Elk Technologies
https://elktech.org
SCOTT STEPHENSON

MODULE 14: TRANSPORT LAYER

Introduction to Networks

Cisco | Networking Academy®
Mind Wide Open™

CCNAv7

Module Objectives

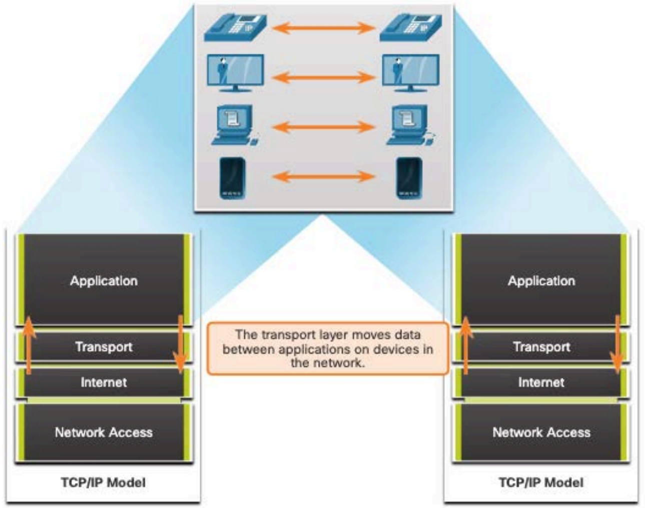
- Module Title: Transport Layer
- Module Objective: Compare the operations of transport layer protocols in supporting end-to-end communication.

Topic Title	Topic Objective
14.1 Transportation of Data	Explain the purpose of the transport layer in managing the transportation of data in end-to-end communication.
14.2 TCP Overview	Explain characteristics of TCP.
14.3 UDP Overview	Explain characteristics of UDP.
14.4 Port Numbers	Explain how TCP and UDP use port numbers.
14.5 TCP Communication Process	Explain how TCP session establishment and termination processes facilitate reliable communication.
14.6 Reliability and Flow Control	Explain how TCP protocol data units are transmitted and acknowledged to guarantee delivery.
14.7 UDP Communication	Compare the operations of transport layer protocols in supporting end-to-end communication.

14.1 Transportation of Data

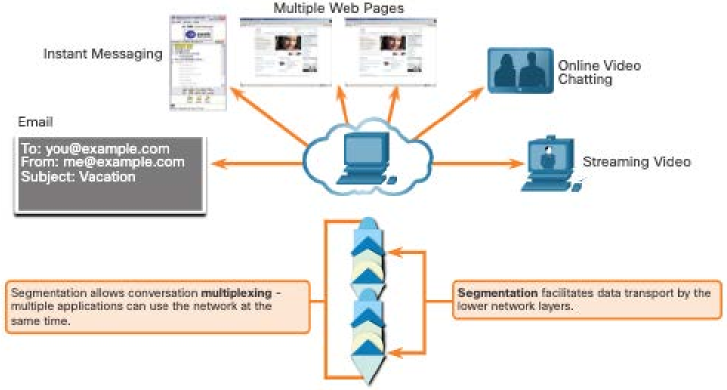
Role of the Transport Layer

- The Transport Layer is responsible for:
  - Establishing a temporary communication session between two applications and delivering data between them.
  - Logical communications between applications running on different hosts.
  - The link between the application layer and the lower layers that are responsible for network transmission.
  - Provides Connection-oriented data stream support, reliability, flow control, and multiplexing.
- TCP/IP uses two protocols to achieve this:
  - **Transmission Control Protocol** (TCP)
  - **User Datagram Protocol** (UDP)



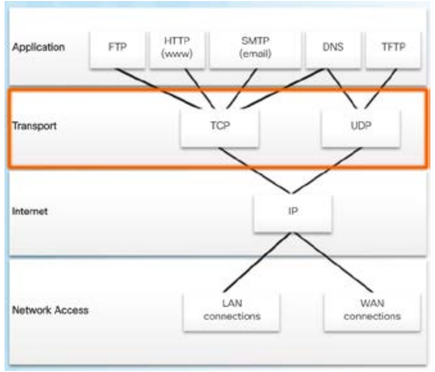
Transport Layer Responsibilities

- The transport layer has the following responsibilities:
  - Tracking the reliability of individual communication between applications on the source and destination hosts.
  - Segmenting data for manageability and reassembling segmented data into streams of application data at the destination.
  - Adding header information.
  - Identify, separating, and managing the proper application for each communication stream.
  - Using segmentation and multiplexing to enable different communication conversations to be interleaved on the same network.
  - Ensuring the correct information is delivered to the correct application.


Multiple Web Pages
Instant Messaging
Online Video Chatting
Email
To: you@example.com
From: me@example.com
Subject: Vacation
Streaming Video
Segmentation allows conversation **multiplexing** - multiple applications can use the network at the same time.
Segmentation facilitates data transport by the lower network layers.

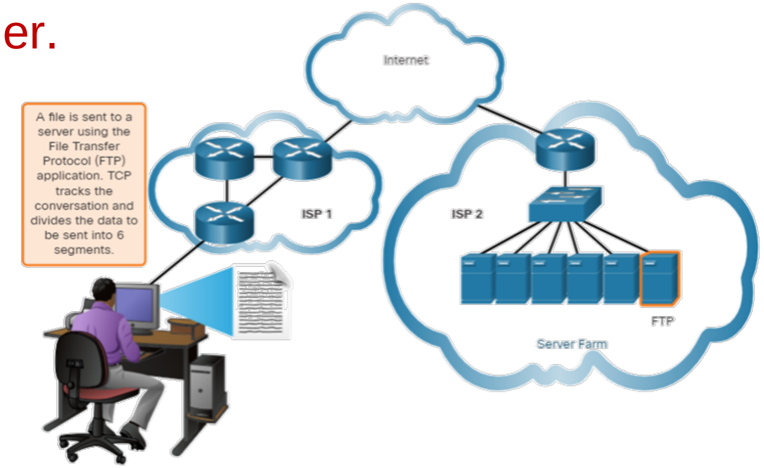
Transport Layer Protocols

- IP does not specify how the delivery or transportation of the packets takes place.
- Transport layer protocols specify how to transfer messages between hosts, and are responsible for managing reliability requirements of a conversation.
- Different applications have different transport reliability requirements
- TCP/IP provides two transport layer protocols:
  - **Transmission Control Protocol** (TCP)
  - **User Datagram Protocol** (UDP)
    - Provides just the basic functions for delivery – no reliability (datagrams are just sent).
    - Very little overhead and data checking – faster than TCP.
    - Fewer delays in transmission.



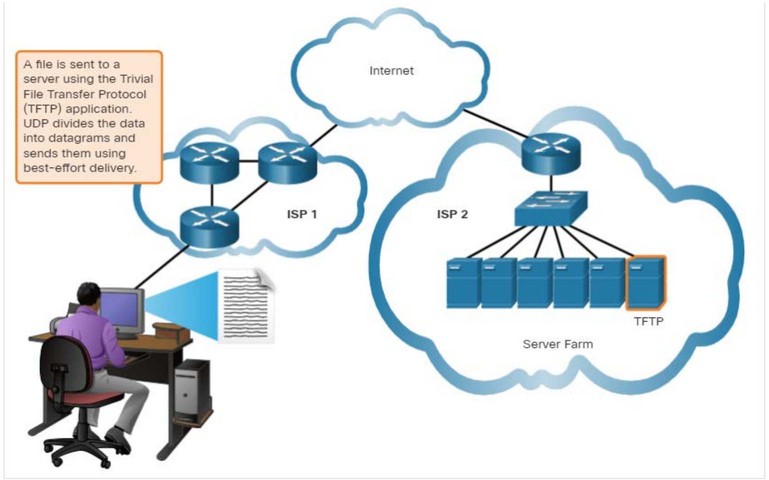
Transmission Control Protocol

- TCP provides reliability and flow control.
- TCP 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.
  - Provides reliable delivery ensuring that all of the data arrives at the destination.


Internet
A file is sent to a server using the File Transfer Protocol (FTP) application. TCP tracks the conversation and divides the data to be sent into 6 segments.
ISP 1
ISP 2
FTP
Server Farm

User Datagram Protocol (UDP)

- UDP provides the basic functions for delivering datagrams between the appropriate applications, with very little overhead and data checking.
  - UDP is a connectionless protocol.
    - Faster delivery mechanism.
  - UDP is known as a best-effort delivery protocol because there is no acknowledgment that the data is received at the destination (no guaranteed delivery).
    - Applications that handle reliability themselves.
    - Applications that can tolerate some data loss, but require little or no delay.


A file is sent to a server using the Trivial File Transfer Protocol (TFTP) application. UDP divides the data into datagrams and sends them using best-effort delivery.

The Right Transport Layer Protocol for the Right Application

- TCP or UDP?
  - There is a trade-off between the value of reliability and the burden it places on the network.
  - Application developers choose the transport protocol based on the requirements of their applications.
  - TCP is better for databases, web browsers, email clients, etc.
    - Require that all data that is sent arrives at the destination in its original condition.
  - UDP is better for live audio or video streaming, VoIP, etc.
    - If one or two segments fail to arrive, if disruption in the stream, they may not be noticeable to the user.
    - Used by request-and-reply applications where the data is minimal, and retransmission can be done quickly.
- **Port numbers** - used by both TCP and UDP to differentiate between applications and communication streams.
- **Checksum** - checked to give assurance that data is not corrupted.

TCP required protocol properties:
- Reliable
- Acknowledges data
- Resend lost data
- Delivers data in sequenced order

UDP required protocol properties:
- Fast
- Low overhead
- Does not require acknowledgements
- Does not resend lost data
- Delivers data as it arrives

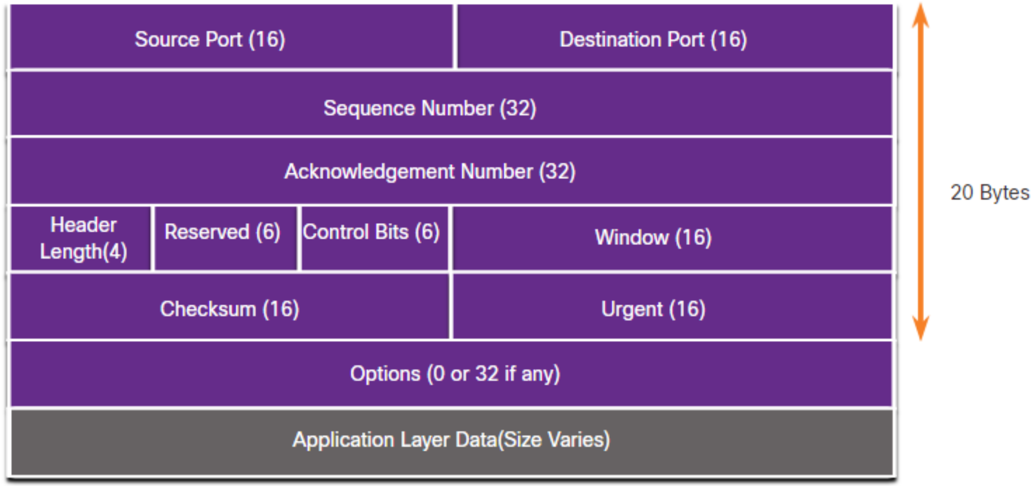
14.2 TCP Overview

TCP Features

- **Establishes a Session** - TCP is a connection-oriented protocol that negotiates and establishes a permanent connection (or session) between source and destination devices prior to forwarding any traffic.
- **Ensures Reliable Delivery** - For many reasons, it is possible for a segment to become corrupted or lost completely, as it is transmitted over the network. TCP ensures that each segment that is sent by the source arrives at the destination.
- **Provides Same-Order Delivery** - Because networks may provide multiple routes that can have different transmission rates, data can arrive in the wrong order.
- **Supports Flow Control** - Network hosts have limited resources (i.e., memory and processing power). When TCP is aware that these resources are overtaxed, it can request that the sending application reduce the rate of data flow.
- **Acknowledgement** – Acknowledging received data.
- **Stateful protocol** – keeps track of the session.

TCP Header

- TCP is a **stateful** protocol which means it keeps track of the state of the communication session.
- TCP records which information it has sent, and which information has been acknowledged.
- Adds 20 bytes of overhead in the segment header.

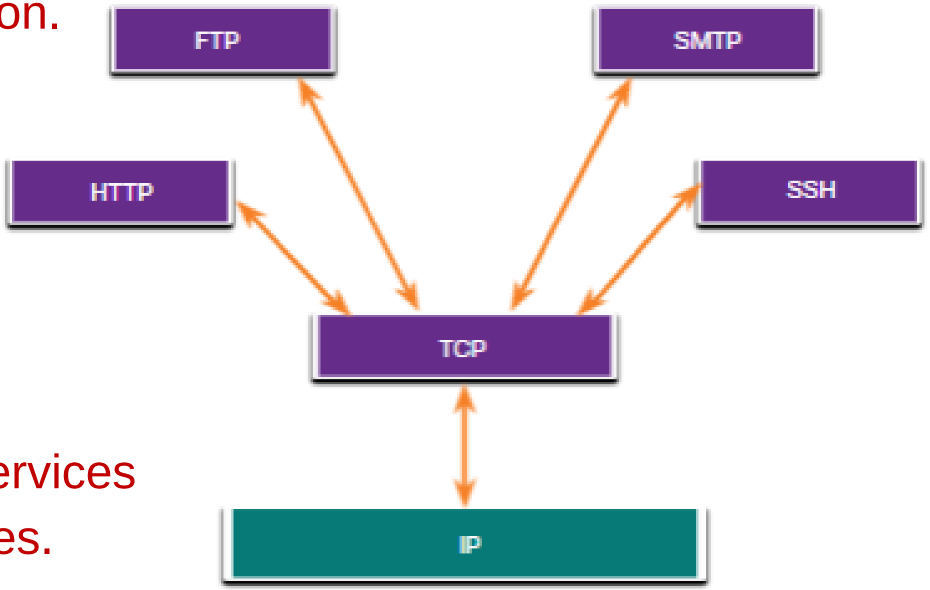


TCP Header Fields

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 used 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.

Applications that Use TCP

- TCP handles all tasks associated with dividing the data stream into segments, providing reliability, controlling data flow, and reordering segments.
- Application that use TCP:
  - **DNS** – server-to-server communication.
  - **HTTP** – web services.
  - **HTTPS** – secure web services.
  - **FTP** – data transfer services.
  - **SMTP** – mail transfer services.
  - **POP3** – mail services.
  - **IMAP** – mail services.
  - **Telnet** – unsecure communication services
  - **SSH** – secure communication services.
  - **RDP** – remote desktop services.



14.3 UDP Overview

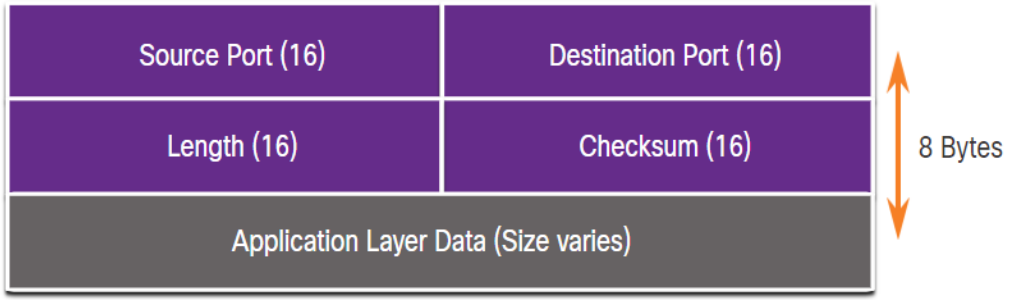
UDP Features

- Use UDP for less overhead and to reduce possible delays .
- Similar to a non-registered letter.
- RFC 768.
- The pieces of communication in UDP are called **Datagrams.**
- Features and responsibilities:
  - **Connectionless** – Best-effort delivery (just sends the datagram).
  - **Unreliable delivery** - Reliability must be handled by the application.
  - **No acknowledgment** – Lost segments are not resent.
  - **No ordered data reconstruction** – Data is reconstructed in the order that it is received.
  - **No flow control** – Does not inform the sender about resource availability.
  - **Stateless protocol** – No session establishment (minimal delay).

UDP Header

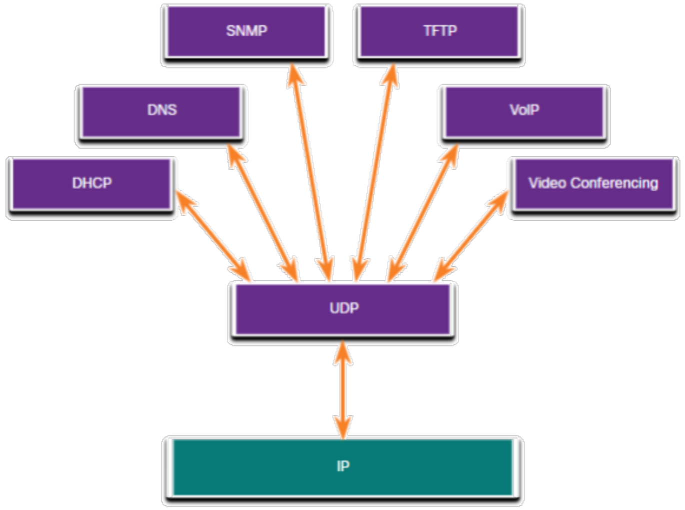
- The UDP header is far simpler than the TCP header because it only has four fields and requires 8 bytes (i.e. 64 bits).

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.
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.



Applications that use UDP

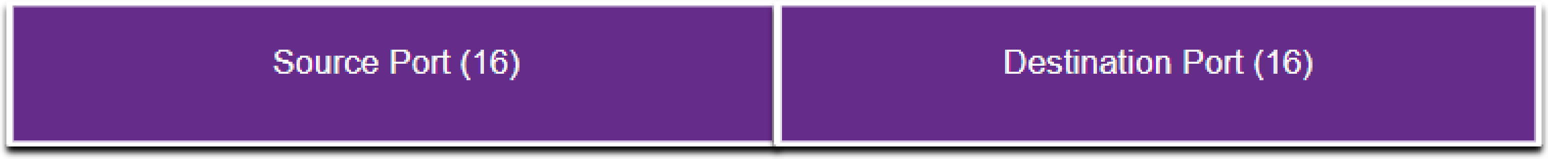
- Live video and multimedia applications - These applications can tolerate some data loss but require little or no delay. Examples include VoIP and live streaming video.
- Simple request and reply applications - Applications with simple transactions where a host sends a request and may or may not receive a reply. Examples include DNS and DHCP.
  - Used for a client-to-server communication.
- Applications that handle reliability themselves - Unidirectional communications where flow control, error detection, acknowledgments, and error recovery is not required, or can be handled by the application. Examples include SNMP and TFTP.



14.4 Port Numbers

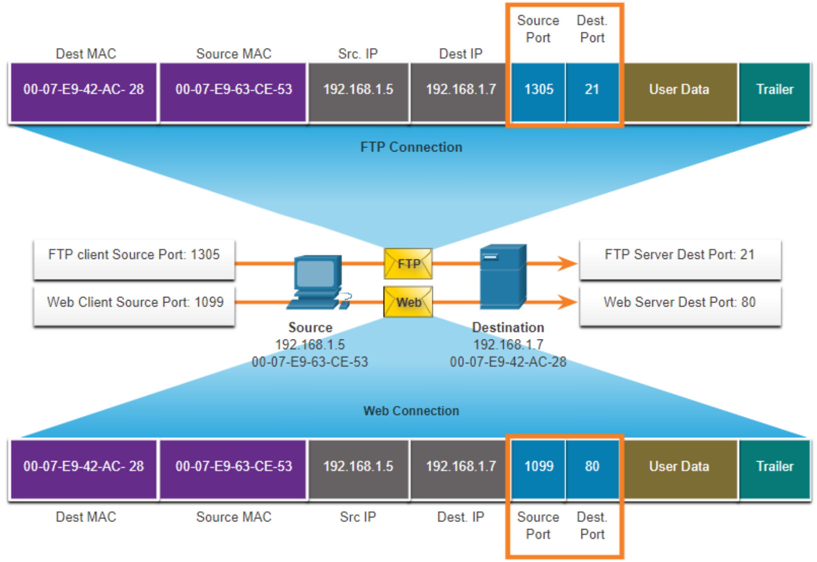
Multiple Separate Communications

- TCP and UDP transport layer protocols use port numbers to manage multiple, simultaneous conversations.
- The source port number is associated with the originating application on the local host whereas the destination port number is associated with the destination application on the remote host.
  - If multiple conversations occur that are using the same service, the source port number is used to track the separate conversations.



Socket Pairs

- The source and destination ports are placed within the segment.
- The segments are then encapsulated within an IP packet.
- 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.
- Source port acts as a return address.
- Two sockets combine to form a **socket pair**: (192.168.1.5:1099, 192.168.1.7:80)



Port Number Groups

Port Group	Number Range	Description
Well-known Ports	0 to 1,023	•These port numbers are reserved for common or popular services and applications such as web browsers, email clients, and remote access clients. •Defined well-known ports for common server applications enables clients to easily identify the associated service required.
Registered Ports	1,024 to 49,151	•These port numbers are assigned by IANA to a requesting entity to use with specific processes or applications. •These processes are primarily individual applications that a user has chosen to install, rather than common applications that would receive a well-known port number. •For example, Cisco has registered port 1812 for its RADIUS server authentication process.
Private and/ or Dynamic Ports	49,152 to 65,535	•These ports are also known as ephemeral ports. •The client's OS usually assign port numbers dynamically when a connection to a service is initiated. •The dynamic port is then used to identify the client application during communication.

Port Number Groups

- Important Well-Known Port Numbers:

Port Number	Protocol	Application
20	TCP	File Transfer Protocol (FTP) - Data
21	TCP	File Transfer Protocol (FTP) - Control
22	TCP	Secure Shell (SSH)
23	TCP	Telnet
25	TCP	Simple Mail Transfer Protocol (SMTP)
53	UDP, TCP	Domain Name Service (DNS)
67	UDP	Dynamic Host Configuration Protocol (DHCP) - Server
68	UDP	Dynamic Host Configuration Protocol - Client
69	UDP	Trivial File Transfer Protocol (TFTP)
80	TCP	Hypertext Transfer Protocol (HTTP)
110	TCP	Post Office Protocol version 3 (POP3)
143	TCP	Internet Message Access Protocol (IMAP)
161	UDP	Simple Network Management Protocol (SNMP)
443	TCP	Hypertext Transfer Protocol Secure (HTTPS)

The netstat Command

- Unexplained TCP connections can pose a major security threat.
- **`netstat`** is an important tool to verify connections.

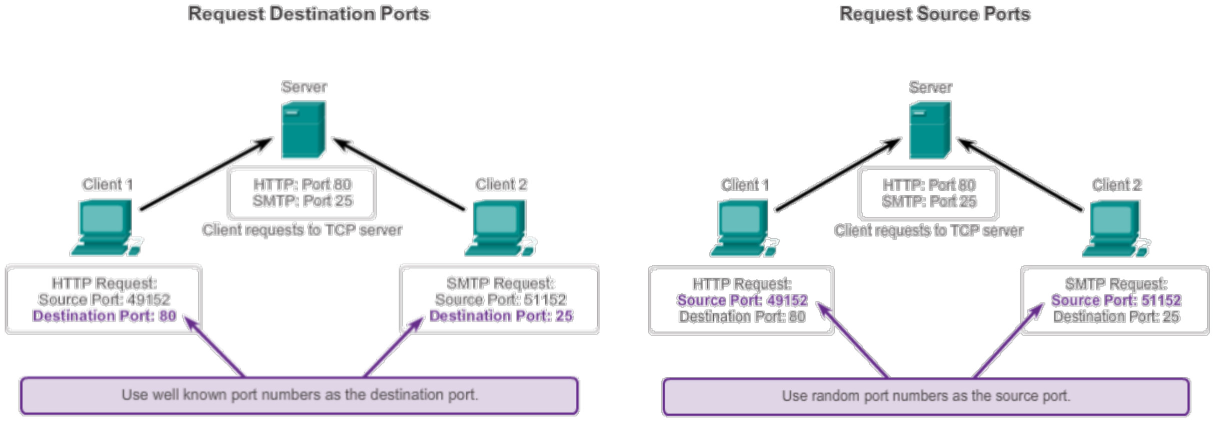
```
C:\> netstat
Active Connections
Proto Local Address            Foreign Address                State
TCP   192.168.1.124:3126       192.168.0.2:netbios-ssn        ESTABLISHED
TCP   192.168.1.124:3158       207.138.126.152:http           ESTABLISHED
TCP   192.168.1.124:3159       207.138.126.169:http           ESTABLISHED
TCP   192.168.1.124:3160       207.138.126.169:http           ESTABLISHED
TCP   192.168.1.124:3161       sc.msn.com:http                ESTABLISHED
TCP   192.168.1.124:3166       www.cisco.com:http             ESTABLISHED
```

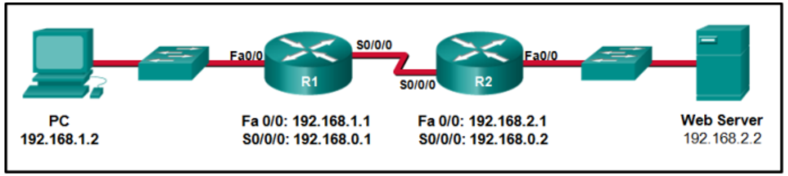
TCP Client and Server Processes

- PC is requesting a page for the Web Server. What is the source and destination sockets?

Source - 192.168.1.2:[1024-65533]

Destination - 192.168.2.2:80

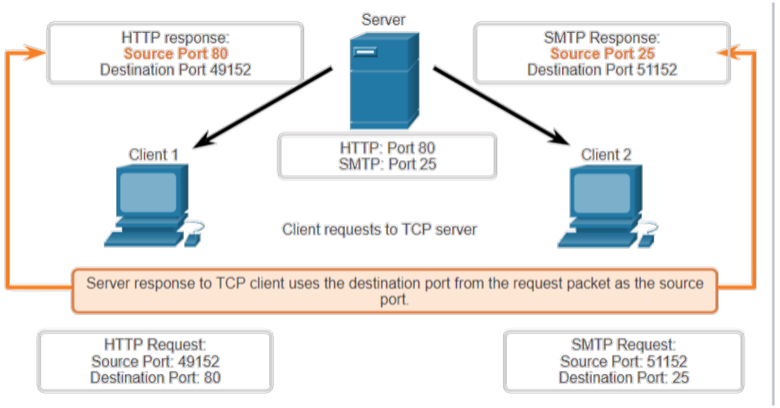




14.5 TCP Communication Process

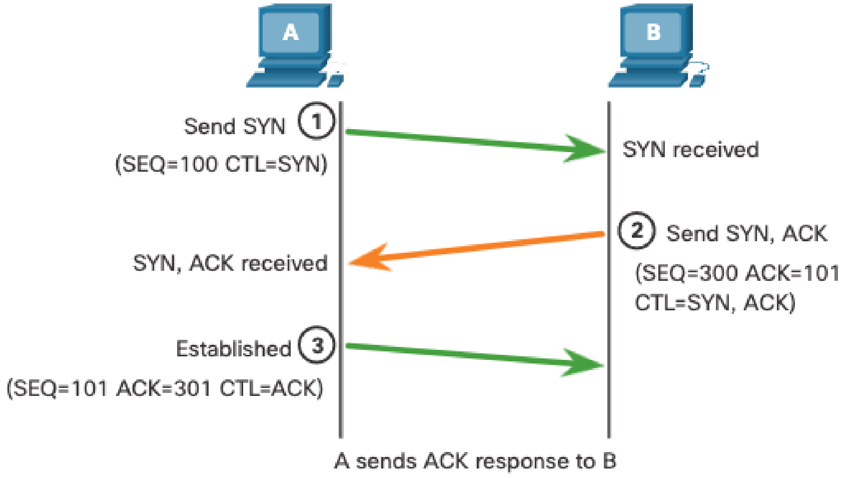
TCP Server Processes

- Each application process running on a server is configured to use a port number.
- An individual server cannot have two services assigned to the same port number within the same transport layer services.
- 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.


Server
HTTP response: Source Port 80 Destination Port 49152
SMTP Response: Source Port 25 Destination Port 51152
Client 1
Client 2
HTTP: Port 80 SMTP: Port 25
Client requests to TCP server
Server response to TCP client uses the destination port from the request packet as the source port.
HTTP Request: Source Port: 49152 Destination Port: 80
SMTP Request: Source Port: 51152 Destination Port: 25

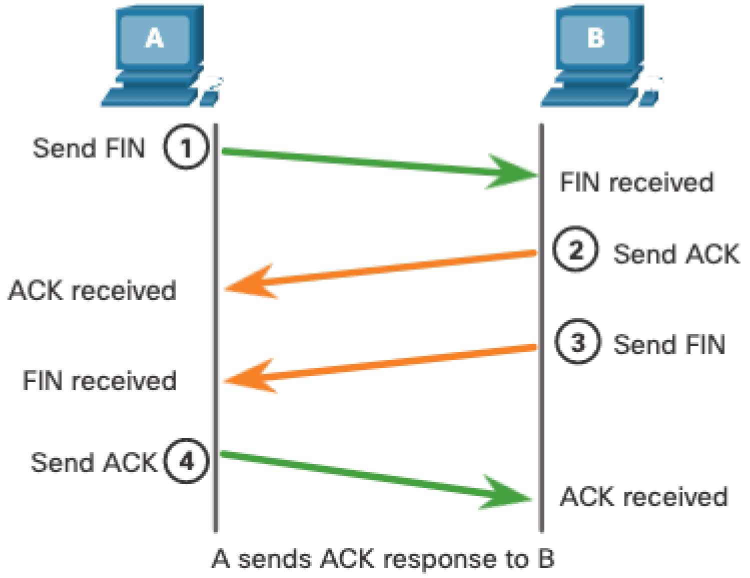
TCP Connection Establishment

- A TCP connection is established with the three-way handshake:
  - Step 1: The initiating client requests a client-to-server communication session with the server(**SYN**) .
  - Step 2: The server acknowledges the client-to-server communication session and requests a server-to-client communication session (**SYN ACK**).
  - Step 3: The initiating client acknowledges the server-to-client communication session (**ACK**).



Session Termination

- The FIN TCP flag is used to terminate a TCP connection:
  - Step 1: When the client has no more data to send in the stream, it sends a segment with the **FIN** flag set.
  - Step 2: The server sends an ***ACK*** to acknowledge the receipt of the FIN to terminate the session from client to server.
  - Step 3: The server sends a **FIN** to the client to terminate the server-to-client session.
  - Step 4: The client responds with an **ACK** to acknowledge the FIN from the server.

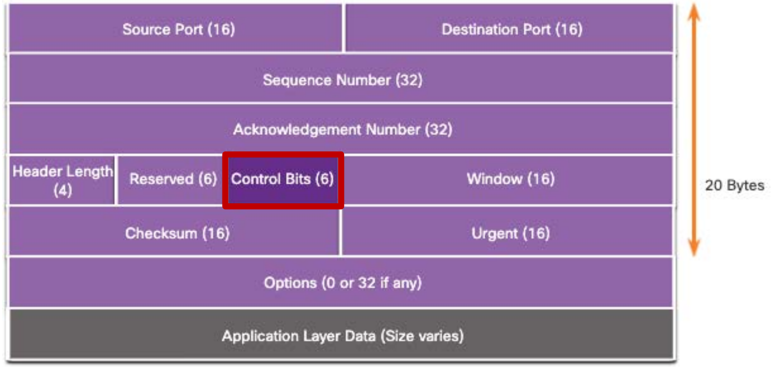


TCP Three-Way Handshake Analysis

- Functions of the Three-Way Handshake:
  - 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

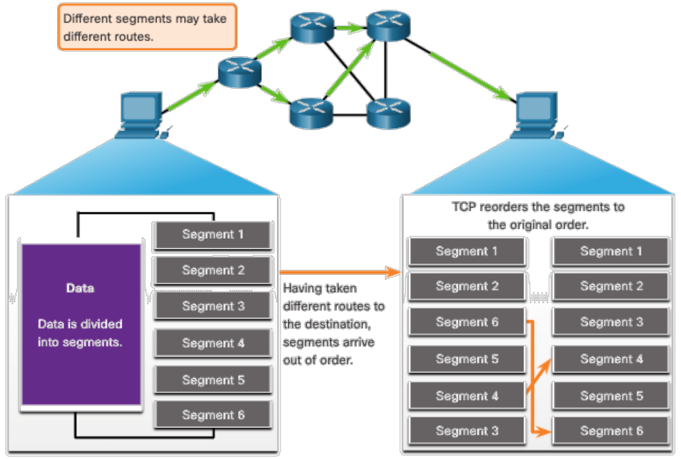
- The six control bit flags are as follows:
  - **URG** - Urgent pointer field significant
  - **ACK** - Acknowledgment flag used in connection establishment and session termination
  - **PSH** - Push function
  - **RST** - Reset the connection when an error or timeout occurs
  - **SYN** - Synchronize sequence numbers used in connection establishment
  - **FIN** - No more data from sender and used in session termination
- ACK and SYN are used in the three-way handshake.



14.6 Reliability and Flow Control

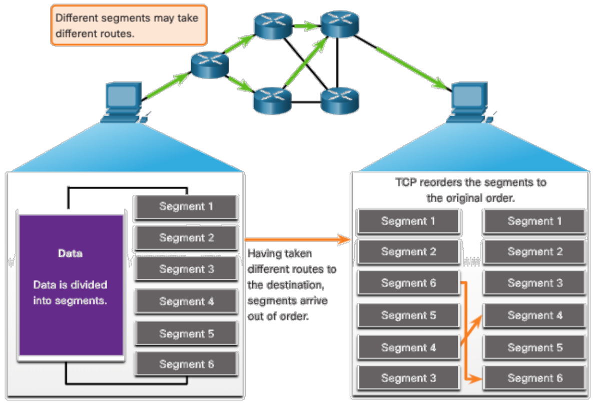
TCP Reliability

- There may be times when TCP segments do not arrive at their destination or arrive out of order.
- Segments received out of order are held for later processing.
- The data is delivered to the application layer only when it has been completely received and reassembled.
- TCP segments are reordered at the destination.


Different segments may take different routes.
Data
Data is divided into segments.
Segment 1
Segment 2
Segment 3
Segment 4
Segment 5
Segment 6
Having taken different routes to the destination, segments arrive out of order.
TCP reorders the segments to the original order.
Segment 1
Segment 2
Segment 6
Segment 5
Segment 4
Segment 3
Segment 1
Segment 2
Segment 3
Segment 4
Segment 5
Segment 6

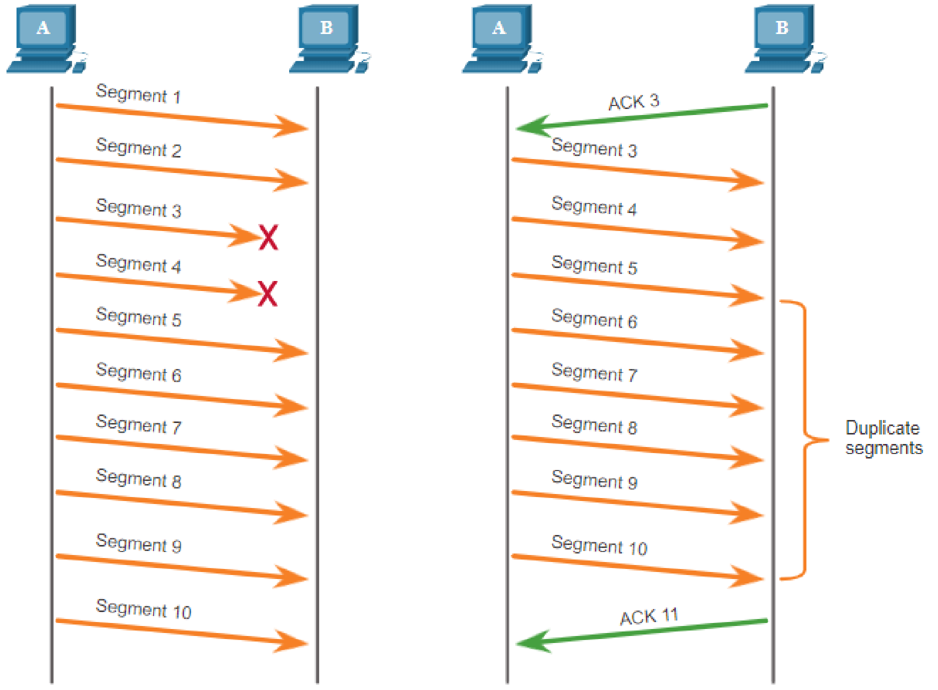
TCP Reliability- Guaranteed and Ordered Delivery

- Sequence numbers are assigned in the header of each packet to achieve this goal.
- Sequence numbers used to reassemble segments into original order
- TCP segments use sequence numbers to uniquely identify and acknowledge each segment, keep track of segment order, and indicate how to reassemble and reorder received segments
- An initial sequence number (ISN) is randomly chosen during the TCP session setup. The ISN is then incremented by the number of transmitted bytes.
- The receiving TCP process buffers the segment data until all data is received and reassembled.
- TCP can also help maintain the flow of packets so that devices do not become overloaded.


Different segments may take different routes.
Data
Data is divided into segments.
Segment 1
Segment 2
Segment 3
Segment 4
Segment 5
Segment 6
Having taken different routes to the destination, segments arrive out of order.
TCP reorders the segments to the original order.
Segment 1
Segment 2
Segment 6
Segment 5
Segment 4
Segment 3
Segment 1
Segment 2
Segment 3
Segment 4
Segment 5
Segment 6

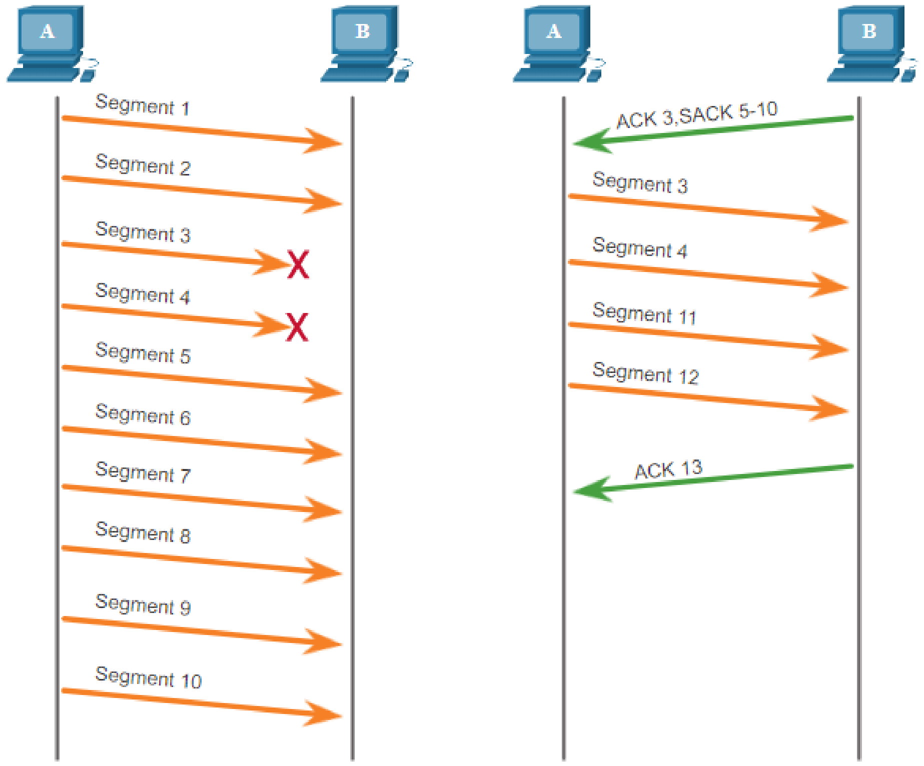
TCP Reliability – Data Loss and Retransmission

- No matter how well designed a network is, data loss occasionally occurs.
- TCP provides methods of managing these segment losses. Among these is a mechanism to retransmit segments for unacknowledged data.



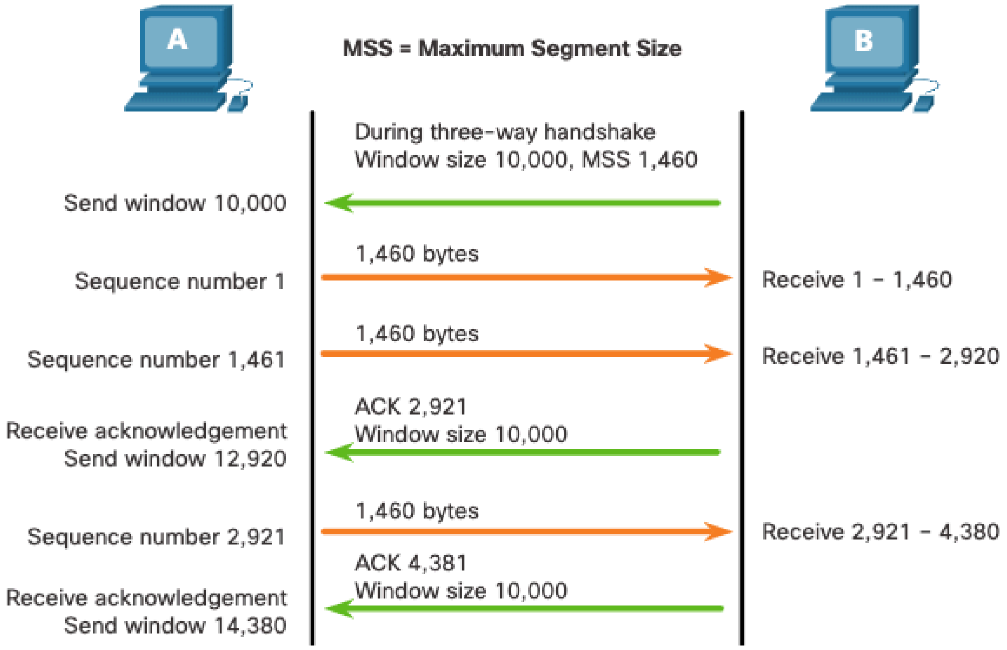
TCP Reliability – Data Loss and Retransmission

- Host operating systems today typically employ an optional TCP feature called **selective acknowledgment** (SACK), negotiated during the three-way handshake.
- If both hosts support SACK, the receiver can explicitly acknowledge which segments (bytes) were received including any discontinuous segments.



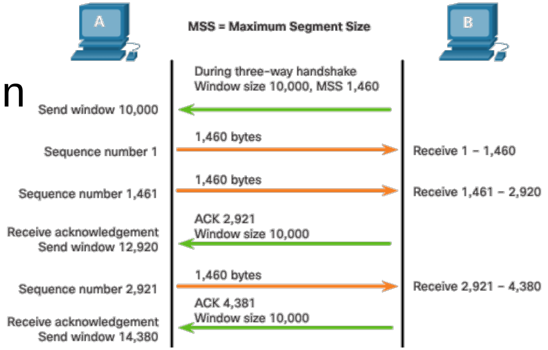
TCP Flow Control – Window Size and Acknowledgments

- TCP also provides mechanisms for flow control as follows:
  - 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.
  - TCP flow control function relies on a 16-bit TCP header field called the **Window size**.



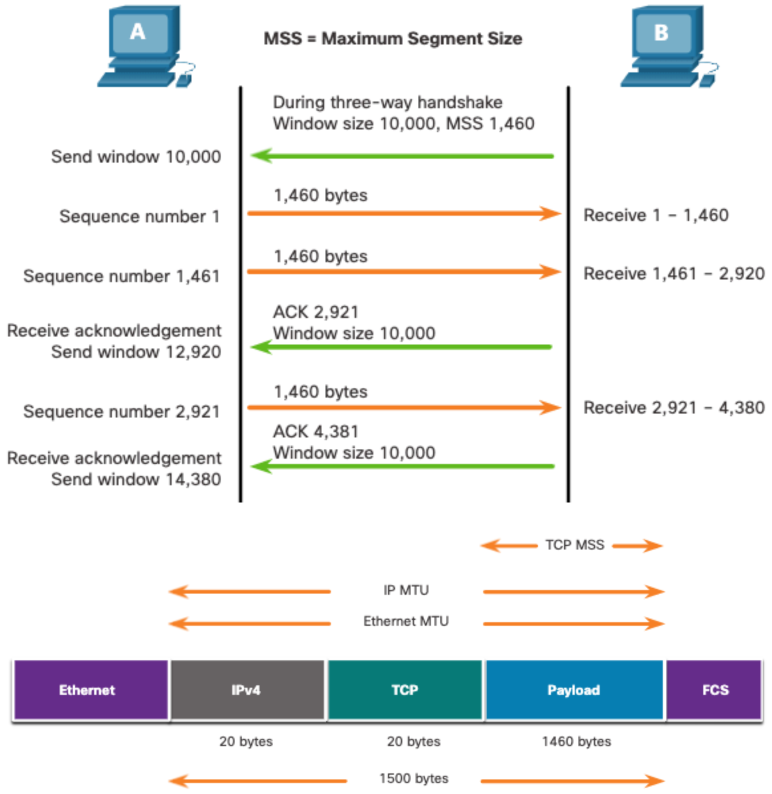
TCP Flow Control – Window Size and Acknowledgments

- **Window Size** - The amount of data or the number of bytes the destination device of a TCP session can accept and process at one time
- The TCP source and destination agree on the initial window size when the TCP session is established
- TCP endpoints can adjust the window size during a session if necessary
- The sequence number and acknowledgement number are used together to confirm receipt.
- The acknowledgment number represents the next byte that the destination expects to receive.
  - In the figure, the source is transmitting 1,460 bytes of data within each segment.
  - Window size agreed on during 3-way handshake.
  - Typically, PC B will not wait for 10,000 bytes before sending an acknowledgment.
  - PC A can adjust its send window as it receives acknowledgments from PC B.



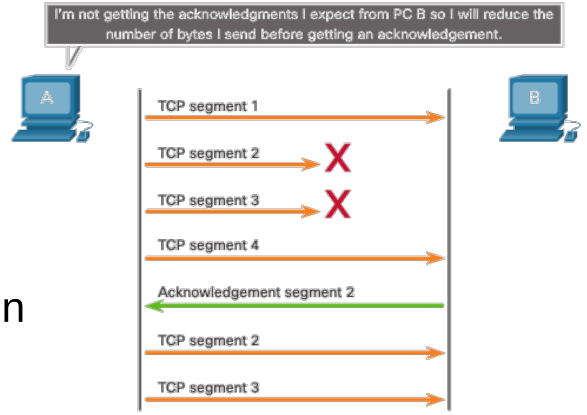
TCP Flow Control – Maximum Segment Size

- **Maximum Segment Size** (MSS) is the maximum amount of data that the destination device can receive.
- 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), which is 1500 bytes be default.
  - 1500 minus 60 (20 bytes for the IPv4 header and 20 bytes for the TCP header) leaves 1460 bytes.



TCP Flow Control – Congestion Avoidance

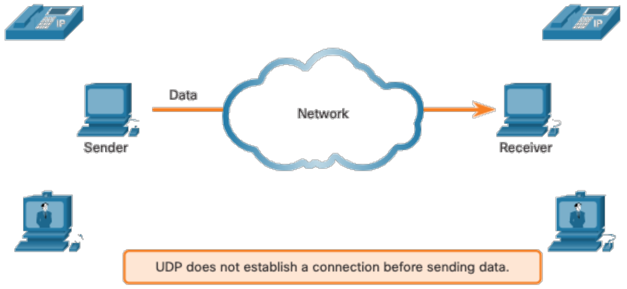
- When congestion occurs on a network, it results in packets being discarded by the overloaded router.
- To avoid and control congestion, TCP employs several congestion handling mechanisms, timers, and algorithms.
  - Undelivered TCP segments trigger re-transmission (TCP segment retransmission can make the congestion even worse).
  - The source can estimate a certain level of network congestion by looking at the rate at which TCP segments are sent but not acknowledged.
  - The source can reduce the number of bytes it sends before receiving an acknowledgement upon congestion detection.
  - The source reduces the number of unacknowledged bytes it sends and not the window size, which is determined by the destination.
  - The destination is usually unaware of the network congestion and sees no need to suggest a new window size.


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.
A
B
TCP segment 1
TCP segment 2
X
TCP segment 3
X
TCP segment 4
Acknowledgement segment 2
TCP segment 2
TCP segment 3

14.7 UDP Communication

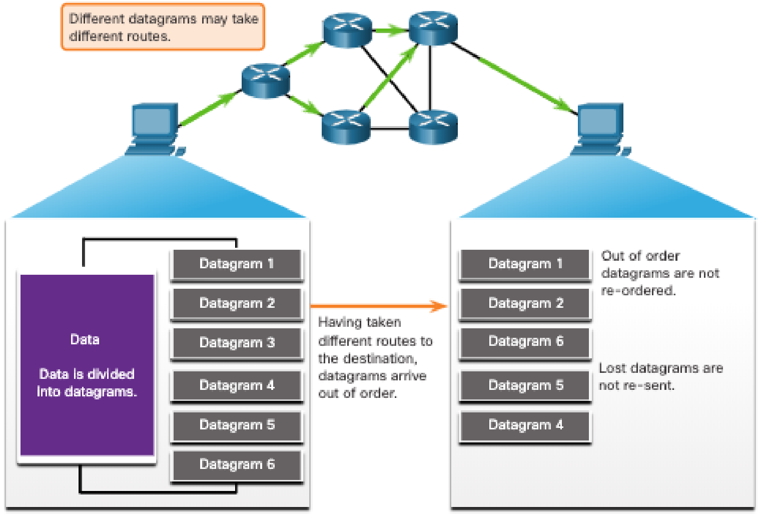
UDP Low Overhead versus Reliability

- UDP just sends datagrams:
  - Is not connection-oriented and does not offer the sophisticated retransmission, sequencing, and flow control mechanisms.
  - Does not establish a connection.
  - Provides low overhead data transport because it has a small datagram header and no network management traffic.
  - Applications running UDP can still use reliability, but it must be implemented in the application layer.
- UDP Server Processes and Requests
  - UDP-based server applications are also assigned well-known or registered port numbers.
  - Requests received on a specific port are forwarded to the proper application based on port numbers.
- The **Remote Authentication Dial-in User Service** (RADIUS) server provides authentication, authorization, and accounting services to manage user access.



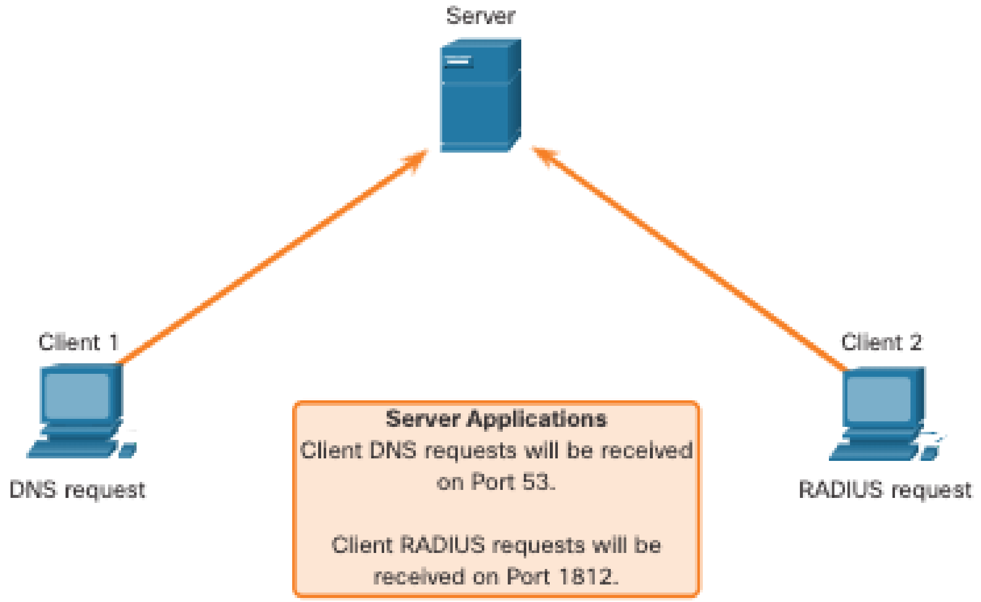
UDP Datagram Reassembly

- UDP does not track sequence numbers the way TCP does.
- UDP has no way to reorder the datagrams into their transmission order.
- UDP simply reassembles the data in the order that it was received and forwards it to the application.
- If necessary, the upper-layer protocols will reassemble in the correct order.


Different datagrams may take different routes.
Data
Data is divided into datagrams.
Datagram 1
Datagram 2
Datagram 3
Datagram 4
Datagram 5
Datagram 6
Having taken different routes to the destination, datagrams arrive out of order.
Datagram 1
Datagram 2
Datagram 6
Datagram 5
Datagram 4
Out of order datagrams are not re-ordered.
Lost datagrams are not re-sent.

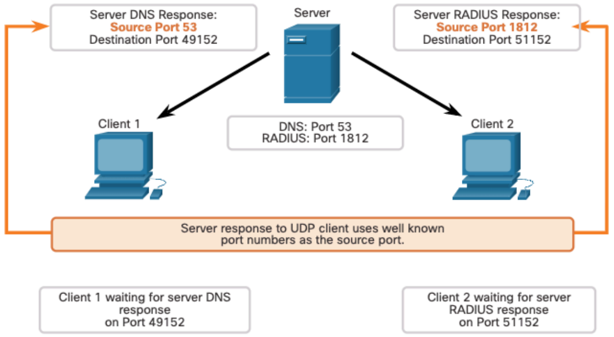
UDP Server Processes and Requests

- UDP-based server applications are assigned well-known or registered port numbers.
- UDP receives a datagram destined for one of these ports, it forwards the application data to the appropriate application based on its port number.


Server
Client 1
Client 2
DNS request
RADIUS request
Server Applications
Client DNS requests will be received on Port 53.
Client RADIUS requests will be received on Port 1812.

UDP Client Processes

- UDP client-server communication is also initiated by a client application.
- The UDP client process dynamically selects a port number from the range of port numbers and uses this as the source port for the conversation.
- The destination port is usually the well-known or registered port number assigned to the server process.
- After a client has selected the source and destination ports, the same pair of ports are used in the header of all datagrams in the transaction.
- Data returning to the client from the server uses a flipped source and destination port numbers in the datagram header.


Server DNS Response:
Source Port 53
Destination Port 49152
Server
Server RADIUS Response:
Source Port 1812
Destination Port 51152
Client 1
DNS: Port 53
RADIUS: Port 1812
Client 2
Server response to UDP client uses well known port numbers as the source port.
Client 1 waiting for server DNS response on Port 49152
Client 2 waiting for server RADIUS response on Port 51152

14.8 Module Practice and Quiz

What did I learn in this module?

- The transport layer is the link between the application layer and the lower layers that are responsible for network transmission.
- The transport layer includes TCP and UDP.
- TCP establishes sessions, ensures reliability, provides same-order delivery, and supports flow control.
- UDP is a simple protocol that provides the basic transport layer functions.
- UDP reconstructs data in the order it is received, lost segments are not resent, no session establishment, and UPD does not inform the sender of resource availability.
- The TCP and UDP transport layer protocols use port numbers to manage multiple simultaneous conversations.
- 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.

What did I learn in this module?

- 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 of each packet.
- Flow control helps maintain the reliability of TCP transmission by adjusting the rate of data flow between source and destination.
- A source might be transmitting 1,460 bytes of data within each TCP segment. This is the typical MSS that a destination device can receive.
- The process of the destination sending acknowledgments as it processes bytes received and the continual adjustment of the source’s send window is known as sliding windows.
- To avoid and control congestion, TCP employs several congestion handling mechanisms.

New Terms and Commands

- Conversation Multiplexing
- Segments
- Datagrams
- Connection-Oriented Protocol
- Connectionless Protocol
- Stateless Protocol
- Flow Control
- Same-Order Delivery
- Socket Pairs
- netstat
- Three-Way Handshake
- SYN
- ACK
- FIN
- URG
- PSH
- RST
- Initial Sequence Number (ISN)
- Selective Acknowledgement (SACK)
- Sliding Window
- Maximum Segment Size (MSS)
- Maximum Transmission Unit (MTU)
- Congestion Avoidance

Elk Technologies

https://elktech.org

Scott Stephenson