Module 9: The Transport Layer

Instructor Materials

CyberOps Associate v1.0

Instructor Materials – Module 9 Planning Guide

This PowerPoint deck is divided in two parts:

- Instructor Planning Guide

  - Information to help you become familiar with the module
  - Teaching aids

- Instructor Class Presentation

  - Optional slides that you can use in the classroom
  - Begins on slide # 8

    Note: Remove the Planning Guide from this presentation before sharing with anyone.

For additional help and resources go to the Instructor Home Page and Course Resources for this course. You also can visit the professional development site on www.netacad.com, the official Cisco Networking Academy Facebook page, or Instructor Only FB group.

What to Expect in this Module

To facilitate learning, the following features within the GUI may be included in this module:

Feature	Description
Animations	Expose learners to new skills and concepts.
Videos	Expose learners to new skills and concepts.
Check Your Understanding (CYU)	Per topic online quiz to help learners gauge content understanding.
Hands-On Labs	Labs designed for working with physical equipment.
Module Quizzes	Self-assessments that integrate concepts and skills learned throughout the series of topics presented in the module.
Module Summary	Briefly recaps module content.

Check Your Understanding

- Check Your Understanding activities are designed to let students quickly determine if they understand the content and can proceed, or if they need to review.

- Check Your Understanding activities **do not** affect student grades.

- There are no separate slides for these activities in the PPT. They are listed in the notes area of the slide that appears before these activities.

Module 9: Activities

What activities are associated with this module?

Page #	Activity Type	Activity Name	Optional?
9.1.11	Check Your Understanding	Compare TCP and UDP Characteristics	Recommended
9.2.5	Video	TCP 3-Way Handshake	Recommended
9.2.6	Lab	Using Wireshark to Observe the TCP 3-Way Handshake	Recommended
9.2.7	Check Your Understanding	TCP Connection and Termination Process	Recommended
9.3.2	Video	TCP Reliability – Sequence Numbers and Acknowledgements	Recommended
9.3.4	Video	TCP Reliability – Data Loss and Retransmission	Recommended
9.3.8	Lab	Exploring Nmap	Recommended
9.3.9	Check Your Understanding	Reliability and Flow Control	Recommended

Module 9: Best Practices

Prior to teaching Module 9, the instructor should:

- Review the activities and assessments for this module.

- Try to include as many questions as possible to keep students engaged during classroom presentation.

Topic 9.1

- Ask the class to share their understanding of the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) and then discuss the protocols with them.

- Explain the transport layer responsibilities and protocols.

- Describe the TCP and UDP headers and their fields. Ask the students to list out some basic differences between the two.

- Define the socket pairs.

Module 9: Best Practices (Contd.)

Topic 9.2

- Explain the TCP server processes to the class.
- Describe the processes of TCP connection establishment and session termination.
- Ask the class what do they know of three-way handshake and then explain the TCP three-way handshake analysis to them.

Topic 9.3

- Ask the class to mention some main features of TCP based on their knowledge.
- Discuss the reliability and flow control features of TCP.
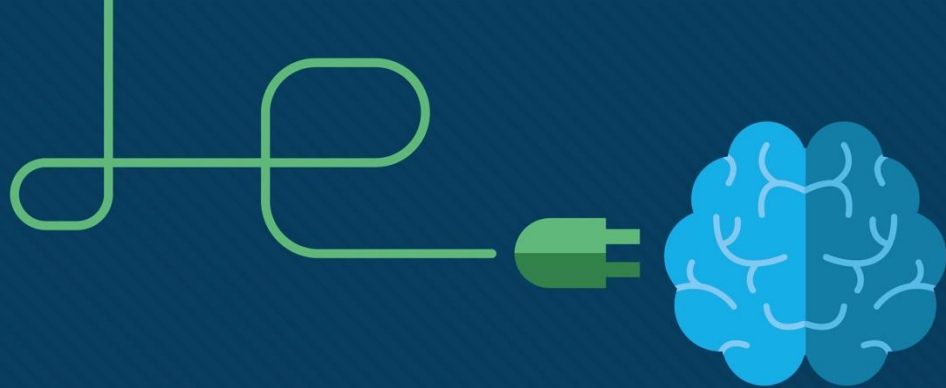
Module 9: The Transport Layer

CyberOps Associate v1.0

Module Objectives

Module Title: The Transport Layer

**Module Objective**: Explain how transport layer protocols support network functionality.

Topic Title	Topic Objective
Transport Layer Characteristics	Explain how transport layer protocols support network communication.
Transport Layer Session Establishment	Explain how the transport layer establishes communication sessions.
Transport Layer Reliability	Explain how the transport layer establishes reliable communications.

9.1 Transport Layer Characteristics

Role of the Transport Layer

- The transport layer is responsible for logical communications between applications running on different hosts.

- As shown in the figure, the transport layer is the link between the application layer and the lower layers that are responsible for network transmission.

- The transport layer has no knowledge of the destination host type, the type of media for which the data must travel, the path taken by the data, the congestion on a link, or the size of the network.

- The transport layer includes two protocols, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).
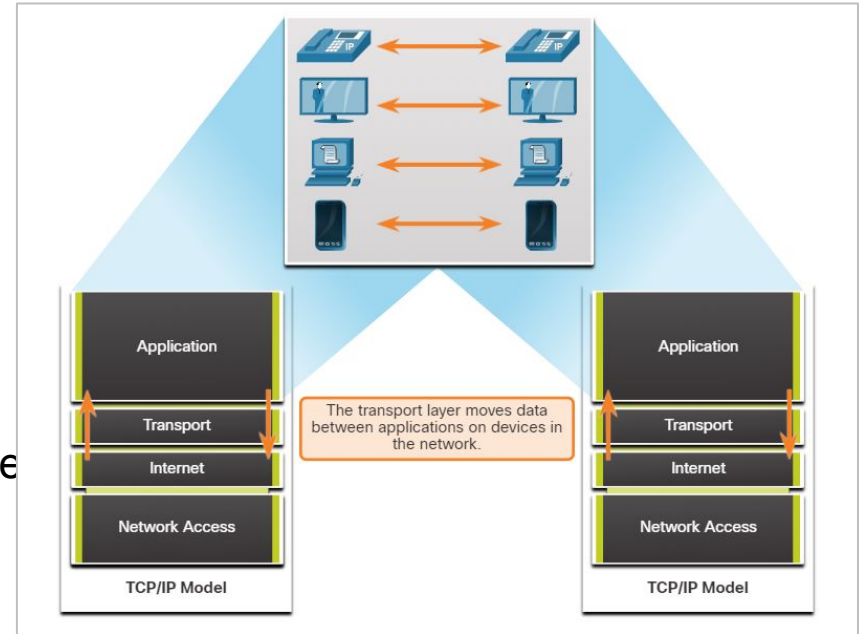
Transport Layer Responsibilities

The transport layer has many responsibilities.

Tracking Individual Conversations

• Each set of data flowing between a source application and a destination application is known as a conversation and is tracked separately.

• It is the responsibility of the transport layer to maintain and track these multiple conversations.

• As shown in the figure, a host may have multiple applications that are communicating across the network simultaneously.

• Most networks have a limitation on the amount of data that can be included in a single packet. Data must be divided into manageable pieces.
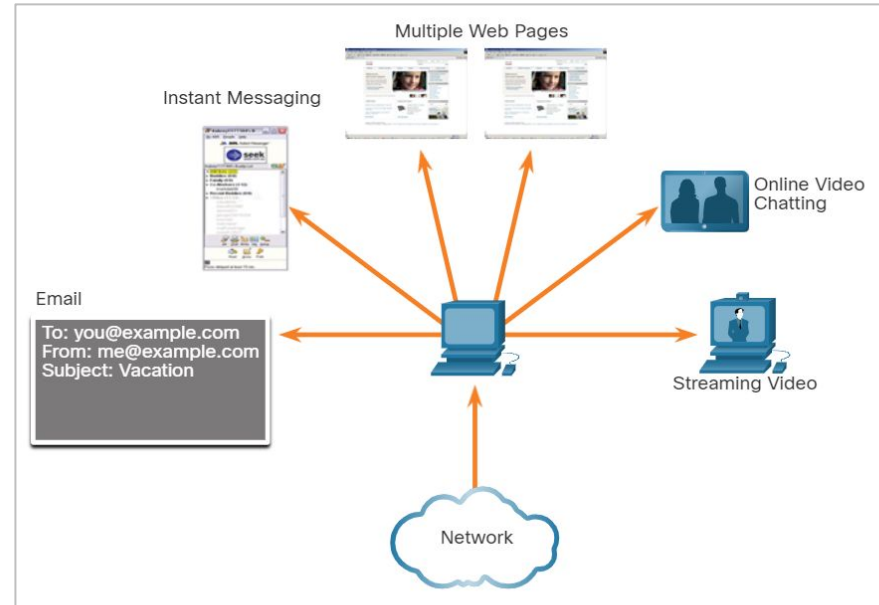
Transport Layer Responsibilities (Contd.)

Segmenting Data and Reassembling Segments

- It is the transport layer responsibility to divide the application data into appropriately sized blocks.

- Depending on the transport layer protocol used, the transport layer blocks are called either segments or datagrams.

- The figure shows the transport layer using different blocks for each conversation.

- The transport layer divides the data into smaller blocks (segments or datagrams) that are easier to manage and transport.
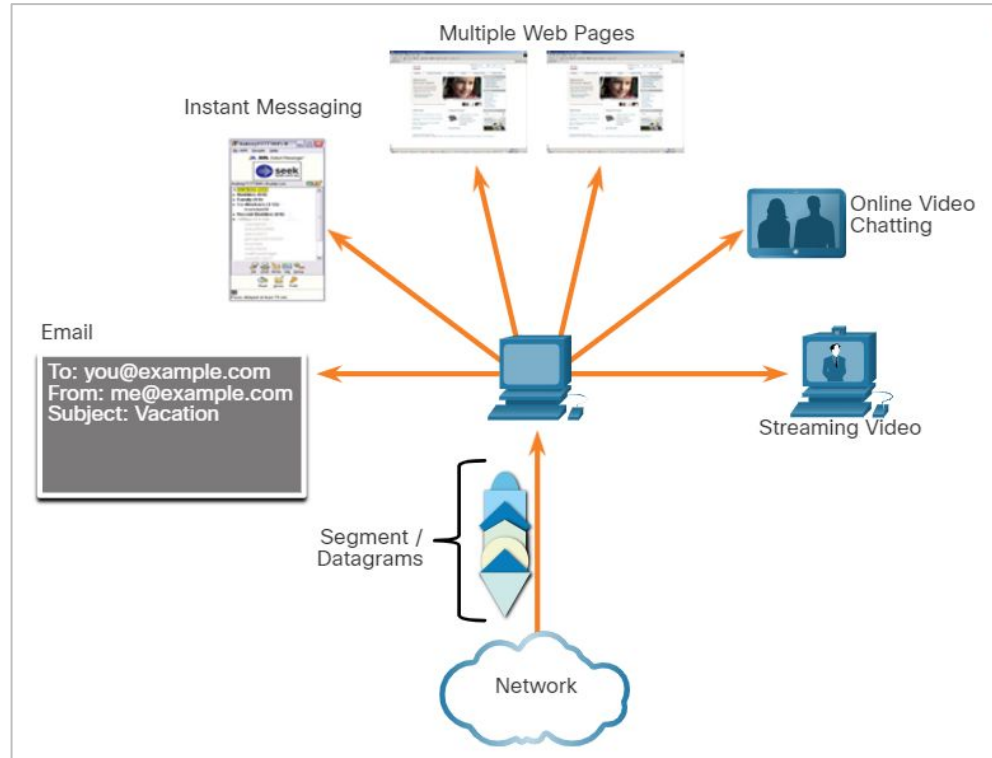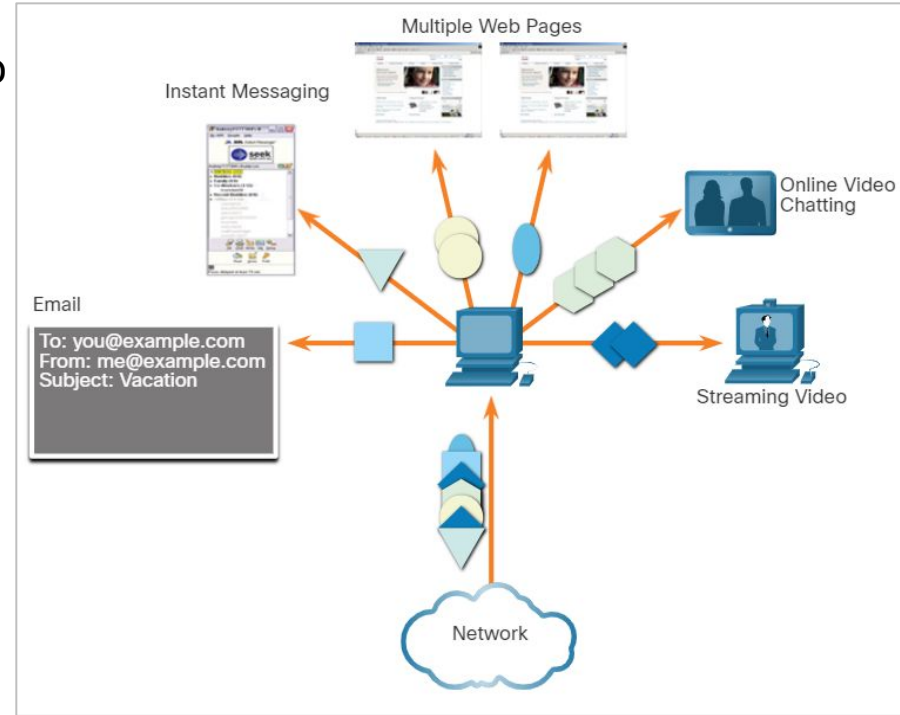
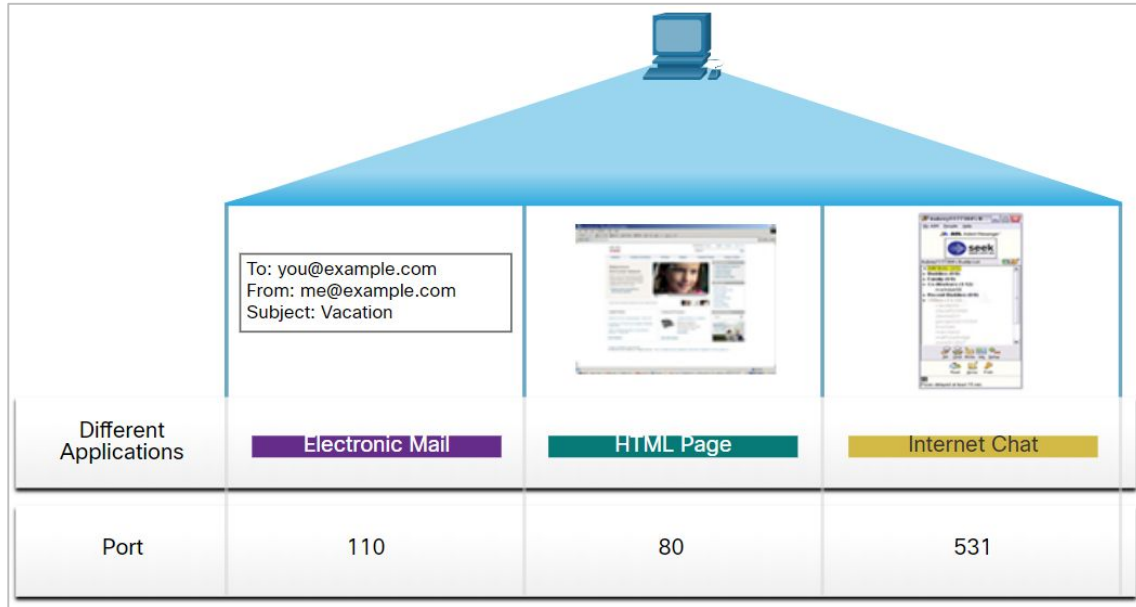# Transport Layer Responsibilities (Contd.)

Add Header Information

- The transport layer protocol also adds header information containing binary data organized into several fields to each block of data.

- The values in these fields enable various transport layer protocols to perform different functions in managing data communication.

- The header information is used by the receiving host to reassemble the blocks of data into a complete data stream for the receiving application layer program.

- The transport layer ensures that even with multiple application running on a device, all applications receive the correct data.

# Transport Layer Responsibilities (Contd.)

Identifying the Applications

- The transport layer must be able to separate and manage multiple communications with different transport requirement needs.

- To pass data streams to the proper applications, the transport layer identifies the target application using an identifier called a port number.

- As shown in the figure, each software process that needs to access the network is assigned a port number unique to that host.

Different Applications	Electronic Mail	HTML Page	Internet Chat
Port	110	80	531

To: you@example.com
From: me@example.com
Subject: Vacation

Transport Layer Responsibilities (Contd.)

Conversation Multiplexing

- Sending some types of data across a network, as one complete communication stream, can consume all the available bandwidth.

- This prevents other communication conversations from occurring at the same time and also make error recovery and retransmission of damaged data difficult.

- As shown in the figure, the transport layer uses segmentation and multiplexing to enable different communication conversations to be interleaved on the same network.

- Error checking can be performed on the data in the segment, to determine if the segment was altered during transmission.
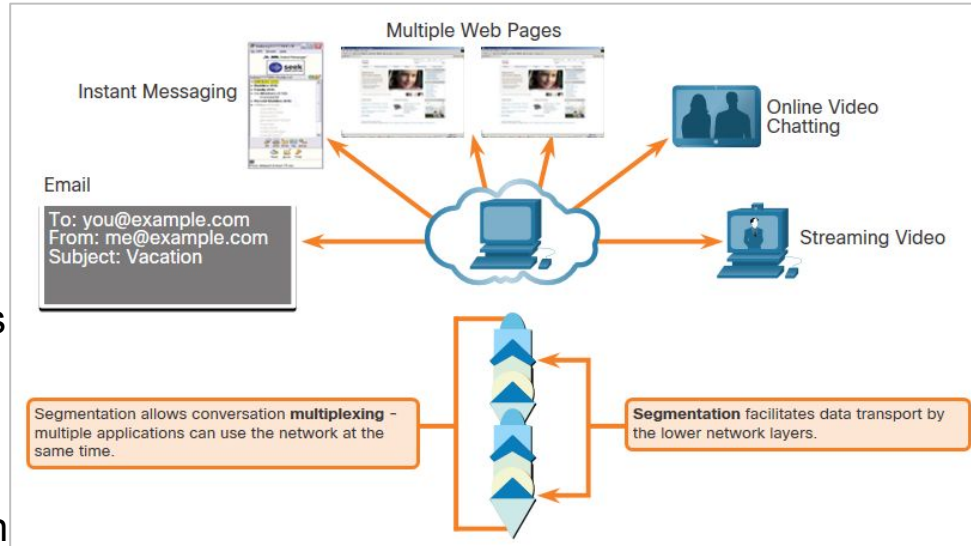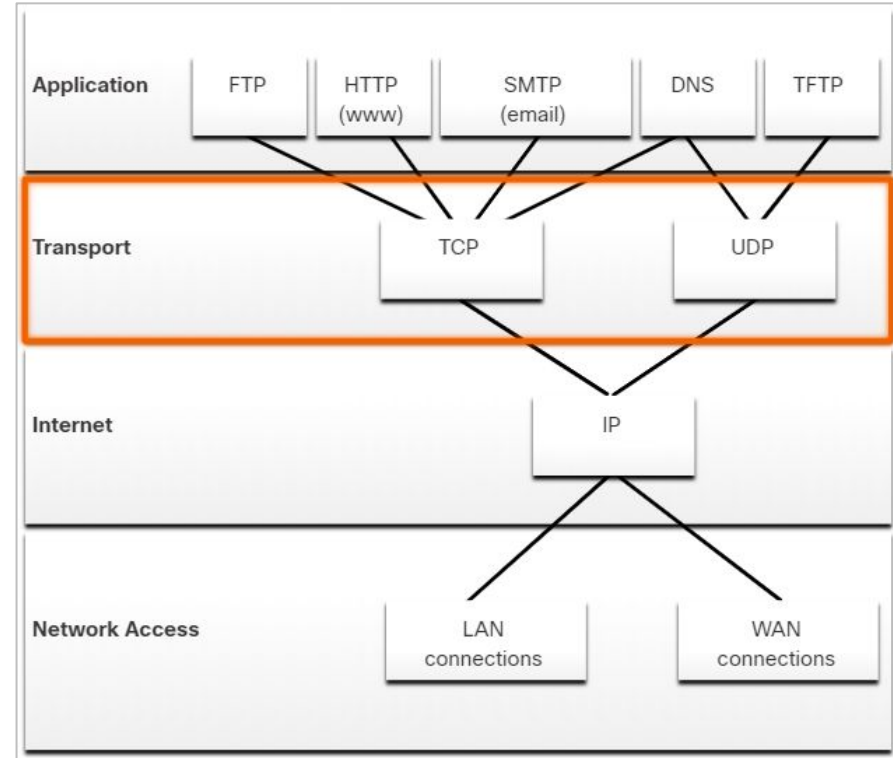
Transport Layer Protocols

- IP is concerned only with the structure, addressing, and routing of packets.

- IP does not specify how the delivery or transportation of the packets takes place.

- Transport layer protocols (TCP and UDP) specify how to transfer messages between hosts, and are responsible for managing reliability requirements of a conversation.

- The transport layer includes the TCP and UDP protocols.

- Different applications have different transport reliability requirements. Therefore, TCP/IP provides two transport layer protocols, as shown in the figure.

Transmission Control Protocol (TCP)

- TCP is considered a reliable, full-featured transport layer protocol, which ensures that all of the data arrives at the destination.

- TCP includes fields which ensure the delivery of the application data. These fields require additional processing by the sending and receiving hosts.

- TCP transport is analogous to sending packages that are tracked from source to destination.

- TCP provides reliability and flow control using these basic operations:

  - Number and track data segments transmitted to a specific host from a specific application

  - Acknowledge received data

  - Retransmit any unacknowledged data after a certain amount of time

  - Sequence data that might arrive in wrong order

  - Send data at an efficient rate that is acceptable by the receiver

  Note: TCP divides data into segments.
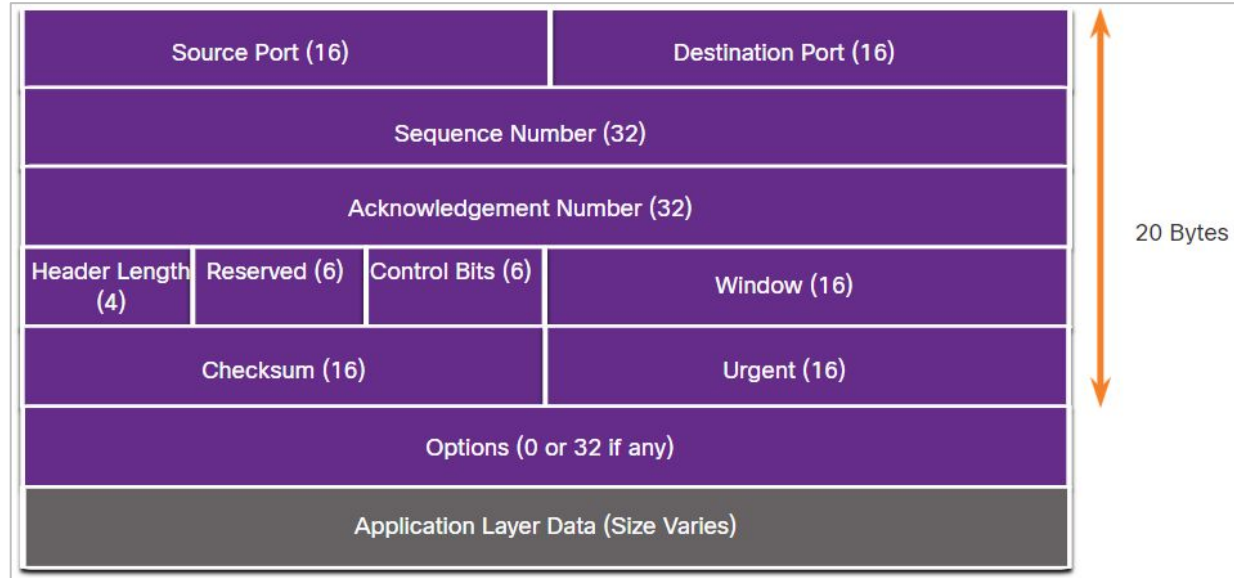
Transmission Control Protocol (TCP) (Contd.)

In order to maintain the state of a conversation and track the information, TCP must first establish a connection between the sender and the receiver. This is why TCP is known as a connection-oriented protocol.
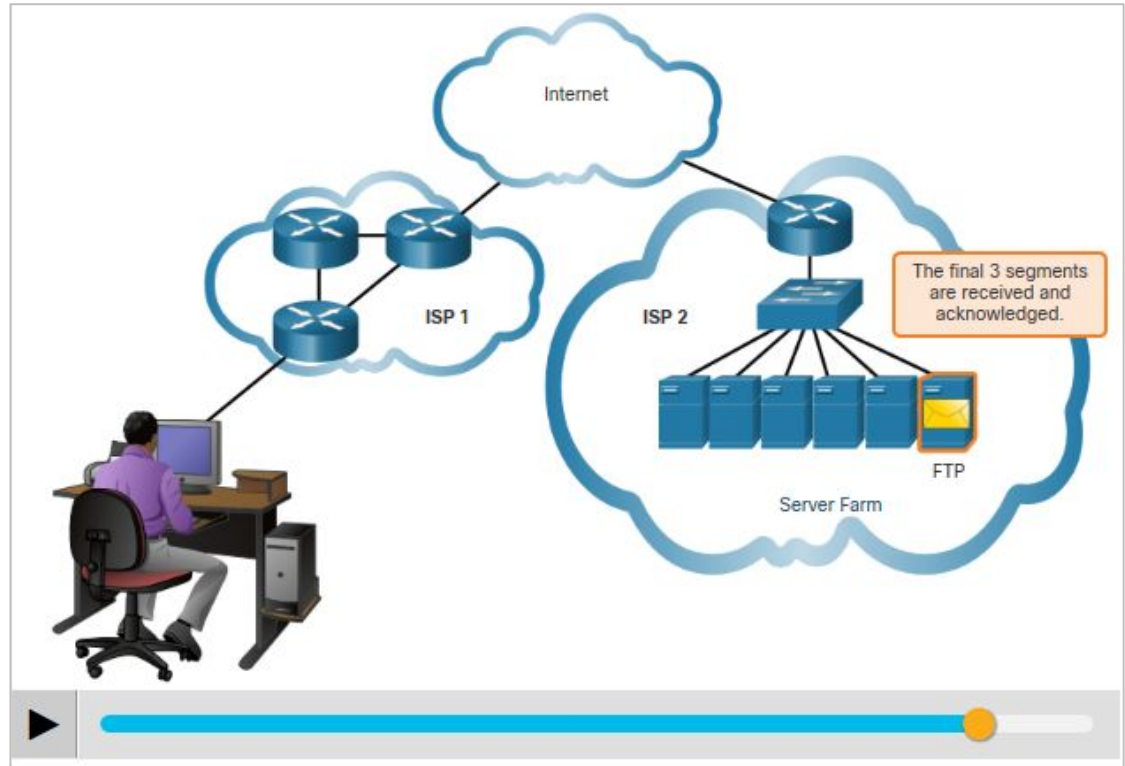
TCP Header

- TCP is a stateful protocol as it keeps track of the state of the communication session.

- To track the state of a session, TCP records which information it has sent and which information has been acknowledged.

- The stateful session begins with the session establishment and ends with the session termination.

- A TCP segment adds 20 bytes (160 bits) of overhead when encapsulating the application layer data. The figure shows the fields in a TCP header.

Source Port (16)		Destination Port (16)	
Sequence Number (32)			
Acknowledgement Number (32)			
Header Length (4)	Reserved (6)	Control Bits (6)	Window (16)
Checksum (16)		Urgent (16)	
Options (0 or 32 if any)			
Application Layer Data (Size Varies)			

20 Bytes

TCP Header Fields

The table identifies and describes the ten fields in a TCP header.

| TCP Header Field	Description
Source Port	A 16-bit field used to identify the source application by port number.
Destination Port	A 16-bit field used to identify the destination application by port number.
Sequence Number	A 32-bit field used for data reassembly purposes.
Acknowledgment Number	A 32-bit field used to indicate that data has been received and the next byte expected from the source.
Header Length	A 4-bit field known as "data offset" that indicates the length of the TCP segment header.
Reserved	A 6-bit field that is reserved for future use.
Control bits	A 6-bit field that includes bit codes, or flags, which indicate the purpose and function of the TCP segment.
Window size	A 16-bit field used to indicate the number of bytes that can be accepted at one time.
Checksum	A 16-bit field used for error checking of the segment header and data.
Urgent	A 16-bit field used to indicate if the contained data is urgent.

User Datagram Protocol (UDP)

- UDP is a simpler transport layer protocol than TCP.

- It does not provide reliability and flow control, which means it requires fewer header fields.

- The sender and the receiver UDP processes do not have to manage reliability and flow control, this means UDP datagrams can be processed faster than TCP segments.

- UDP provides the basic functions for delivering datagrams between the appropriate applications, with very little overhead and data checking.

- UDP is a connectionless protocol. Because UDP does not provide reliability or flow control, it does not require an established connection.

- UDP is also known as a stateless protocol. Because UDP does not track information sent or received between the client and server.

Note: *UDP divides data into datagrams that are also referred to as segments.*

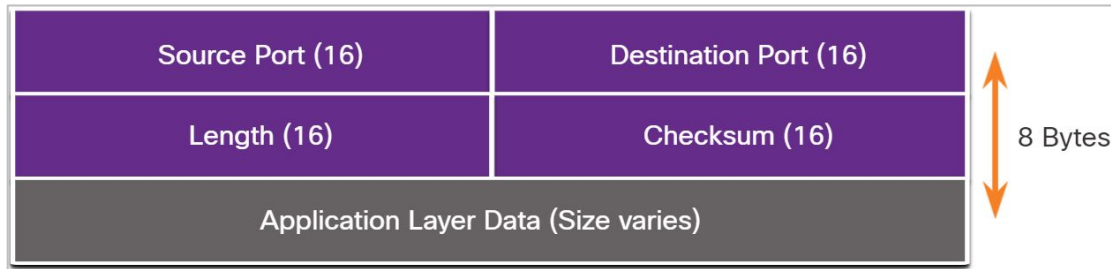# User Datagram Protocol (UDP) (Contd.)

- UDP is also known as a best-effort delivery protocol because there is no acknowledgment that the data is received at the destination.

- UDP is like placing a regular, nonregistered, letter in the mail. The sender of the letter is not aware of the availability of the receiver to receive the letter. Nor is the post office responsible for tracking the letter or informing the sender if the letter does not arrive at the final destination.
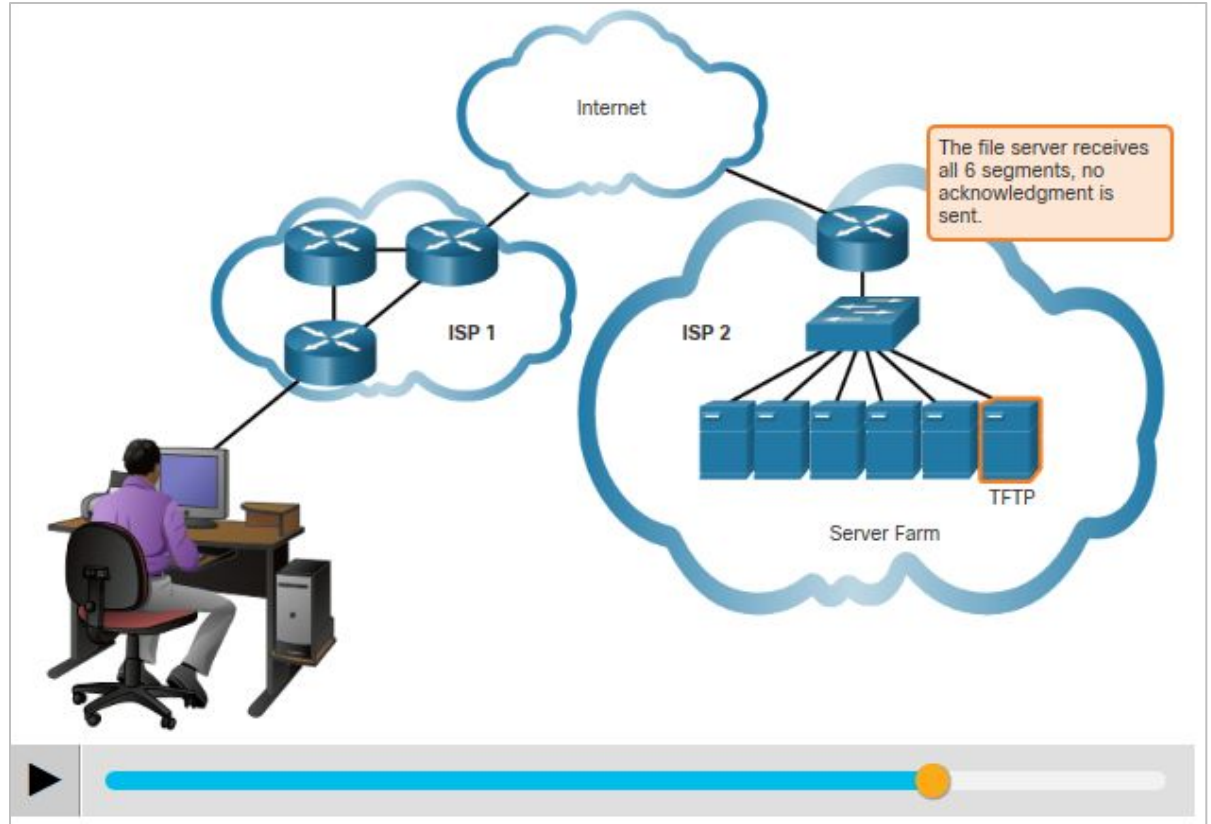
UDP Header

- UDP is a stateless protocol meaning neither the client, nor the server, tracks the state of the communication session. If reliability is required when using UDP as the transport protocol, it must be handled by the application.

- The requirements for delivering live video and voice over the network is the data continues to flow quickly. Live video and voice applications can tolerate some data loss and are perfectly suited to UDP.

- The blocks of communication in UDP are called datagrams, or segments. These datagrams are sent as best effort by the transport layer protocol.

- The UDP header is only has four fields and requires 8 bytes (64 bits). The figure shows the fields in a UDP header.

Source Port (16)	Destination Port (16)	
Length (16)	Checksum (16)	8 Bytes
Application Layer Data (Size varies)		

UDP Header Fields

The table identifies and describes the four fields in a UDP header.

| UDP Header Field	Description
Source Port	A 16-bit field used to identify the source application by port number.
Destination Port	A 16-bit field used to identify the destination application by port number.
Sequence Number	A 32-bit field used for data reassembly purposes.
Length	A 16-bit field that indicates the length of the UDP datagram header.
Checksum	A 16-bit field used for error checking of the datagram header and data.

Socket Pairs

- The source and destination ports are placed within the segment. The segments are then encapsulated within an IP packet.

- The IP packet contains the IP address of the source and destination. The combination of the source IP address and source port number, or the destination IP address and destination port number is known as a socket.

- Sockets enable multiple processes, running on a client, to distinguish themselves from each other, and multiple connections to a server process to be distinguished from each other.

- The source port number acts as a return address for the requesting application.

- The transport layer keeps track of this port and the application that initiated the request so that when a response is returned, it can be forwarded to the correct application.
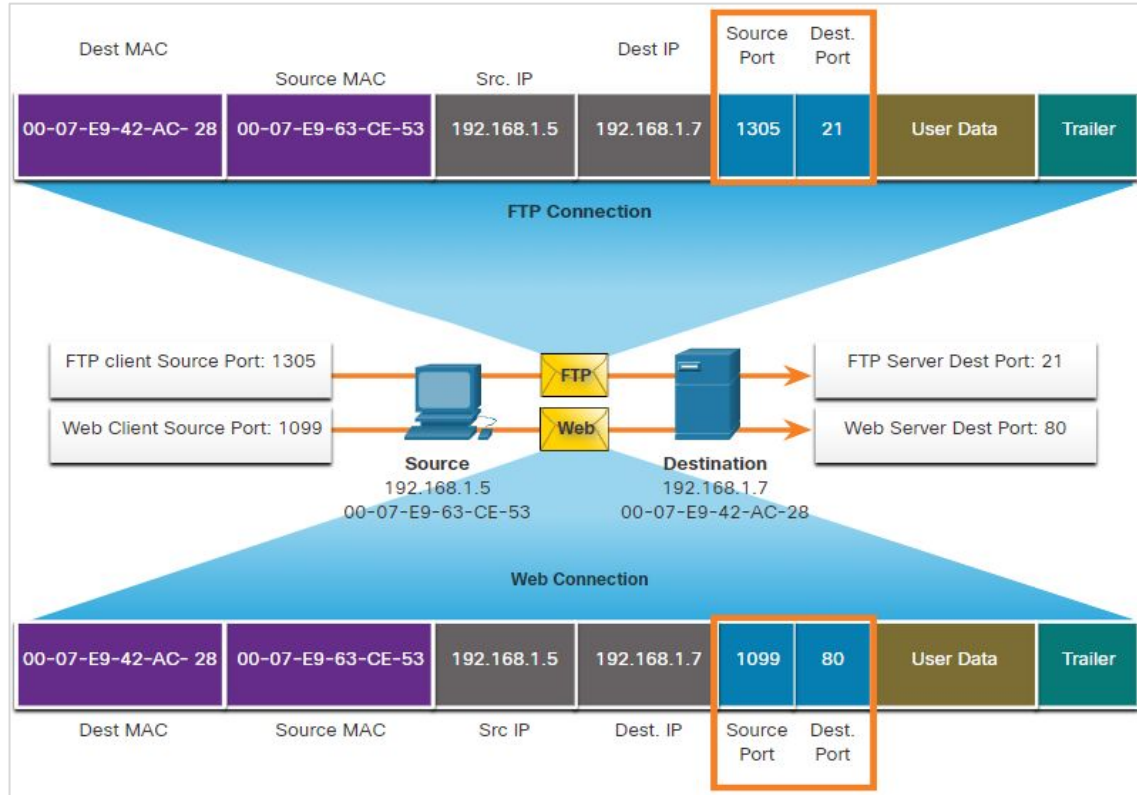
Socket Pairs (Contd.)

- In the figure, the PC is simultaneously requesting FTP and web services from the destination server.

- The FTP request generated by the PC includes the Layer 2 MAC addresses and the Layer 3 IP addresses. The request also identifies the source port number 1305 and destination port, identifying the FTP services on port 21.

- The host also has requested a web page from the server using the same Layer 2 and Layer 3 addresses.
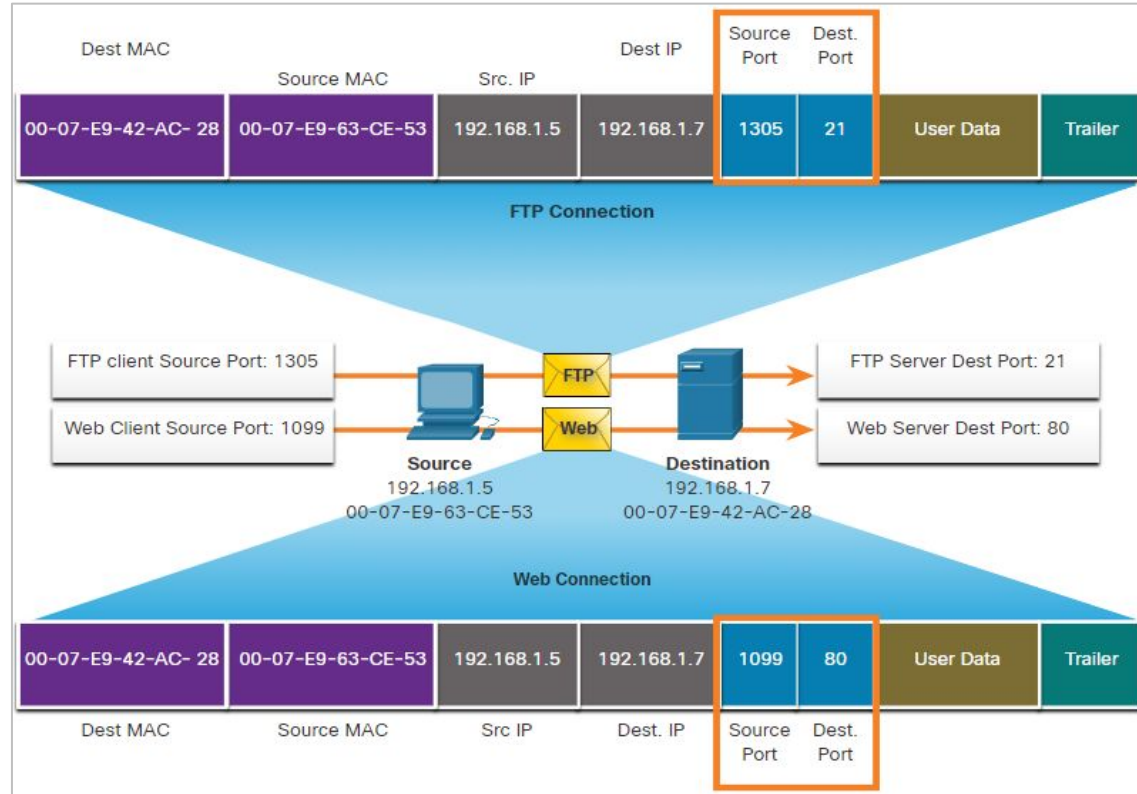
Socket Pairs (Contd.)

- It is using the source port number 1099 and destination port identifying the web service on port 80.

- The socket is used to identify the server and service being requested by the client.

- A client socket with 1099 representing the source port number might be 192.168.1.5:1099. The socket on a web server might be 192.168.1.7:80. Together, these two sockets combine to form a *socket pair*: 192.168.1.5:1099, 192.168.1.7:80

Dest MAC	Source MAC	Src. IP	Dest IP	Source Port	Dest. Port	User Data	Trailer
00-07-E9-42-AC- 28	00-07-E9-63-CE-53	192.168.1.5	192.168.1.7	1305	21	User Data	Trailer

FTP Connection

FTP client Source Port: 1305 → FTP → FTP Server Dest Port: 21

Web Client Source Port: 1099 → Web → Web Server Dest Port: 80

Source
192.168.1.5
00-07-E9-63-CE-53

Destination
192.168.1.7
00-07-E9-42-AC-28

Web Connection

Dest MAC	Source MAC	Src IP	Dest. IP	Source Port	Dest. Port	User Data	Trailer
00-07-E9-42-AC-28	00-07-E9-63-CE-53	192.168.1.5	192.168.1.7	1099	80	User Data	Trailer

cisco

9.2 Transport Layer Session Establishment
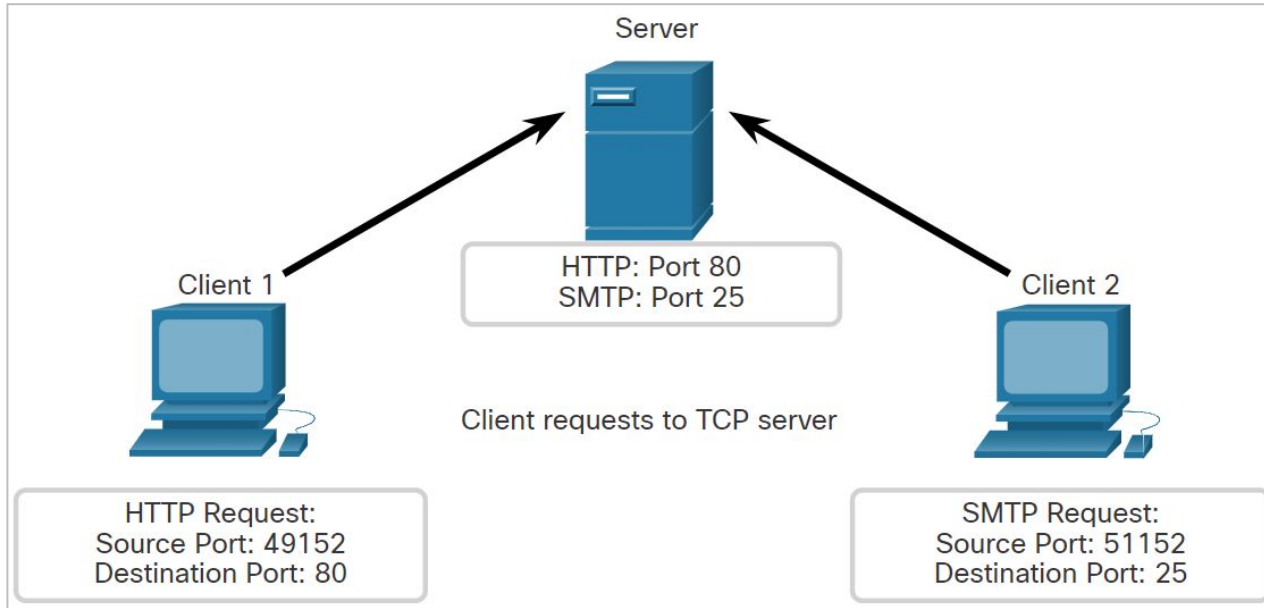
TCP Server Processes

- Each application process running on a server is configured to use a port number. The port number is either automatically assigned or configured manually by a system administrator.

- An individual server cannot have two services assigned to the same port number within the same transport layer services.

- A host running a web server application and a file transfer application cannot have both configured to use the same port, such as TCP port 80.

- An active server application assigned to a specific port is considered open, which means that the transport layer accepts, and processes segments addressed to that port.

- Any incoming client request addressed to the correct socket is accepted, and the data is passed to the server application.

- There can be many ports open simultaneously on a server, one for each active server application.

TCP Server Processes (Contd.)

Clients Sending TCP Requests

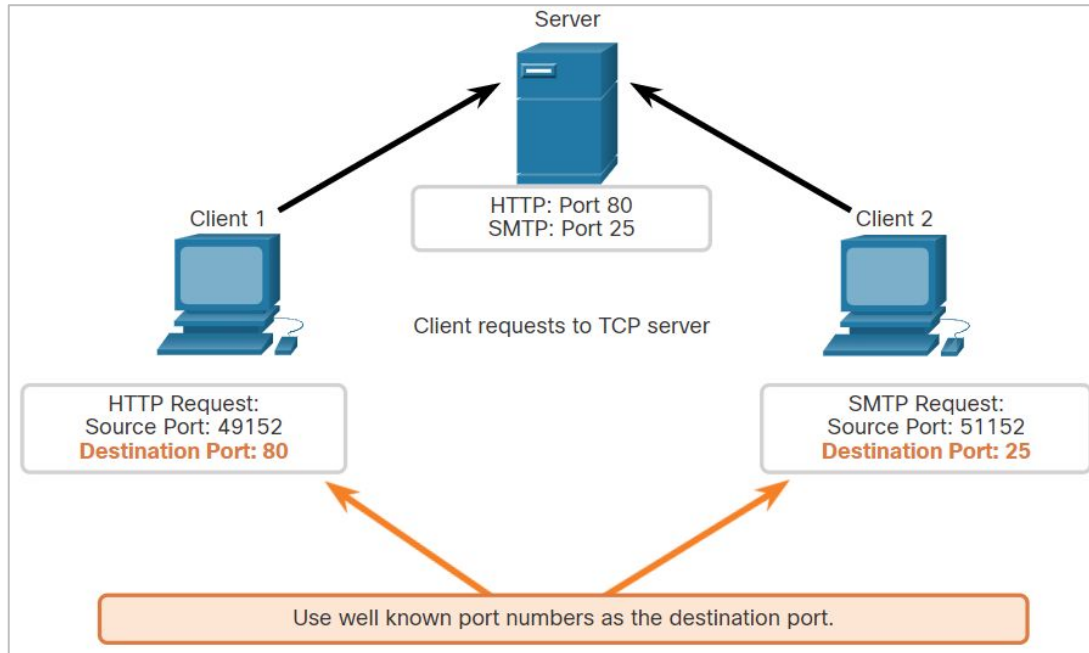Client 1 is requesting web services and Client 2 is requesting email service of the same sever.

Server

HTTP: Port 80
SMTP: Port 25

Client 1

Client 2

Client requests to TCP server

HTTP Request:
Source Port: 49152
Destination Port: 80

SMTP Request:
Source Port: 51152
Destination Port: 25

TCP Server Processes (Contd.)

Request Destination Ports

Client 1 is requesting web services using well-known destination port 80 (HTTP) and Client 2 is requesting email service using well-known port 25 (SMTP).

Server

HTTP: Port 80
SMTP: Port 25

Client 1

Client 2

Client requests to TCP server

HTTP Request:
Source Port: 49152
Destination Port: 80

SMTP Request:
Source Port: 51152
Destination Port: 25

Use well known port numbers as the destination port.

TCP Server Processes (Contd.)

Request Source Ports

Client requests dynamically generate a source port number. In this case, Client 1 is using source port 49152 and Client 2 is using source port 51152.
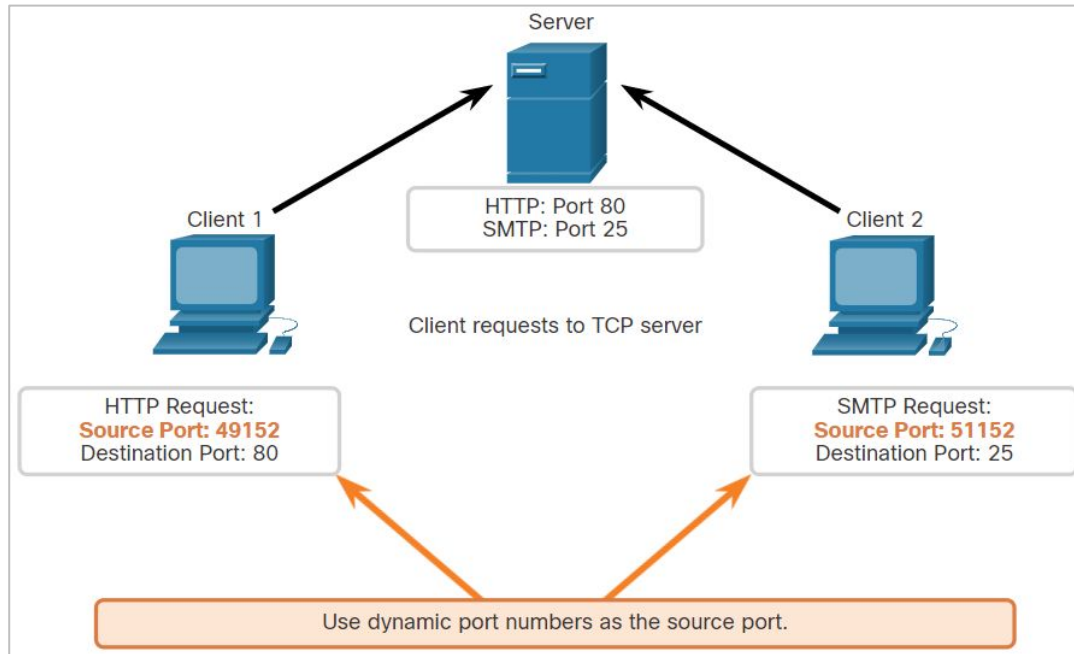
TCP Server Processes (Contd.)

Response Destination Ports

When the server responds to the client requests, it reverses the destination and source ports of the initial request. Notice that the Server response to the web request now has destination port 49152 and the email response now has destination port 51152.
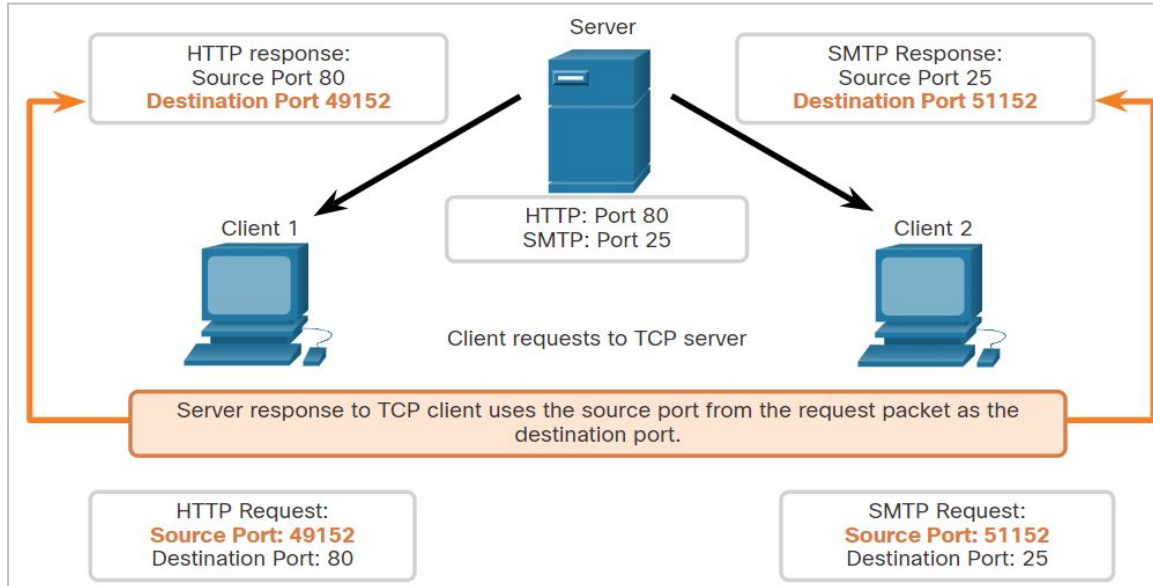
TCP Server Processes (Contd.)

Response Source Ports

The source port in the server response is the original destination port in the initial requests.
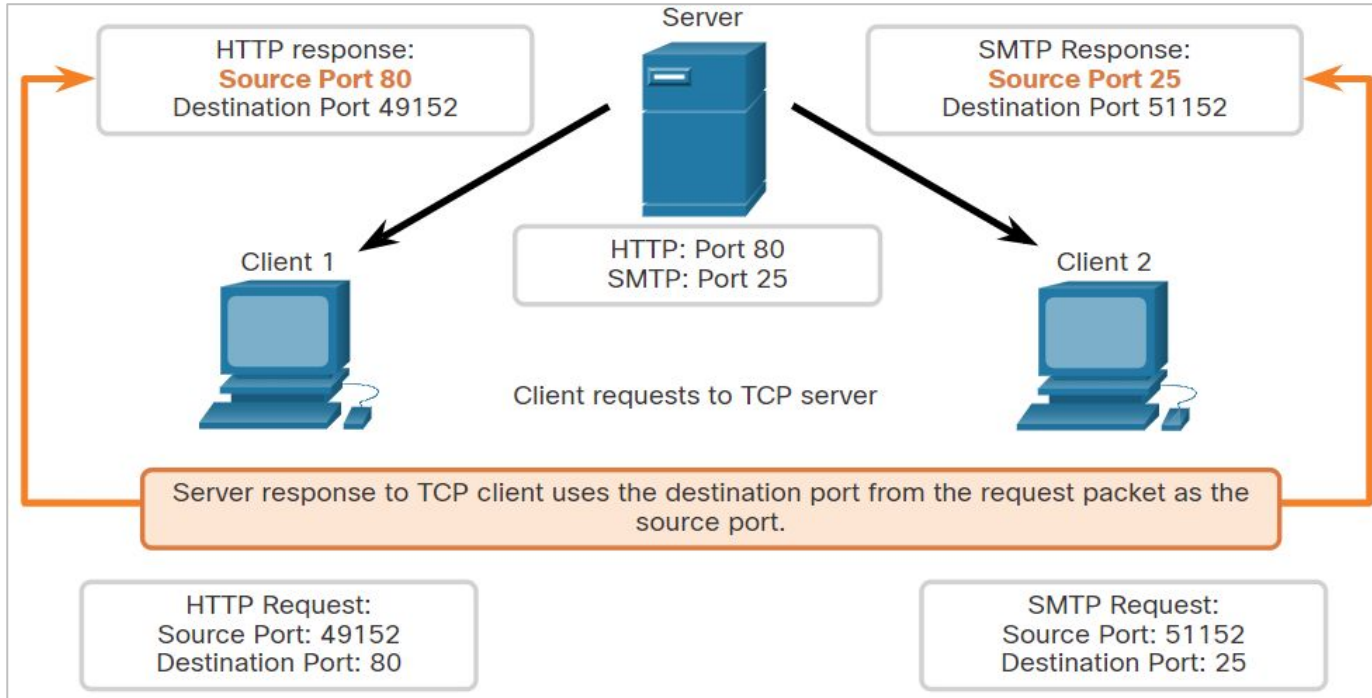
TCP Connection Establishment

- In TCP connections, the host client establishes the connection with the server using the three-way handshake process.

- The three-way handshake validates that the destination host is available to communicate.

- The TCP connection establishment steps are:

  - **Step 1. SYN:** The initiating client requests a client-to-server communication session with the server.
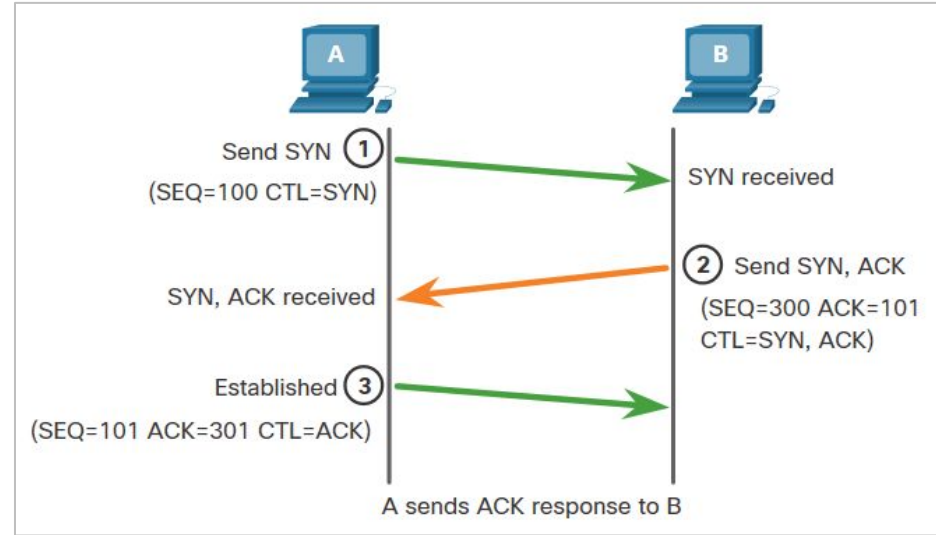

Send SYN ①
(SEQ=100 CTL=SYN)

SYN received

A sends SYN request to B
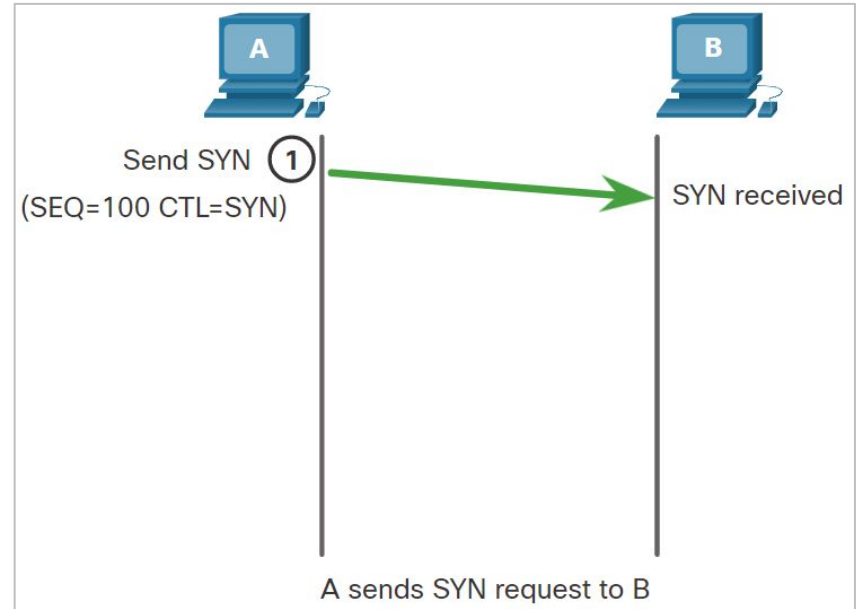
TCP Connection Establishment (Contd.)

Step 2. ACK and SYN: The server acknowledges the client-to-server communication session and requests a server-to-client communication session.
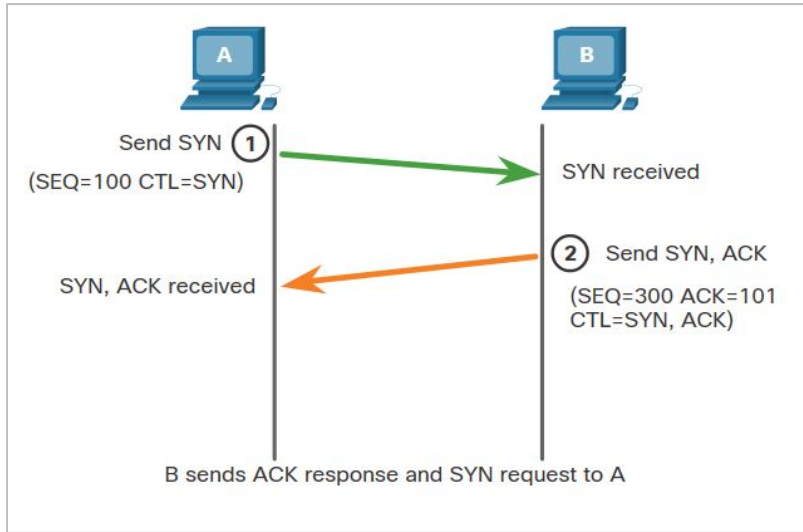
Step 3. ACK: The initiating client acknowledges the server-to-client communication session.
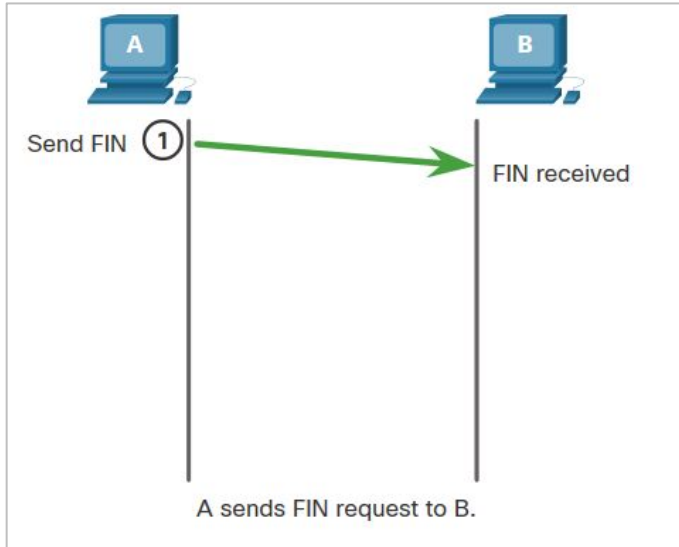
Session Termination

- To close a connection, the Finish (FIN) control flag must be set in the segment header.

- To end each one-way TCP session, a two-way handshake, consisting of a FIN segment and an Acknowledgment (ACK) segment, is used.

- Therefore, to terminate a single conversation supported by TCP, four exchanges are needed to end both sessions. Either the client or the server can initiate the termination.

- The terms client and server are used as a reference for simplicity, but any two hosts that have an open session can initiate the termination process.

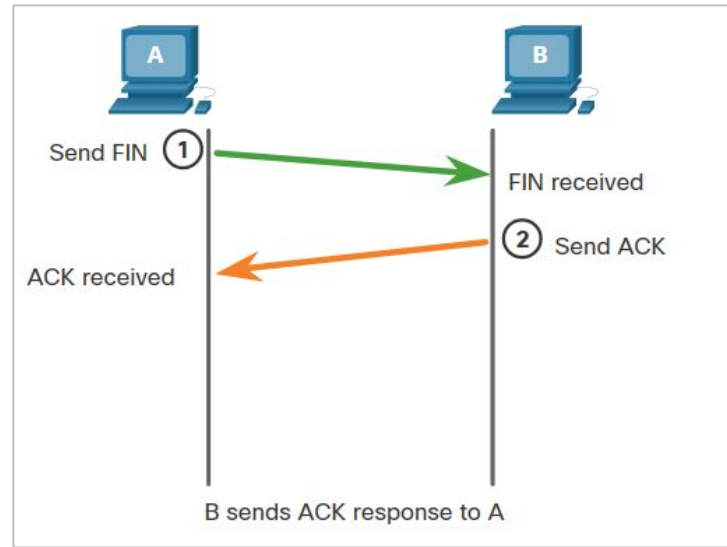- When all segments have been acknowledged, the session is closed.

Session Termination (Contd.)

The session termination steps are:

Step 1. FIN: When the client has no more data to send in the stream, it sends a segment with the FIN flag set.
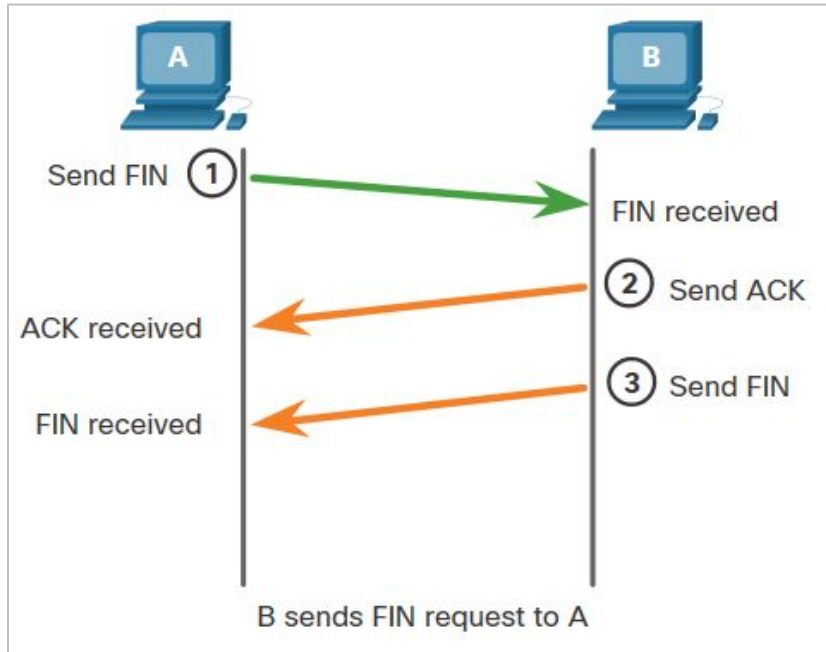
Step 2. ACK: The server sends an ACK to acknowledge the receipt of the FIN to terminate the session from client to server.
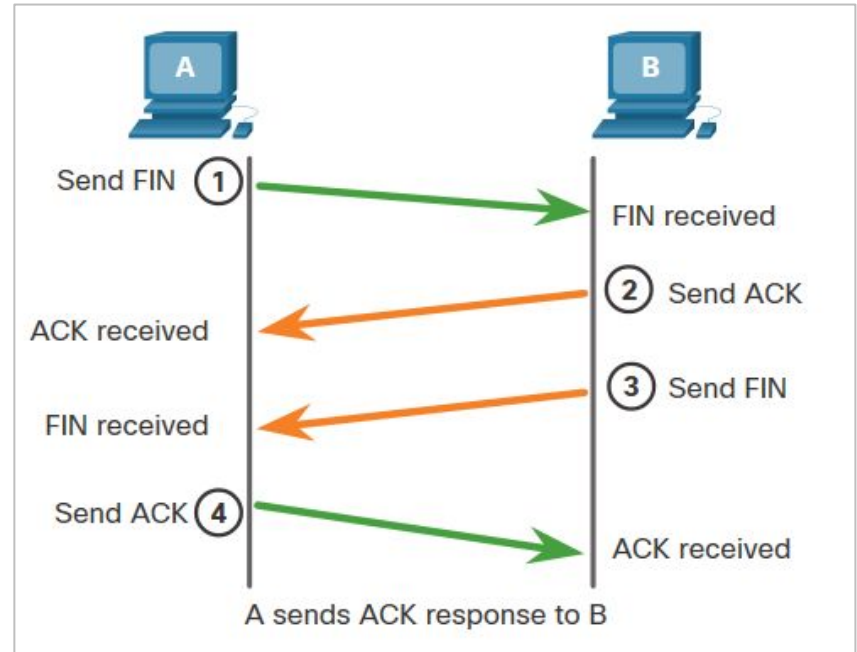
Session Termination (Contd.)

Step 3. FIN: The server sends a FIN to the client to terminate the server-to-client session.

Step 4. ACK: The client responds with an ACK to acknowledge the FIN from the server.

TCP Three-way Handshake Analysis

- Hosts maintain state, track each data segment within a session, and exchange information about the data is received using the information in the TCP header.

- TCP is a full-duplex protocol, where each connection represents two one-way communication sessions. To establish the connection, the hosts perform a three-way handshake. As shown in the figure, control bits in the TCP header indicate the progress and status of the connection.

- The functions of the three-way handshake are:

  - It establishes that the destination device is present on the network.

  - It verifies that the destination device has an active service and is accepting requests on the destination port number that the initiating client intends to use.

  - It informs the destination device that the source client intends to establish a communication session on that port number.

- After the communication is completed the sessions are closed, and the connection is terminated. The connection and session mechanisms enable TCP reliability function.

TCP Three-way Handshake Analysis (Contd.)

The six bits in the Control Bits field of the TCP segment header are also known as flags. A flag is a bit that is set to either on or off. The six control bits flags are as follows:

- **URG** - Urgent pointer field significant
- **ACK** - Acknowledgment flag used in connection establishment and session termination
- **PSH** - Push function
- **RST** - Reset the connection when an error or timeout occurs
- **SYN** - Synchronize sequence numbers used in connection establishment
- **FIN** - No more data from sender and used in session termination
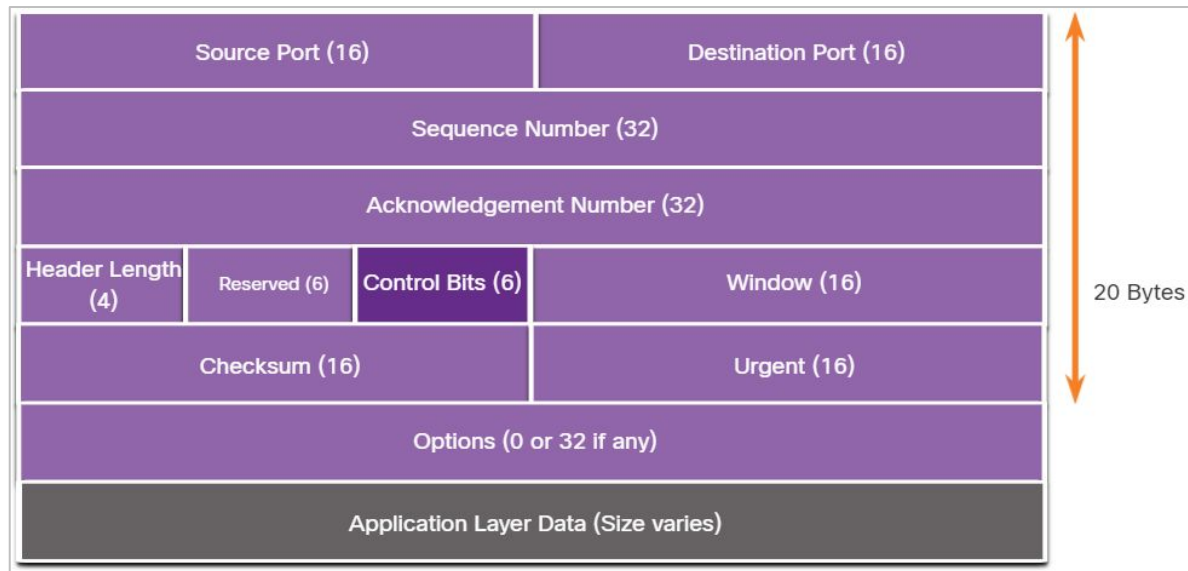
Video – TCP 3-Way Handshake

Watch the video to learn more of the TCP 3-way handshake.
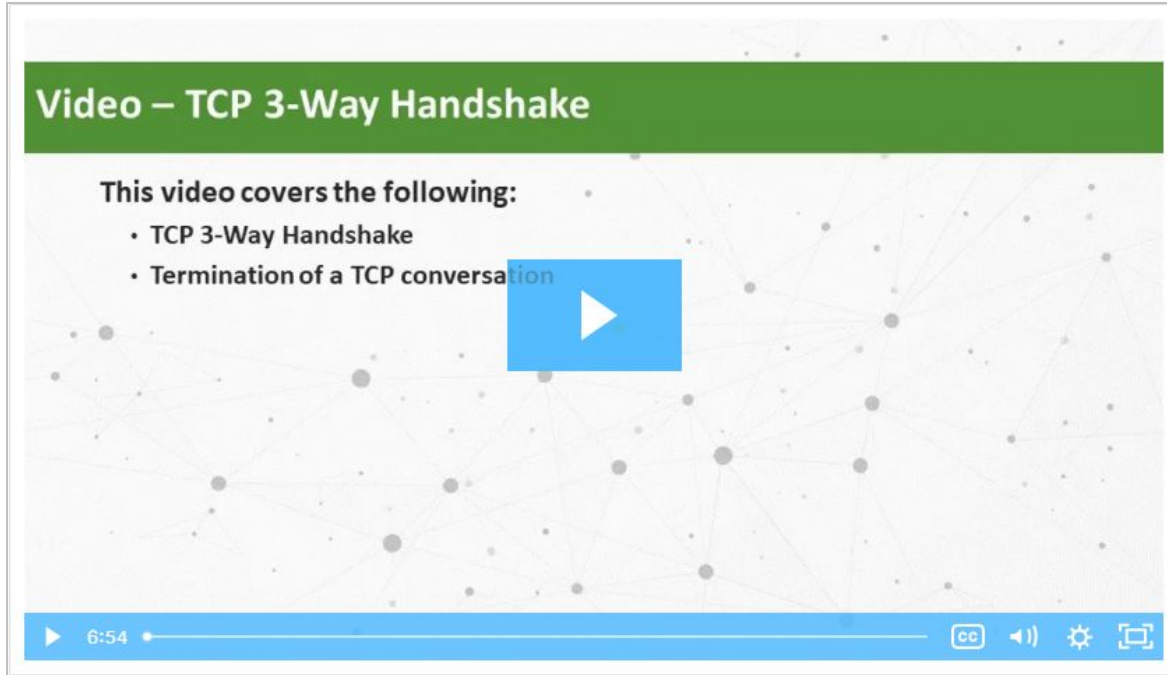
Lab – Using Wireshark to Observe the TCP 3-Way Handshake

In this lab, you will complete the following objectives:

- **Part 1:** Prepare the Hosts to Capture the Traffic

- **Part 2:** Analyze the Packets using Wireshark

- **Part 3:** View the Packets using tcpdump

9.3 Transport Layer Reliability

TCP Reliability - Guaranteed and Ordered Delivery

- There may be times when either TCP segments do not arrive at their destination or arrive out of order.

- For the original message to be understood by the recipient, all the data must be received and the data in these segments must be reassembled into the original order.

- Sequence numbers are assigned in the header for each packet to achieve this goal. The sequence number represents the first data byte of the TCP segment.

- During session setup, an initial sequence number (ISN) is set, which represents the starting value of the bytes that are transmitted to the receiving application.

- As data is transmitted during the session, the sequence number is incremented by the number of bytes that have been transmitted.

- This data byte tracking enables each segment to be uniquely identified and acknowledged. Missing segments can then be identified.

- The ISN is effectively a random number which prevents certain types of malicious attacks.

TCP Reliability - Guaranteed and Ordered Delivery (Contd.)

- Segment sequence numbers indicate how to reassemble and reorder received segments, as shown in the figure.

- The receiving TCP process places the data from a segment into a receiving buffer.

- Segments are then placed in the proper sequence order and passed to the application layer when reassembled.

- Any segments that arrive with sequence numbers that are out of order are held for later processing.

- Then, when the segments with the missing bytes arrives, these segments are processed in order.
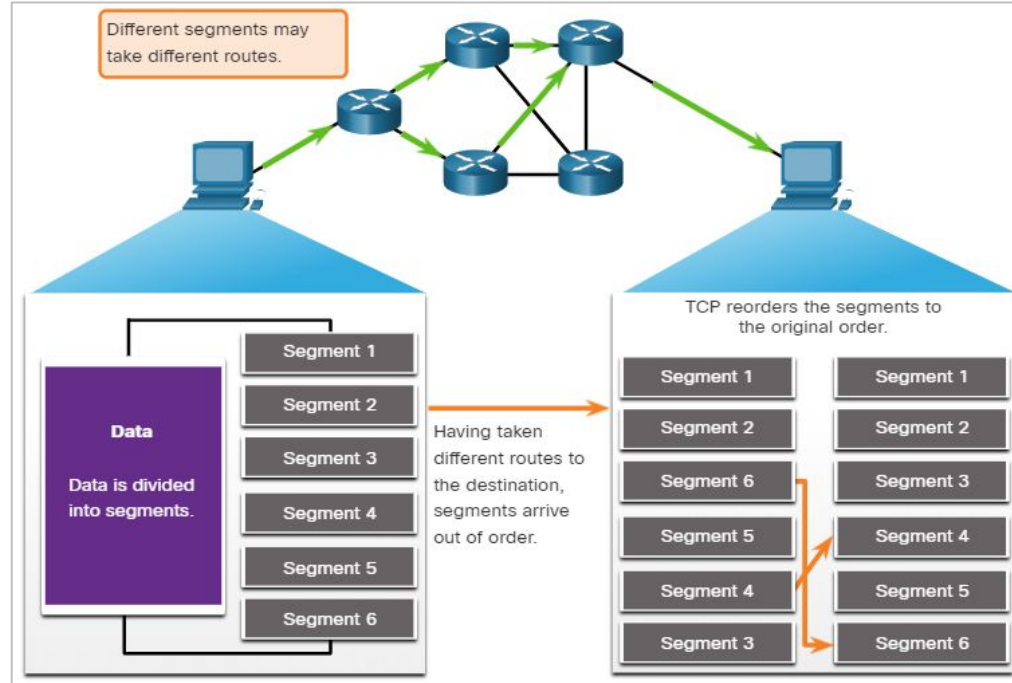
Video - TCP Reliability – Sequence Numbers and Acknowledgements

- One of the functions of TCP is to ensure that each segment reaches its destination. The TCP services on the destination host acknowledge the data that have been received by the source application.

- Click Play in the figure to view a lesson on TCP sequence numbers and acknowledgments.

Video – TCP Reliability- Sequence Numbers and Acknowledgements

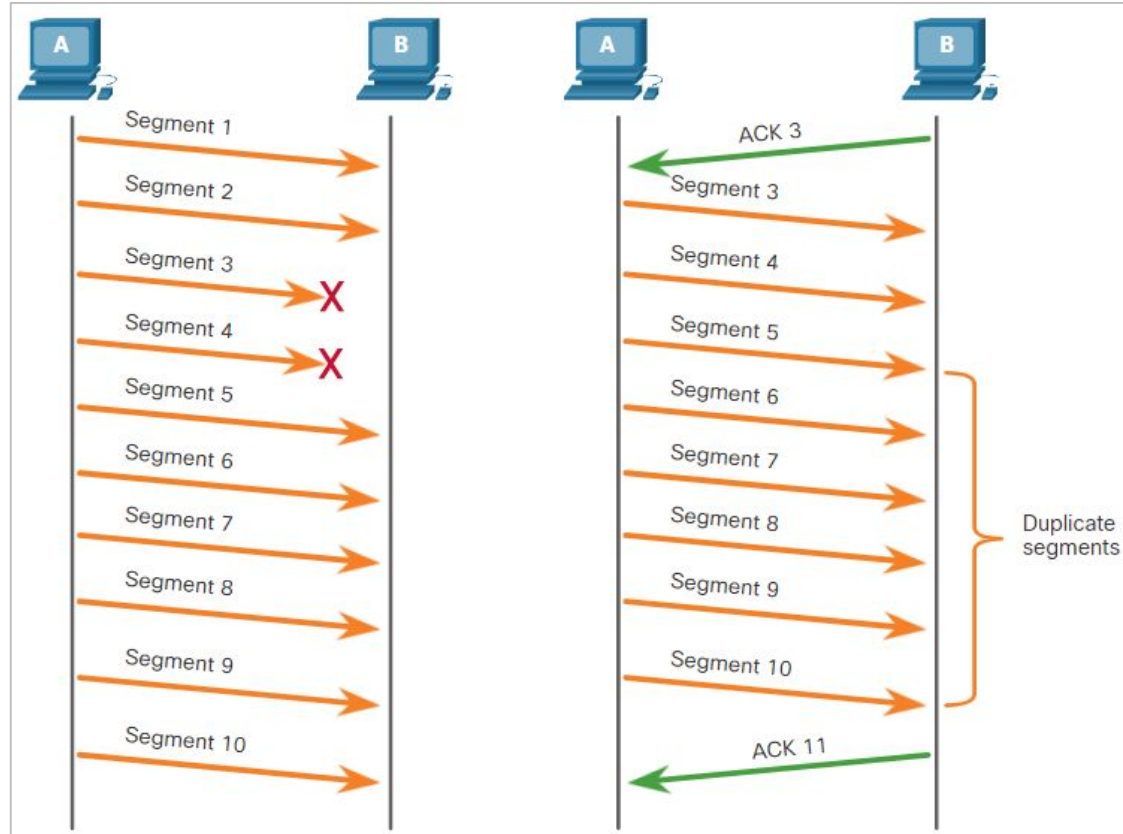This video depicts a simplified example of the TCP operations.

4:06
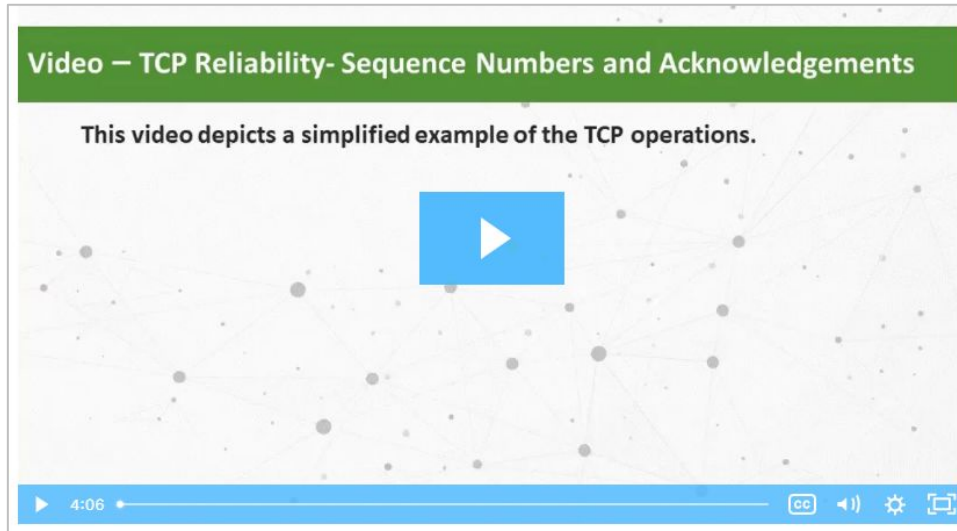
TCP Reliability - Data Loss and Retransmission

- TCP provides methods of managing the segment losses by retransmitting the segments for unacknowledged data.

- The sequence (SEQ) number and acknowledgement (ACK) number are used together to confirm receipt of the bytes of data contained in the transmitted segments.

- The SEQ number identifies the first byte of data in the segment being transmitted.

- TCP uses the ACK number sent back to the source to indicate the next byte that the receiver expects to receive. This is called expectational acknowledgement.

- Prior to later enhancements, TCP could only acknowledge the next byte expected.
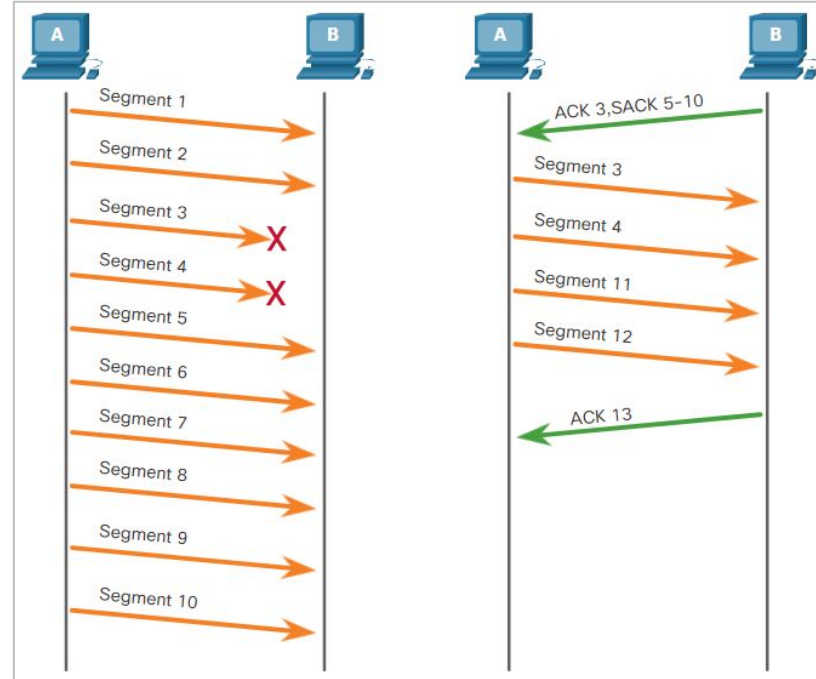
TCP Reliability - Data Loss and Retransmission (Contd.)

- Host operating systems employ an optional TCP feature called selective acknowledgment (SACK), negotiated during the three-way handshake.

- If both hosts support SACK, the receiver can acknowledge which segments (bytes) were received including any discontinuous segments.

- The sending host would only need to retransmit the missing data.

- In the figure, host A sends segments 1 through 10 to host B.

- If all the segments arrive except for segments 3 and 4, host B can acknowledge that it has received segments 1 and 2 (ACK 3), and selectively acknowledge segments 5 through 10 (SACK 5-10). Host A would only need to resend segments 3 and 4.

Video - TCP Reliability - Data Loss and Retransmission

Click Play in the figure to view a lesson on TCP retransmission.

Video — TCP Reliability- Data Loss and Retransmission

This video shows the process of resending segments that are not initially received by the destination.

3:24
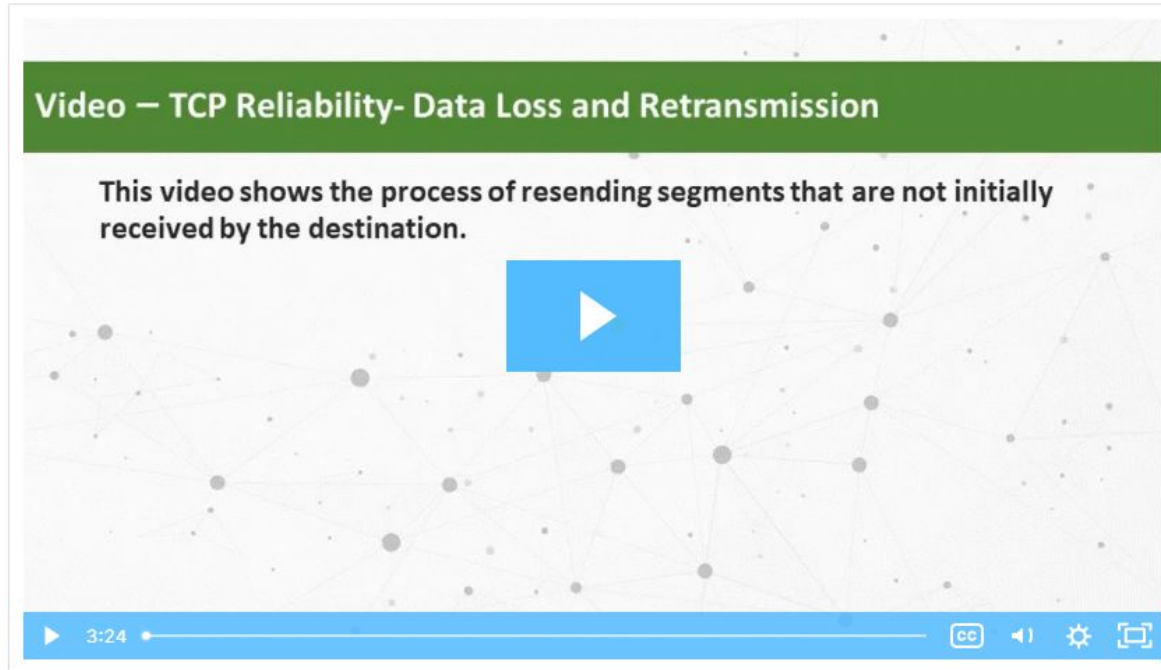
TCP Flow Control - Window Size and Acknowledgments

- TCP also provides mechanisms for flow control. Flow control is the amount of data that the destination can receive and process reliably.

- Flow control helps maintain the reliability of TCP transmission by adjusting the rate of data flow between source and destination for a given session.

- To accomplish this, the TCP header includes a 16-bit field called the window size.

- The window size that determines the number of bytes that can be sent before expecting an acknowledgment.

- The acknowledgment number is the number of the next expected byte.

- The window size is the number of bytes that the destination device of a TCP session can accept and process at one time.

TCP Flow Control - Window Size and Acknowledgments (Contd.)

- The figure shows an example of window size and acknowledgments.

- The window size is included in every TCP segment so the destination can modify the window size at any time depending on buffer availability.

- The initial window size is agreed upon when the TCP session is established during the three-way handshake.

- The source device must limit the number of bytes sent to the destination device based on the window size of the destination. Only after the source receives an acknowledgment, it can continue sending more data for the session.
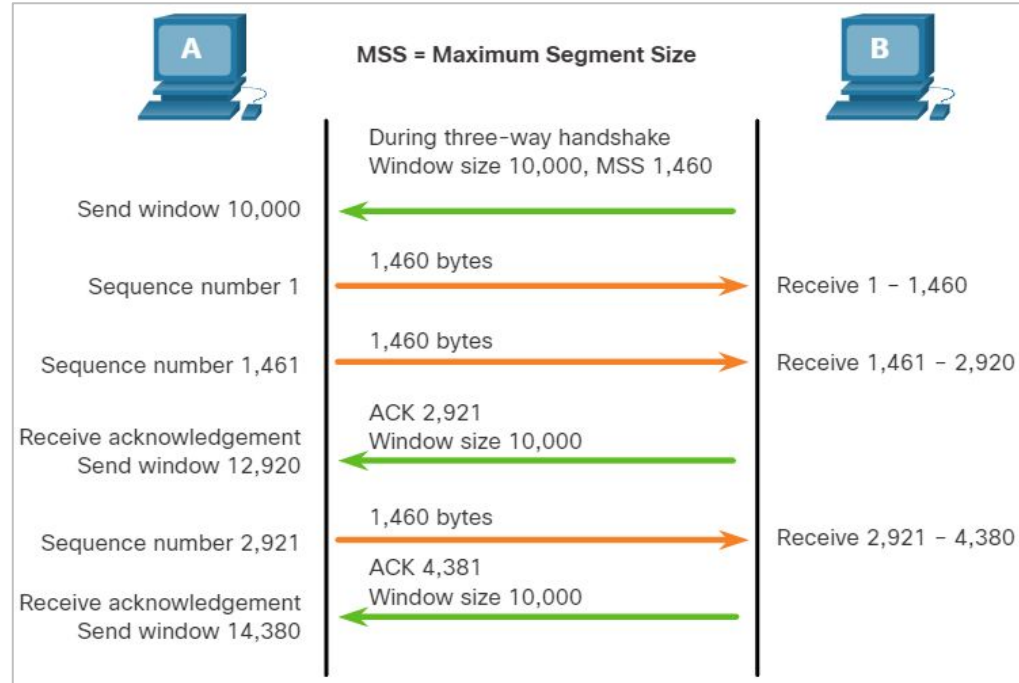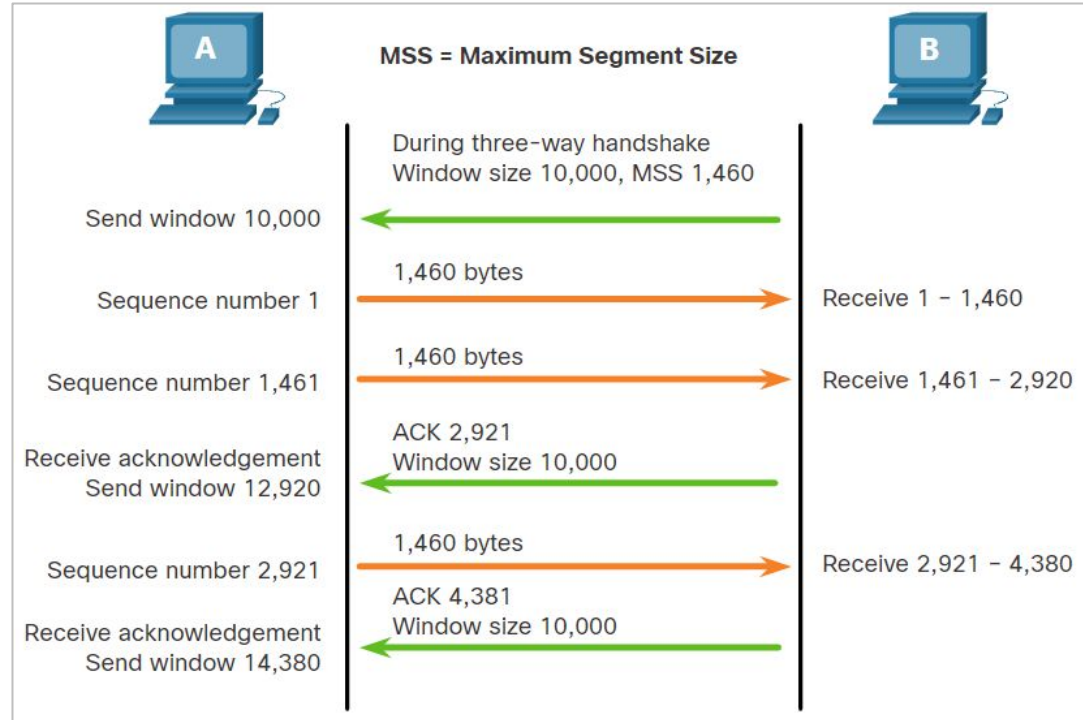
TCP Flow Control - Window Size and Acknowledgments (Contd.)

- The destination will not wait for all the bytes for its window size to be received before replying with an acknowledgment.

- As the bytes are received and processed, the destination will send acknowledgments to inform the source that it can continue to send additional bytes.

- A destination sending acknowledgments as it processes bytes received, and the continual adjustment of the source send window, is known as sliding windows.

- If the availability of the destination's buffer space decreases, it may reduce its window size to inform the source to reduce the number of bytes it should send without receiving an acknowledgment.

Note: Devices today use the sliding windows protocol. The receiver sends an acknowledgment after every two segments it receives. The advantage of sliding windows is that it allows the sender to continuously transmit segments, as long as the receiver is acknowledging previous segments.

TCP Flow Control - Maximum Segment Size (MSS)

- In the figure, the source is transmitting 1,460 bytes of data within each TCP segment. This is the Maximum Segment Size (MSS) that the destination device can receive.

- The MSS is part of the options field in the TCP header that specifies the largest amount of data, in bytes, that a device can receive in a single TCP segment.

- The MSS size does not include the TCP header.

- The MSS is included during the three-way handshake.

MSS = Maximum Segment Size

During three-way handshake
Window size 10,000, MSS 1,460

Send window 10,000

1,460 bytes
Sequence number 1   Receive 1 – 1,460

1,460 bytes
Sequence number 1,461   Receive 1,461 – 2,920

ACK 2,921
Window size 10,000
Receive acknowledgement
Send window 12,920

1,460 bytes
Sequence number 2,921   Receive 2,921 – 4,380
ACK 4,381
Window size 10,000
Receive acknowledgement
Send window 14,380

TCP Flow Control - Maximum Segment Size (MSS) (Contd.)

- A common MSS is 1,460 bytes when using IPv4. A host determines the value of its MSS field by subtracting the IP and TCP headers from the Ethernet maximum transmission unit (MTU).

- On an Ethernet interface, the default MTU is 1500 bytes. Subtracting the IPv4 header of 20 bytes and the TCP header of 20 bytes, the default MSS size will be 1460 bytes, as shown in the figure.

		TCP MSS →
	IP MTU	→
	Ethernet MTU	→

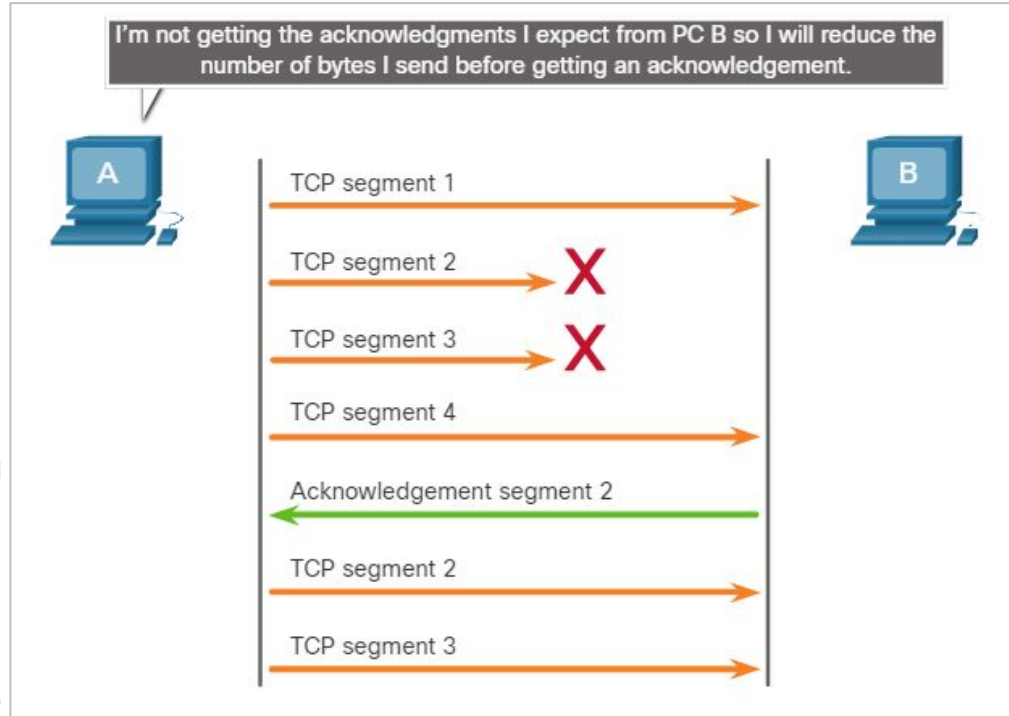| Ethernet	IPv4	TCP	Payload	FCS
	20 bytes	20 bytes	1460 bytes	

1500 bytes
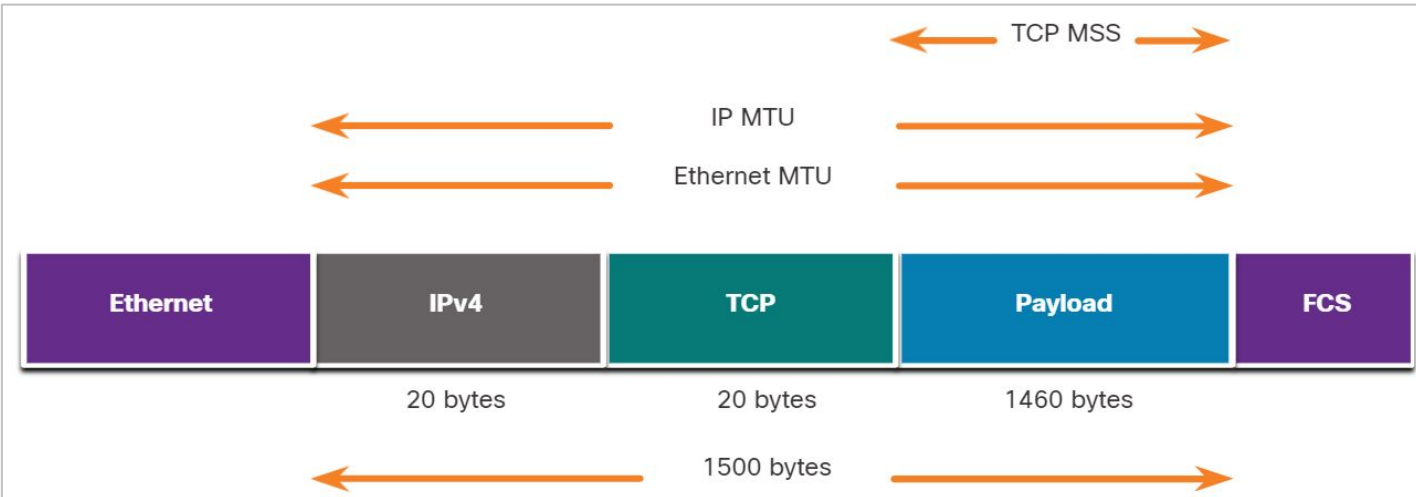
TCP Flow Control - Congestion Avoidance

- When congestion occurs on a network, it results in packets being discarded by the overloaded router.

- When packets containing TCP segments do not reach their destination, they are left unacknowledged.

- By determining the rate at which TCP segments are sent but not acknowledged, the source can assume a certain level of network congestion.

- Whenever there is congestion, retransmission of lost TCP segments from the source will occur.

- If the retransmission is not properly controlled, the additional retransmission of the TCP segments can make the congestion even worse.

- Not only are new packets with TCP segments introduced into the network, but the feedback effect of the retransmitted TCP segments that were lost will also add to the congestion.

- To avoid and control congestion, TCP employs several congestion handling mechanisms, timers, and algorithms.

TCP Flow Control - Congestion Avoidance (Contd.)

- If the source determines that the TCP segments are either not being acknowledged or not acknowledged in a timely manner, then it can reduce the number of bytes it sends before receiving an acknowledgment.

- As shown in the figure, PC A senses there is congestion and therefore, reduces the number of bytes it sends before receiving an acknowledgment from PC B.

- Acknowledgment numbers are for the next expected byte and not for a segment. The segment numbers used are simplified for illustration purposes.

I'm not getting the acknowledgments I expect from PC B so I will reduce the number of bytes I send before getting an acknowledgement.

TCP segment 1
TCP segment 2 X
TCP segment 3 X
TCP segment 4
Acknowledgement segment 2
TCP segment 2
TCP segment 3

Lab – Exploring Nmap

- Port scanning is usually part of a reconnaissance attack.

- There are a variety of port scanning methods that can be used.

- We will explore how to use the Nmap utility. Nmap is a powerful network utility that is used for network discovery and security auditing.

9.4 The Transport Layer Summary

What Did I Learn in this Module?

- The transport layer is the link between the application layer and the lower layers of the OSI model that are responsible for network transmission.

- The transport layer includes TCP and UDP. Transport layer protocols specify how to transfer messages between hosts and is responsible for managing reliability requirements of a conversation.

- The transport layer is responsible for tracking conversations (sessions), segmenting data and reassembling segments, adding segment header information, identifying applications, and conversation multiplexing.

- TCP is stateful and reliable. It acknowledges data, resends lost data, and delivers data in sequenced order. TCP is used for email and the web.

- UDP is stateless and fast. It has low overhead, does not requires acknowledgments, does not resend lost data, and processes data in the order in which it arrives. UDP is used for VoIP and DNS.

What Did I Learn in this Module? (Contd.)

- The TCP and UDP transport layer protocols use port numbers to manage multiple simultaneous conversations. This is why the TCP and UDP header fields identify a source and destination application port number.

- The three-way handshake establishes that the destination device is present on the network. It verifies that the destination device has an active service that is accepting requests on the destination port number that the initiating client intends to use.

- The six control bits flags are: URG, ACK, PSH, RST, SYN, and FIN and are used to identify the function of TCP messages that are sent.

- For the original message to be understood by the recipient, all the data must be received and the data in these segments must be reassembled into the original order.

- Host operating systems today typically employ an optional TCP feature called selective acknowledgment (SACK), which is negotiated during the three-way handshake.

- Flow control helps maintain the reliability of TCP transmission by adjusting the rate of data flow between source and destination.

New Terms and Commands

• Transmission Control Protocol (TCP)	• URG, ACK, PSH, RST, SYN, FIN
• User Datagram Protocol (UDP)	• Selective Acknowledgment (SACK)
• Socket Pairs	• Maximum Segment Size (MSS)
• Initial Sequence Number (ISN)	• Ethernet Maximum Transmission Unit (MTU)