Unit IV: Transport Layer

LNMIIT
The LNM Institute of
Information Technology

Course: Computer Networks
Instructor: Saurabh Kumar
LNMIIT, Jaipur

March 22, 2021

Outline

- Transport Layer Service

Outline

- Transport Layer Service
- Connection less Transport: UDP

Outline

- Transport Layer Service
- Connection less Transport: UDP
- Connection Oriented Transport: TCP

Outline

- Transport Layer Service
- Connection less Transport: UDP
- Connection Oriented Transport: TCP
- Congestion Control

Transport Layer Responsibilities

- Process-to-Process delivery

Transport Layer Responsibilities

- Process-to-Process delivery
- Client-Server paradigm

Transport Layer Responsibilities

- Process-to-Process delivery
- Client-Server paradigm
  - Local host
  - Local process

Transport Layer Responsibilities

- Process-to-Process delivery
- Client-Server paradigm
  - Local host
  - Local process
  - Remote host
  - Remote process

Transport Layer Responsibilities

- Process-to-Process delivery
- Client-Server paradigm
  - Local host
  - Local process
  - Remote host
  - Remote process
  - Addressing

Transport Layer Responsibilities

- Process-to-Process delivery
- Client-Server paradigm
  - Local host
  - Local process
  - Remote host
  - Remote process
  - Addressing
  - IANA Ranges: Well-known ports, registered ports, dynamic ports

Transport Layer Responsibilities

- Process-to-Process delivery
- Client-Server paradigm
  - Local host
  - Local process
  - Remote host
  - Remote process
  - Addressing
  - IANA Ranges: Well-known ports, registered ports, dynamic ports
  - Socket address

Transport Layer Responsibilities

- Process-to-Process delivery
- Client-Server paradigm
  - Local host
  - Local process
  - Remote host
  - Remote process
  - Addressing
  - IANA Ranges: Well-known ports, registered ports, dynamic ports
  - Socket address
  - Multiplexing and Demultiplexing of processes

Transport Layer Responsibilities

- Process-to-Process delivery
- Client-Server paradigm
  - Local host
  - Local process
  - Remote host
  - Remote process
  - Addressing
  - IANA Ranges: Well-known ports, registered ports, dynamic ports
  - Socket address
  - Multiplexing and Demultiplexing of processes
  - Connectionless versus Conection-oriented service

Transport Layer Responsibilities

- Process-to-Process delivery
- Client-Server paradigm
  - Local host
  - Local process
  - Remote host
  - Remote process
  - Addressing
  - IANA Ranges: Well-known ports, registered ports, dynamic ports
  - Socket address
  - Multiplexing and Demultiplexing of processes
  - Connectionless versus Conection-oriented service
  - Reliable versus Unreliable service

Transport Layer Responsibilities

- Process-to-Process delivery
- Client-Server paradigm
  - Local host
  - Local process
  - Remote host
  - Remote process
  - Addressing
  - IANA Ranges: Well-known ports, registered ports, dynamic ports
  - Socket address
  - Multiplexing and Demultiplexing of processes
  - Connectionless versus Conection-oriented service
  - Reliable versus Unreliable service

UDP

- The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol.
- It provides process-to-process communication instead of host-to-host communication.
- It performs very limited error checking.
- Advantage: UDP is a very simple protocol using a minimum of overhead.

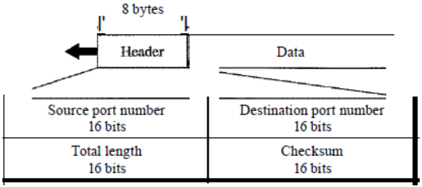
UDP

- Well-Known Ports for 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
123	NTP	Network Time Protocol
135	RPC	Remote Procedure Call
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

UDP

- User Datagram



8 bytes

Header

Data

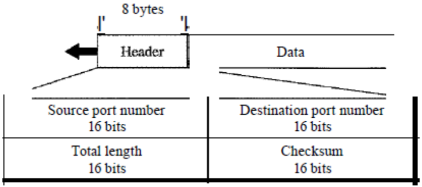
Source port number 16 bits	Destination port number 16 bits
Total length 16 bits	Checksum 16 bits

Figure: 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 range – ?).
  - If source host is client, port number is an ephemeral port number requested by the process and chosen by the UDP software running on the source host.
  - If the source host is the server, the port number is a well-known port number.

UDP

- User Datagram



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

Figure: User Datagram Format

- Destination Port Number
  - This is the port number used by the process running on the destination host.
  - It is also 16 bits long.
  - If the destination host is the server, the port number is a well-known port number.
  - If the destination host is the client, the port number is an ephemeral port number.

UDP

- User Datagram

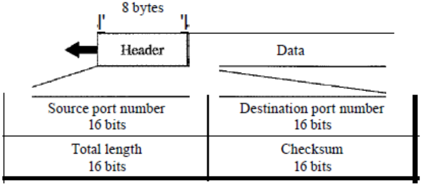


Figure: User Datagram Format

- Length
  - This is a 16-bit field that defines the total length of the user datagram (header plus data).
    UDP Length = IP Length – IP header's length
- Checksum: This field is used to detect errors over the entire user datagram (header plus data).

UDP

- UDP Operation
  - Connectionless Services
    - UDP provides a connectionless service.
    - User datagram sent by UDP is an independent datagram.
    - There is no relationship between the different user datagrams, even if they are coming from the same source process and going to the same destination program.
    - The user datagrams are not numbered.
    - There is no connection establishment and no connection termination.
    - Each user datagram can travel on a different path.
    - Process that uses UDP cannot send a stream of data to UDP and expect UDP to chop them into different related user datagrams.
    - Each request must be small enough to fit into one user datagram.
    - Only those processes sending short messages should use UDP.

UDP

- UDP Operation
  - Flow and Error Control
    - UDP is a very simple, unreliable transport protocol.
    - 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.
    - 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.
    - The lack of flow control and error control means that the process using UDP should provide these mechanisms.

UDP

- UDP Operation
  - Encapsulation and Decapsulation: To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages in an IP datagram.
  - Queuing

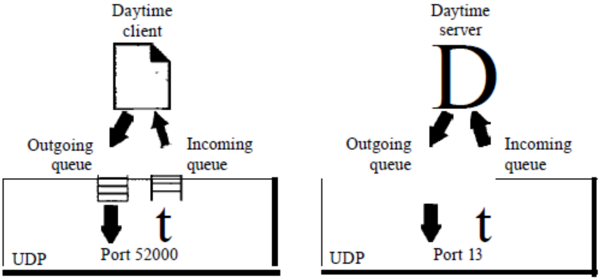


Figure: Queues in UDP

UDP

- Uses of UDP
  - UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process such as FTP that needs to send bulk data.

UDP

- Uses of UDP
  - UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process such as FTP that needs to send bulk data.
  - 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. It can easily use UDP.

UDP

- Uses of UDP
  - UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process such as FTP that needs to send bulk data.
  - 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. It can easily use UDP.
  - UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.

UDP

- Uses of UDP
  - UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process such as FTP that needs to send bulk data.
  - 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. It can easily use UDP.
  - 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

- Uses of UDP
  - UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process such as FTP that needs to send bulk data.
  - 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. It can easily use UDP.
  - 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).

TCP

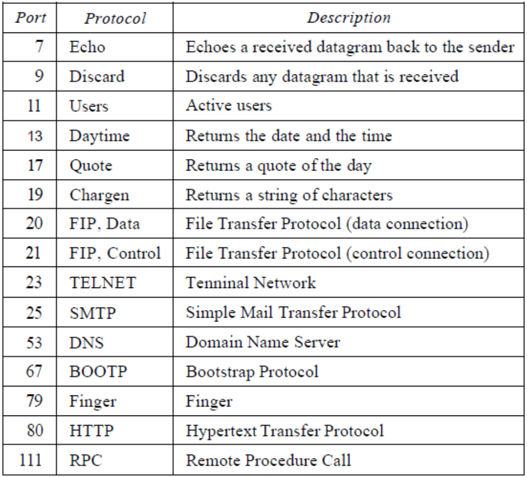
- TCP is called a connection-oriented, reliable transport protocol.
- It adds connection-oriented and reliability features to the services of IP.

TCP

- 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 delivery
  - Stream delivery service
  - Full duplex communication
  - Connection-oriented service
  - Reliable service

TCP

- Process-to-Process Delivery



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	FIP, Data	File Transfer Protocol (data connection)
21	FIP, Control	File Transfer Protocol (control connection)
23	TELNET	Tenninal 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

Figure: Well-known ports used by TCP

TCP

- Stream Delivery Service
    - TCP, unlike UDP, is a stream-oriented protocol.
    - 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.
    - The sending process produces (writes to) the stream of bytes, and the receiving process consumes (reads from) them.
    - Sending and Receiving buffers
    - Concept of Segments

TCP

- Stream Delivery Service
  - TCP, unlike UDP, is a stream-oriented protocol.
  - 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.
  - The sending process produces (writes to) the stream of bytes, and the receiving process consumes (reads from) them.
  - Sending and Receiving buffers
  - Concept of Segments
- 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.

TCP

- Connection-Oriented Service
  - 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.

TCP

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

TCP Features

- Numbering System
  - TCP has no field for a segment number value in the segment header.
  - There are two fields: sequence number and the acknowledgment number.

TCP Features

- Numbering System
  - TCP has no field for a segment number value in the segment header.
  - There are two fields: sequence number and the acknowledgment number.
  - Byte 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 2^{32} - 1 for the number of the first byte.

TCP Features

- Numbering System
  - TCP has no field for a segment number value in the segment header.
  - There are two fields: sequence number and the acknowledgment number.
  - Byte 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 2^{32} - 1 for the number of the first byte.
  - Sequence 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.

TCP Features

- Numbering System

  Example: 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?

TCP Features

- Numbering System

  Example: 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?

  Solution: The following shows the sequence number for each segment:

TCP Features

- Numbering System
  Example: 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?
  Solution: The following shows the sequence number for each segment:
  Segment 1 Sequence Number: 10,001 (range: 10,001 to 11,000)

TCP Features

- Numbering System
  Example: 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?
  Solution: 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)

TCP Features

- Numbering System

  Example: 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?

  Solution: 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)

TCP Features

- Numbering System
  Example: 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?
  Solution: 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)

TCP Features

- Numbering System

  Example: 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?

  Solution: 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

- Numbering System

  Example: 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?

  Solution: 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
    - 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 Features

- Flow Control
  - TCP, unlike UDP, provides flow control.
  - 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.
  - The numbering system allows TCP to use a byte-oriented flow control.

TCP Features

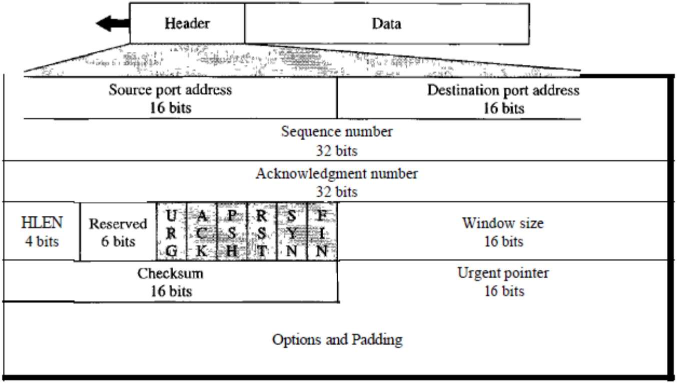
- Flow Control
  - TCP, unlike UDP, provides flow control.
  - The receiver of the data controls the amount of data that are to be sent by the sender.
  - This is done to prevent the receiver from being overwhelmed with data.
  - The numbering system allows TCP to use a byte-oriented flow control.
- Error Control
  - To provide reliable service, TCP implements an error control mechanism.
  - Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented.

TCP Features

- Flow Control
  - TCP, unlike UDP, provides flow control.
  - 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.
  - The numbering system allows TCP to use a byte-oriented flow control.
- Error Control
  - To provide reliable service, TCP implements an error control mechanism.
  - Although error control considers a segment as the unit of data for error detection (loss or corrupted segments), error control is byte-oriented.
- Congestion Control
  - TCP, unlike UDP, takes into account congestion in the network.
  - The amount of data sent by a sender is not only controlled by the receiver (flow control), but is also determined by the level of congestion in the network.

Segment

- A packet in TCP is known as *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

Figure: TCP Segment Format

Segment

- 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:
  - A 16-bit field that defines the port number of the application program in the host that is sending the segment.
  - It serves the same purpose as the source port address in the UDP header.

Segment

- 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:
  - A 16-bit field that defines the port number of the application program in the host that is sending the segment.
  - It serves the same purpose as the source port address in the UDP header.
- Destination port address:
  - A 16-bit field that defines the port number of the application program in the host that is receiving the segment.
  - It serves the same purpose as the destination port address in the UDP header.

Segment

- Sequence number: (why required – ?)

Segment

- Sequence number: (why required – ?)– stream oriented protocol

Segment

- Sequence number: (why required – ?)– stream oriented protocol
  - A 32-bit field defines the number assigned to the first byte of data contained in this segment.
  - 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.

Segment

- Sequence number: (why required – ?)– stream oriented protocol
  - A 32-bit field defines the number assigned to the first byte of data contained in this segment.
  - 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.

Segment

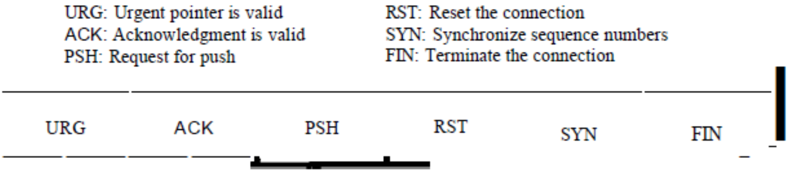
- Header Length:
  - This 4-bit field indicates the number of 4-byte words in the TCP header.
  - The length of the header can be between 20 and 60 bytes.
  - Therefore, the value of this field can be between 5 (5 x 4 =20) and 15 (15 x 4 =60).

Segment

- 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.

Segment

- 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: This field defines 6 different control bits or flags as shown in figure below.



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

Figure: Control field

Segment

- Window Size:
  - This field 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.

Segment

- Window Size:
  - This field 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.
- Checksum:
  - This 16-bit field contains the checksum.
  - The inclusion of the checksum in the UDP datagram is optional, whereas the inclusion of the checksum for TCP is mandatory.

Segment

- Window Size:
  - This field 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.
- Checksum:
  - This 16-bit field contains the checksum.
  - The inclusion of the checksum in the UDP datagram is optional, whereas the inclusion of the checksum for TCP is mandatory.
- 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.

Segment

- Window Size:
  - This field 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.
- Checksum:
  - This 16-bit field contains the checksum.
  - The inclusion of the checksum in the UDP datagram is optional, whereas the inclusion of the checksum for TCP is mandatory.
- 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: There can be up to 40 bytes of optional information in the TCP header.

A TCP Connection

- In TCP, connection-oriented transmission requires three phases: connection establishment, data transfer, and connection termination.

A TCP Connection

- In TCP, connection-oriented transmission requires three phases: connection establishment, data transfer, and connection termination.
- Connection Establishment

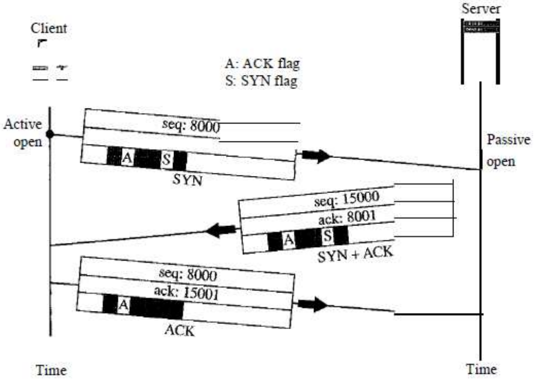


Figure: Connection establishment using three-way handshaking

A TCP Connection

- Connection Establishment: 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. It consumes one sequence number. When the data transfer starts, the sequence number is incremented by 1.

    Note: 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. It consumes one sequence number.

    Note: SYN +ACK segment cannot carry data, but does consume one sequence number.
  - 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. Note that 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.

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

A TCP Connection

- Connection Establishment
  - Simultaneous Open: A rare situation, called a simultaneous open, may occur when both processes issue an active open. In this case, both TCPs transmit a SYN + ACK segment to each other, and one single connection is established between them.

A TCP Connection

- Connection Establishment
  - Simultaneous Open: A rare situation, called a simultaneous open, may occur when both processes issue an active open. In this case, both TCPs transmit a SYN + ACK segment to each other, and one single connection is established between them.
  - SYN Flooding Attack:
    - This happens when a malicious attacker sends a large number of SYN segments to a server, pretending that each of them is coming from a different client by faking the source IP addresses in the datagrams.
    - The server, assuming that the clients are issuing an active open, allocates the necessary resources, such as creating communication tables and setting timers.
    - The TCP server then sends the SYN +ACK segments to the fake clients, which are lost.
    - During this time, a lot of resources are occupied without being used.
    - If, during this short time, the number of SYN segments is large, the server eventually runs out of resources and may crash.
    - The SYN flooding attack belongs to a type of security attack known as a denial-of-service attack, in which an attacker monopolizes a system with so many service requests that the system collapses and denies service to every request.

A TCP Connection

- Data Transfer: bi-directional data communication takes place.
    - Pushing Data: Delayed transmission and delayed delivery of data may not be acceptable by the application program.
    - Situation is handled by TCP efficiently.
    - The application program at the sending site can request a push operation.

A TCP Connection

- Data Transfer: bi-directional data communication takes place.
  - Pushing Data: Delayed transmission and delayed delivery of data may not be acceptable by the application program.
  - Situation is handled by TCP efficiently.
  - The application program at the sending site can request a push operation.
  - Urgent Data
    - TCP is a stream-oriented protocol, implies that 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.
    - However, on occasion an application program needs to send urgent bytes.
    - It means that the sending application program wants a piece of data to be read out of order by the receiving application program.
    - The solution is to send a segment with the URG bit set.

A TCP Connection

- Data 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.

A TCP Connection

- Data 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.
  - Three-way handshake

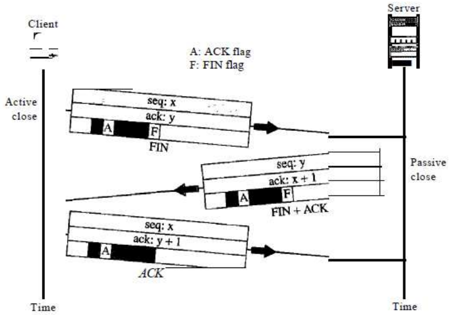


Figure: Connection termination using three-way handshake

A TCP Connection

- Data Termination:
  - Three-way handshake
    - 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.

      Note: The FIN segment consumes one sequence number ifit 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.

      Note: 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 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.

A TCP Connection

- Data 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.
    - It can occur when the server needs all the data before processing can begin. (Example: sorting)
    - 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.
    - After half-closing of the connection, data can travel from the server to the client and acknowledgments can travel from the client to the server.
    - The client cannot send any more data to the server.

Flow Control

- Few 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.

Flow Control

- Few 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.
- The sliding window protocol used by TCP uses the characteristics of both GBN and SR protocols.
    - It does not use NAKs similar to GBN.
    - The receiver holds the out-of-order segments until the missing ones arrive, similar to SR.

Flow Control

- Few 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.
- The sliding window protocol used by TCP uses the characteristics of both GBN and SR protocols.
  - It does not use NAKs similar to GBN.
  - The receiver holds the out-of-order segments until the missing ones arrive, similar to SR.
- Difference between TCP sliding window and sliding window at DLL
  - The sliding window of TCP is byte-oriented, while sliding window at DLL is frame-oriented.
  - TCP's sliding window is of variable size, while sliding window at DLL is of fixed size.

Flow Control

- The window is opened, closed, or shrunk.
- These three activities are in the control of the receiver (and depend on congestion in the network), not the sender.
- The sender must obey the commands of the receiver in this matter.

Flow Control

- The window is opened, closed, or shrunk.
- These three activities are in the control of the receiver (and depend on congestion in the network), not the sender.
- The sender must obey the commands of the receiver in this matter.
- Problem 1: 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?

Flow Control

- The window is opened, closed, or shrunk.
- These three activities are in the control of the receiver (and depend on congestion in the network), not the sender.
- The sender must obey the commands of the receiver in this matter.
- Problem 1: 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 $= 5000 - 1000 = 4000$.

Flow Control

- The window is opened, closed, or shrunk.
- These three activities are in the control of the receiver (and depend on congestion in the network), not the sender.
- The sender must obey the commands of the receiver in this matter.
- Problem 1: 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.

Flow Control

- The window is opened, closed, or shrunk.
- These three activities are in the control of the receiver (and depend on congestion in the network), not the sender.
- The sender must obey the commands of the receiver in this matter.
- Problem 1: 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.

Flow Control

- The window is opened, closed, or shrunk.
- These three activities are in the control of the receiver (and depend on congestion in the network), not the sender.
- The sender must obey the commands of the receiver in this matter.
- Problem 1: 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.
- Problem 2: 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?

Flow Control

- The window is opened, closed, or shrunk.
- These three activities are in the control of the receiver (and depend on congestion in the network), not the sender.
- The sender must obey the commands of the receiver in this matter.
- Problem 1: 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.
- Problem 2: 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.

Error Control

- 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, and time-out.
- Note:
  - 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 yet guarantees that no out-of-order segment is delivered to the process.