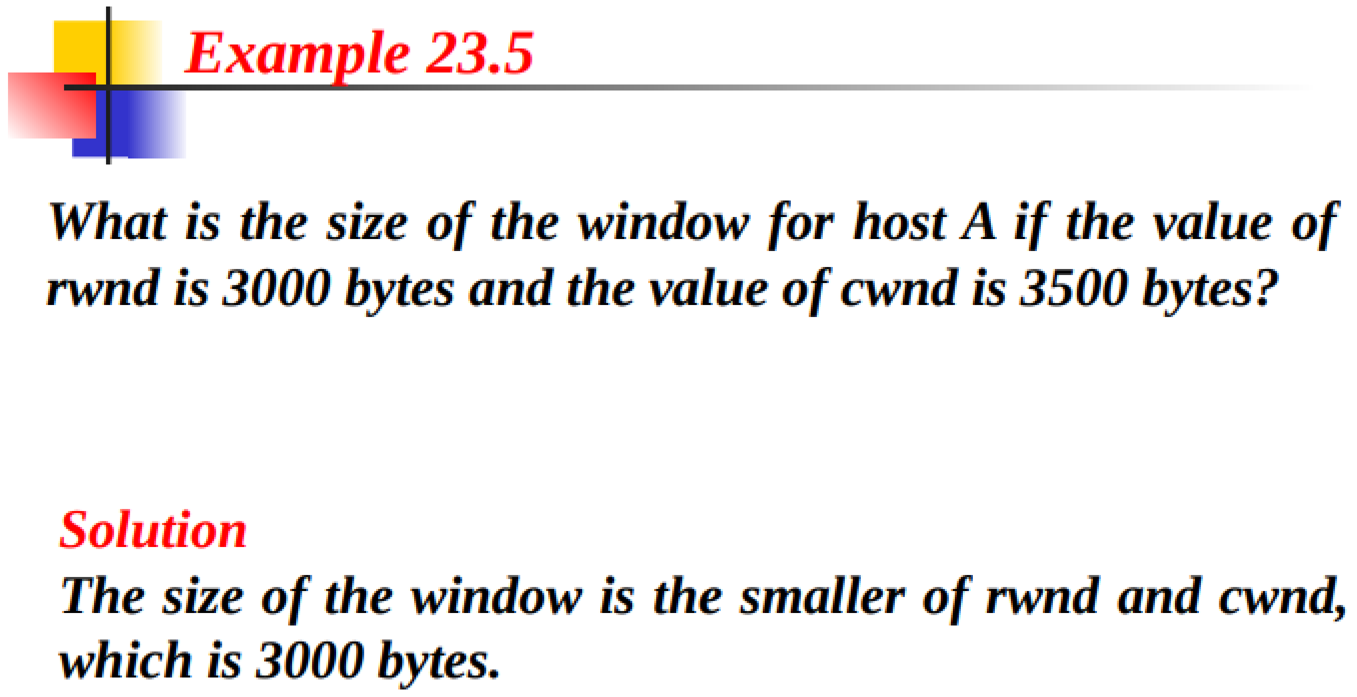
Example 23.4

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.

TCP Sliding Window: Flow Control



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

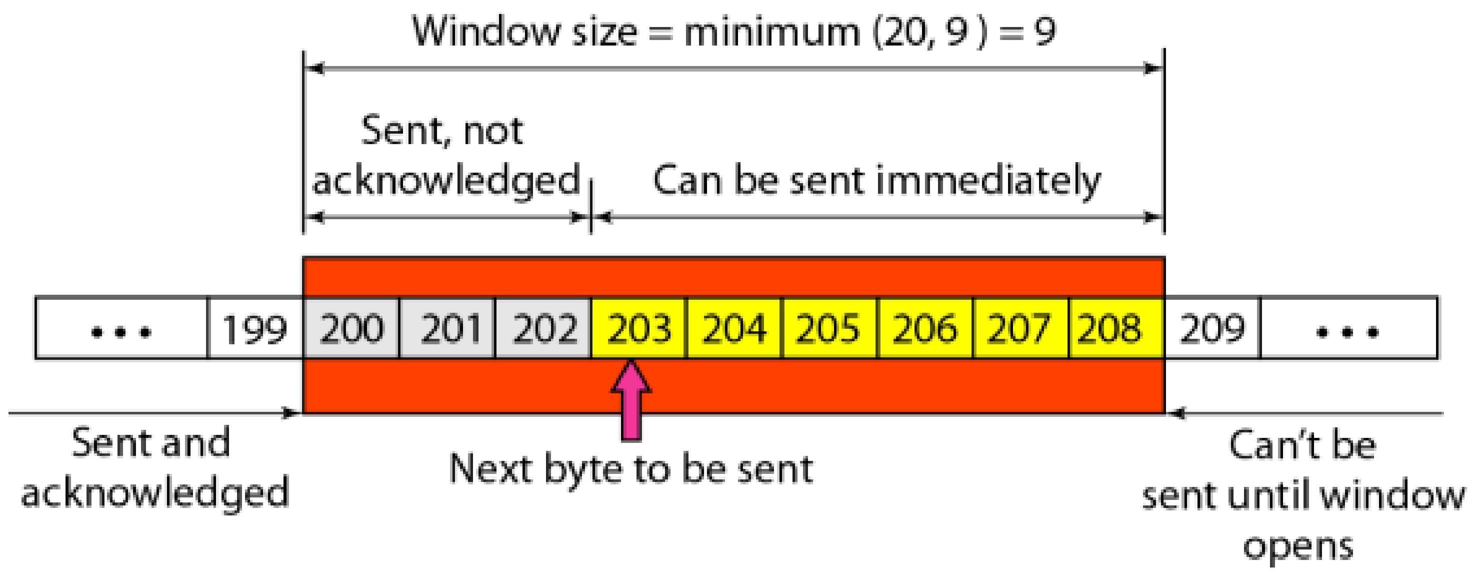
The size of the window is the smaller of rwnd and cwnd, which is 3000 bytes.

TCP Sliding Window: Flow Control

Example 23.6

Figure 23.23 shows an unrealistic example of a sliding window. 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.

TCP Sliding Window: Flow Control



TCP: 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, and time-out

TCP: 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.

- Acknowledgment: TCP uses acknowledgments to confirm the receipt of data segments.
  - ➢ Control segments that carry no data but consume a sequence number are also acknowledged.
  - ➢ ACK segments are never acknowledged.

TCP: Error Control

Retransmission:

- When a segment is corrupted, lost, or delayed, it is retransmitted.
  - 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.

TCP: Error Control

Retransmission after RTO:

- A recent implementation of TCP maintains one retransmission time-out (RTO) timer for all outstanding segments.
- The value of RTO is dynamic in TCP and is updated based on the round-trip time (RTT) of segments.
- If Ack is not received within RTO, the segment is retransmitted.

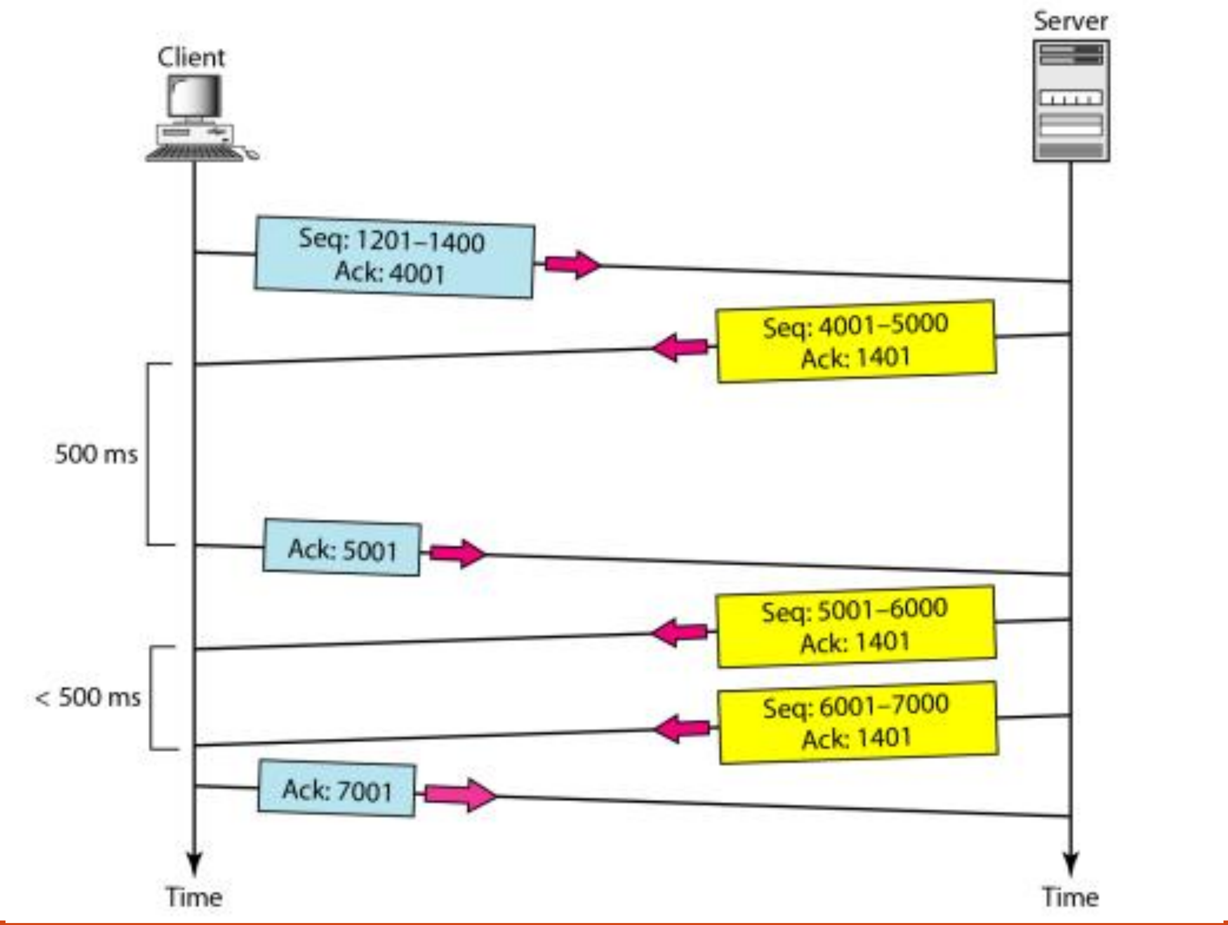
Retransmission after three duplicate ACK segments:

- 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.
- To alleviate this situation, most implementations today follow the three-duplicate-ACKs rule and retransmit the missing segment immediately

Normal Operation

- The first scenario shows bidirectional data transfer between two systems.
- 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.
- However, the acknowledgment needs to be delayed for 500 ms to see if any more segments arrive.
- When the timer matures, it triggers an acknowledgment. This is so because the client has no knowledge if other segments are coming; it cannot delay the acknowledgment forever.

Normal Operation

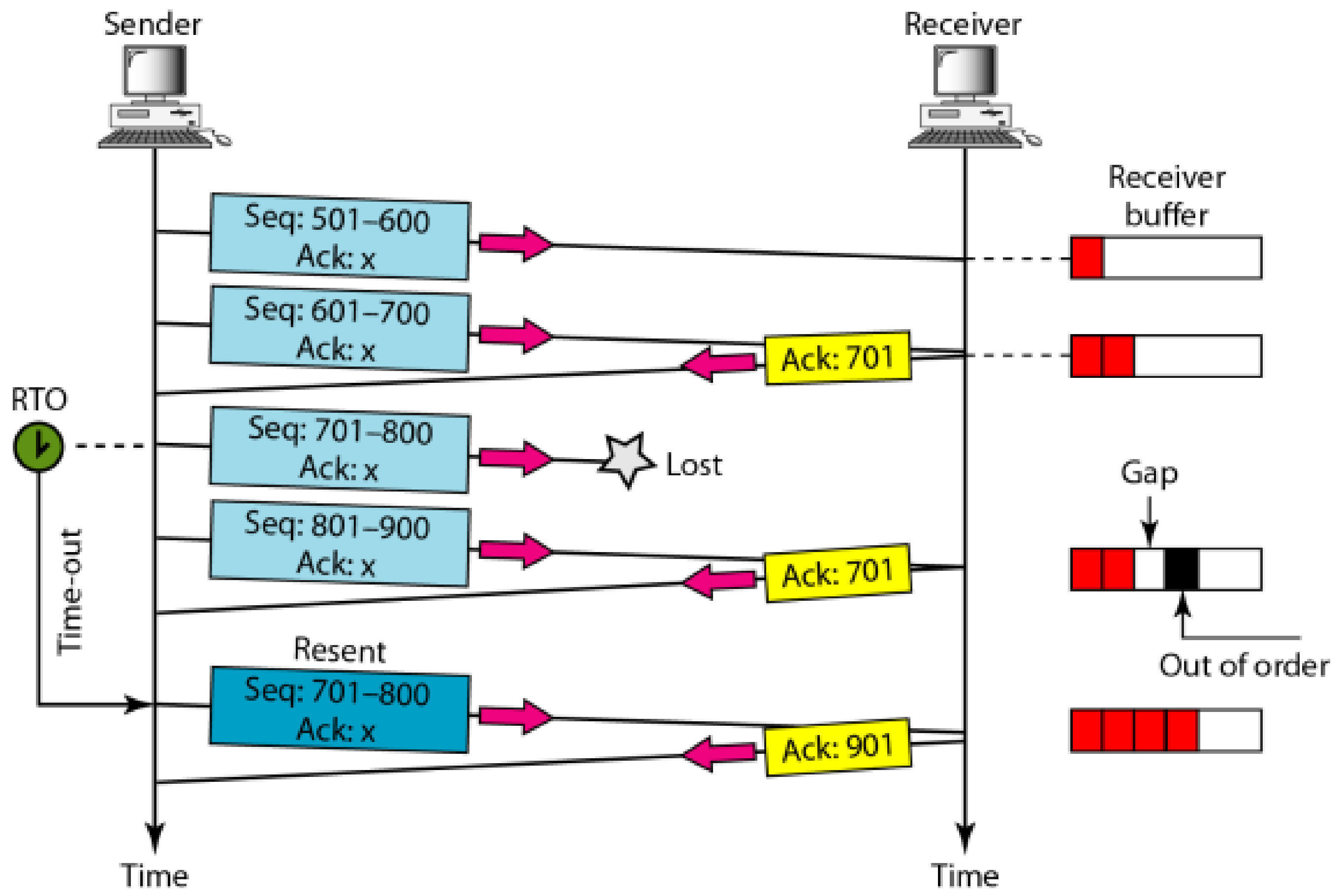


Lost Segment

- In this 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.
- Note that the receiver stores bytes 801 to 900, but never delivers these bytes to the application until the gap is filled.
- When the timer matures, the sending TCP resends segment 3, which arrives this time and is acknowledged properly

The receiver TCP delivers only ordered data to the process.

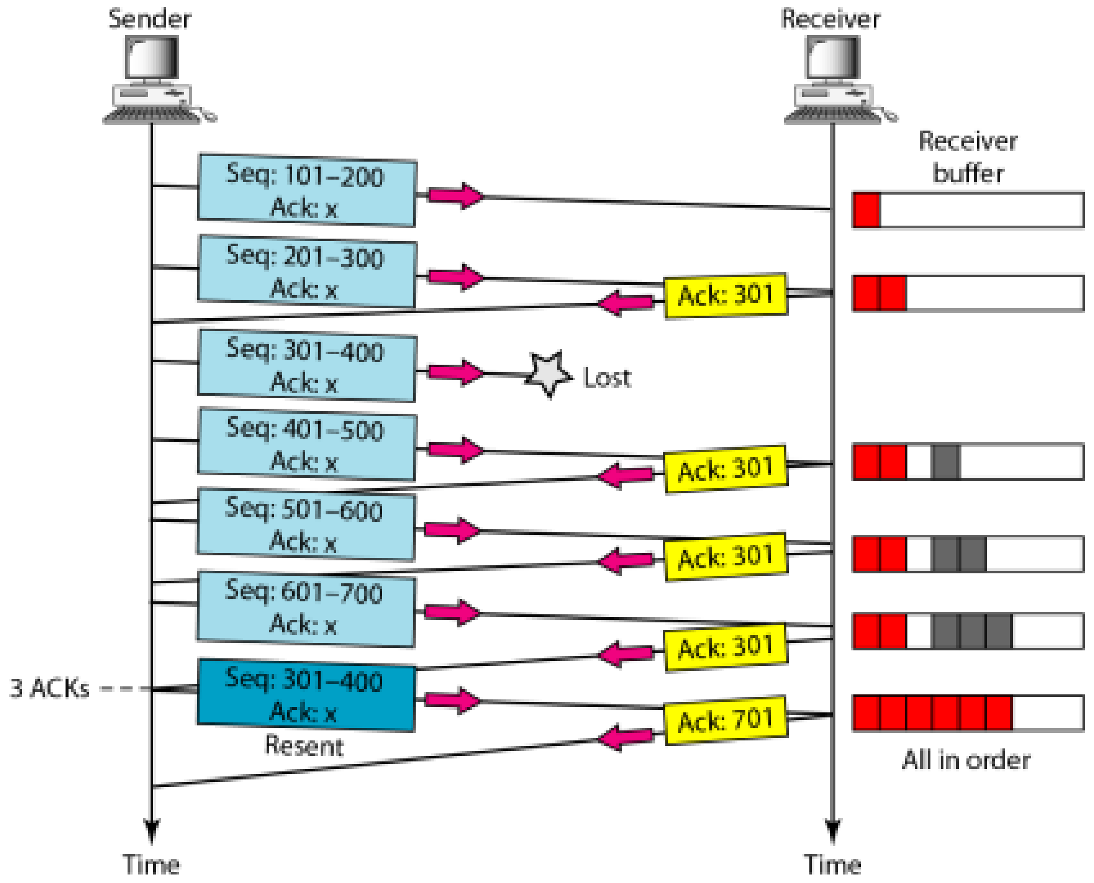
Lost Segment



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

Fast Retransmission


Sender
Receiver
Receiver buffer
Seq: 101–200 Ack: x
Seq: 201–300 Ack: x
Ack: 301
Seq: 301–400 Ack: x
Lost
Seq: 401–500 Ack: x
Ack: 301
Seq: 501–600 Ack: x
Ack: 301
Seq: 601–700 Ack: x
Ack: 301
3 ACKs
Seq: 301–400 Ack: x
Resent
Ack: 701
All in order
Time
Time