Cisco Networking Academy
Mind Wide Open

Chapter 9: Transport Layer

Introduction to Networks v5.1

Chapter Outline

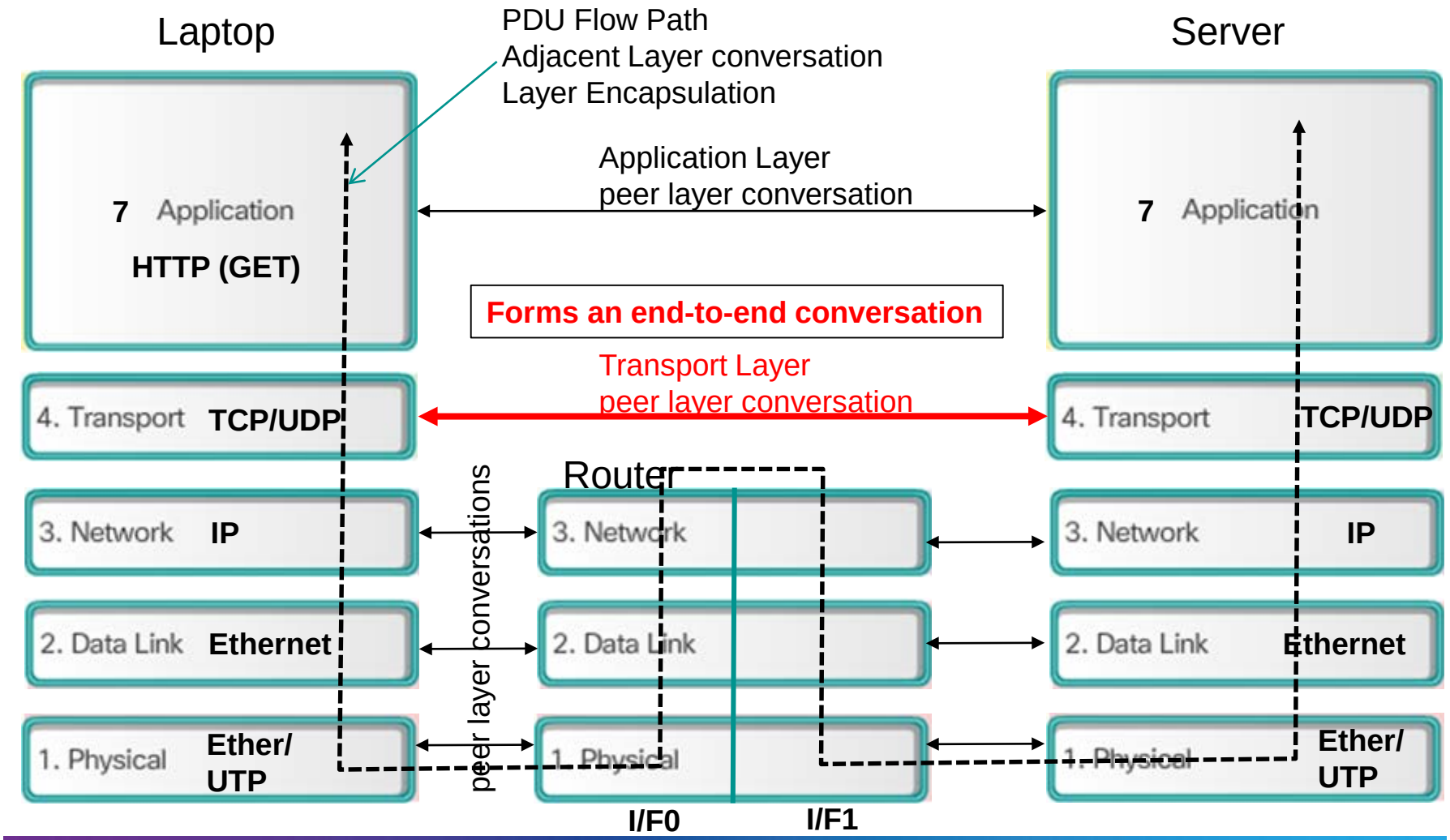
Section 9.1: Transport Layer Protocols

Upon completion of this section, you should be able to:

- Describe the purpose of the transport layer in managing the transportation of data in end-to-end communication.
- Describe characteristics of the TCP and UDP protocols, including port numbers and their uses.

Topic 9.1.1: Transportation of Data

Transport Layer



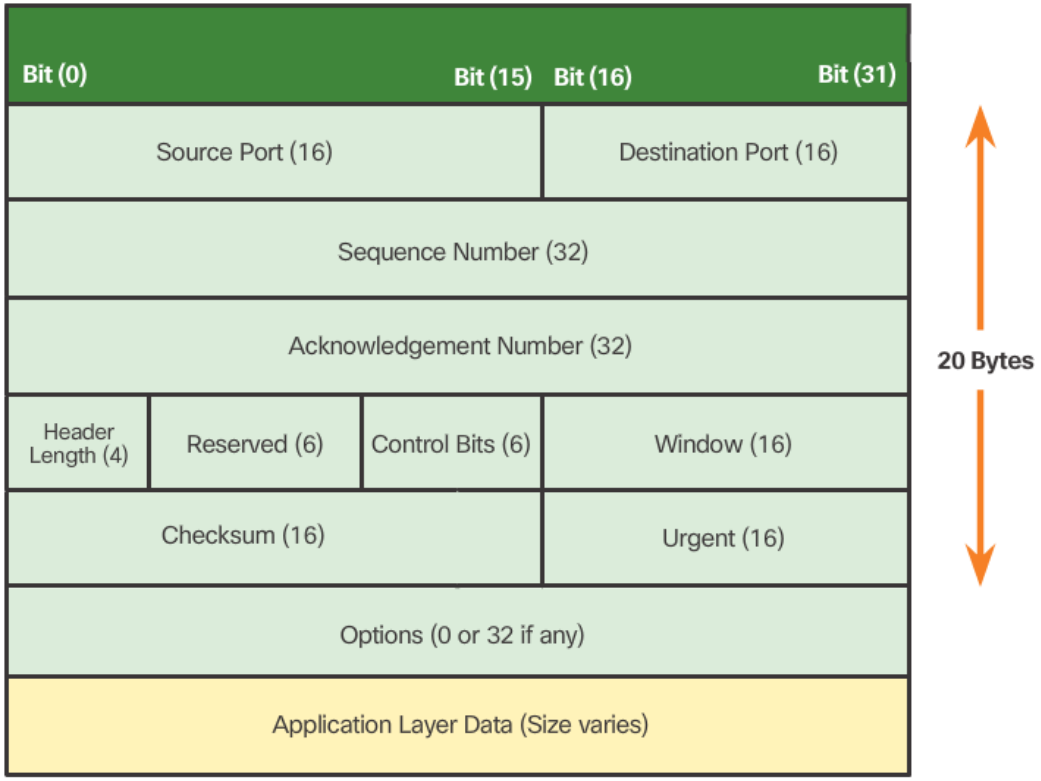
Transport Layer Protocols

- **Transmission Control Protocol - TCP**
  - Connection-Oriented
- **User Datagram Protocol – UDP**
  - Connectionless

Transmission Control Protocol - TCP

TCP Header

- TCP is a stateful protocol. It keeps track of the state of the communication session by recording which information it has sent and which information has been acknowledged.
- Each TCP segment has 20 bytes of overhead in the header encapsulating the application layer data, as shown in this image.



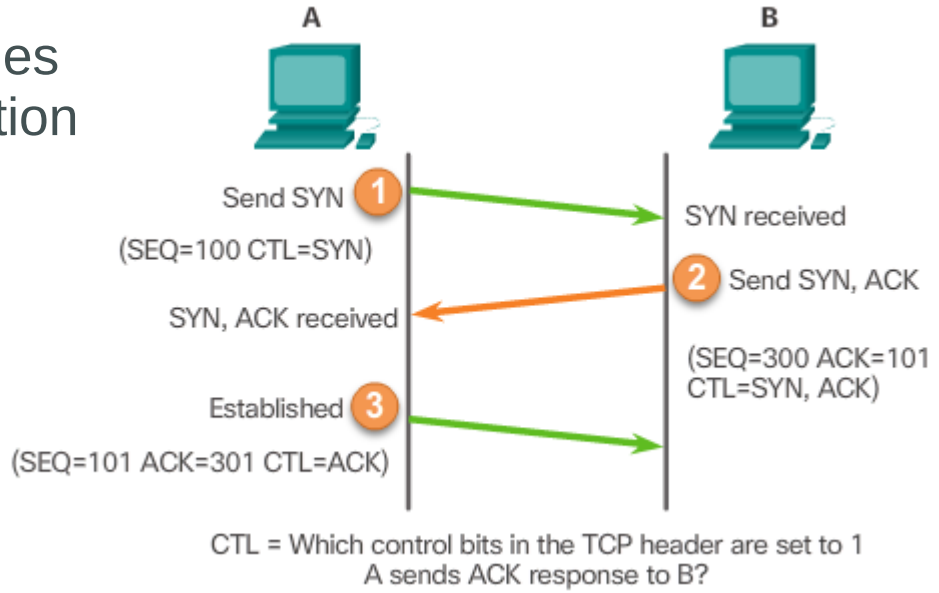
TCP Features

- **Segmentation**
- **Connection-Oriented**
  - Maintains Connection State
- **Ordered Delivery**
  - Restores Transmitted Packet Order
- **Reliable Service**
  - Retransmission
- **Flow and Congestion Control**
  - Packet Rate Control
- **Multiplexing**
  - Multiple Simultaneous Applications

TCP Connection Establishment

A TCP connection is established in three steps:

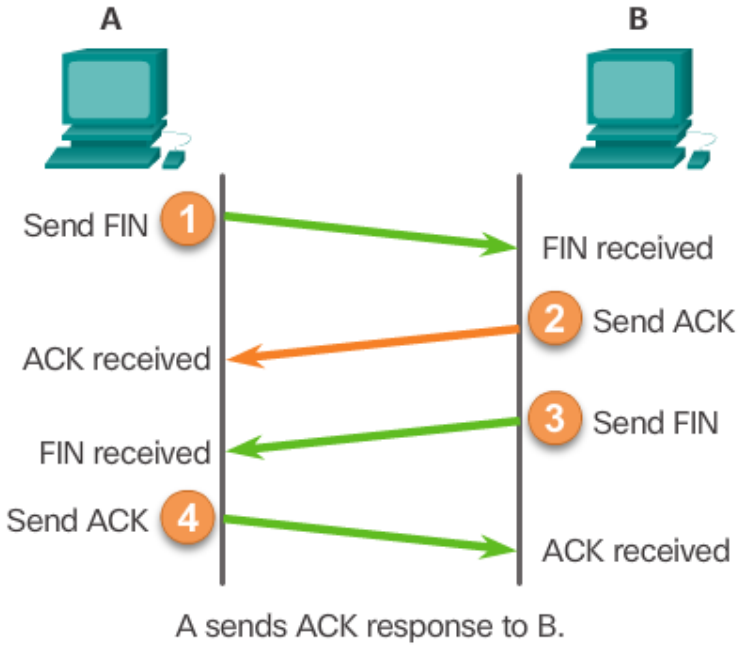
1. The initiating client requests a client-to-server communication session with the server.
2. The server acknowledges the client-to-server communication session and requests a server-to-client communication session.
3. The initiating client acknowledges the server-to-client communication session.



TCP Session Termination

The FIN TCP flag is used to terminate a TCP connection.

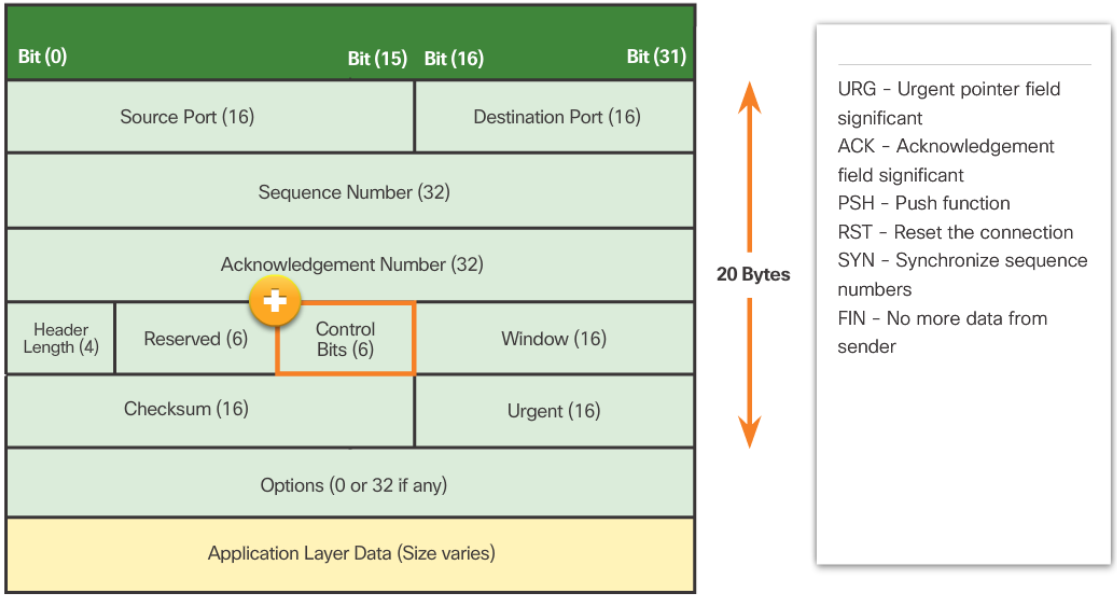
1. When the client has no more data to send in the stream, it sends a segment with the FIN flag set.
2. The server sends an ACK to acknowledge the receipt of the FIN to terminate the session from client to server.
3. The server sends a FIN to the client to terminate the server-to-client session.
4. The client responds with an ACK to acknowledge the FIN from the server.
5. When all segments have been acknowledged, the session is closed.



TCP Three-Way Handshake Analysis

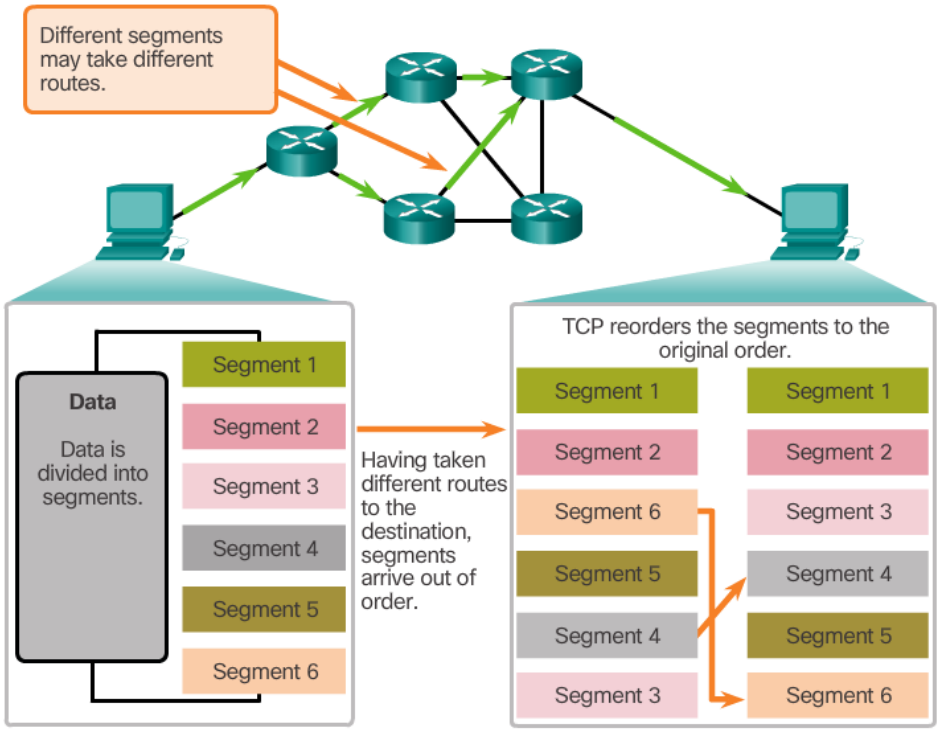
The three-way handshake:

- Establishes that the destination device is present on the network.
- 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
- Informs the destination device that the source client intends to establish a communication session on that port number.



TCP Reliability – Ordered Delivery

- 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.
- 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.

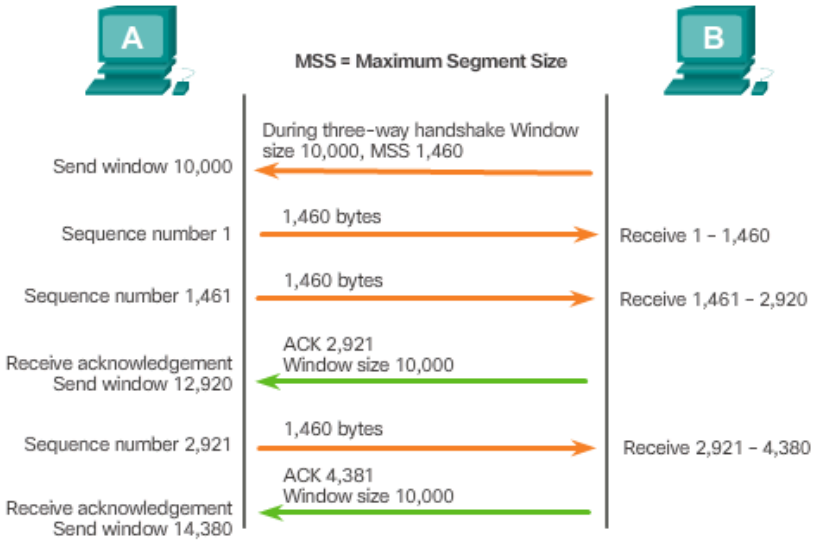


TCP Reliability - Sequence Numbers and Acknowledgements

- TCP is designed to confirm that each segment reached its destination.
- The TCP process on the destination host acknowledges the data it has received from the source application.
- TCP allows for the retransmission of missed segments.
- TCP session termination allows for parties to gracefully end a TCP session when no data is to be transferred (FIN flag).
- The video on page 9.2.2.2 covers TCP Sequence Numbers and Acknowledgements.

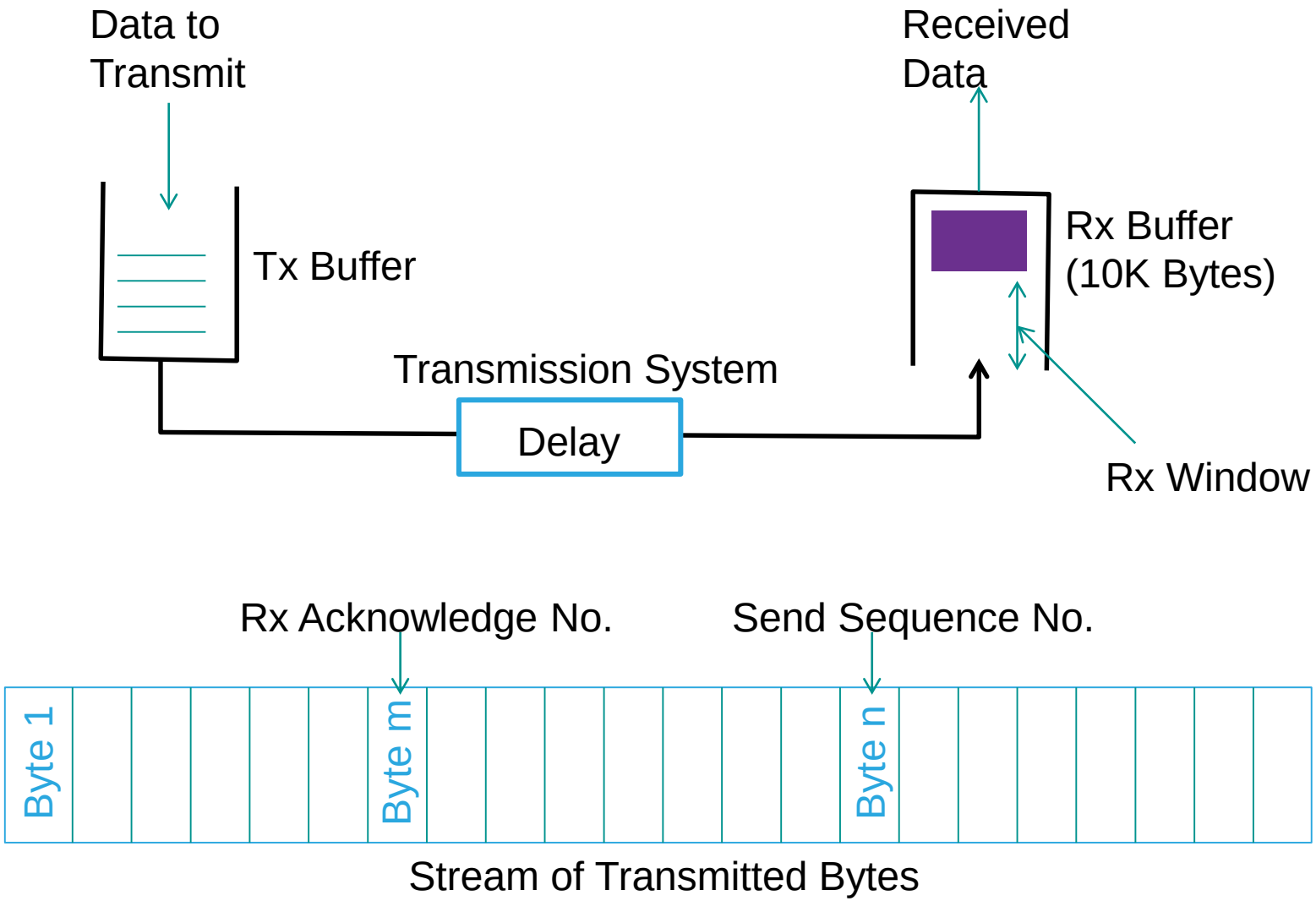
TCP Flow Control – Window Size and Acknowledgements

- TCP provides mechanisms for flow control.
- Flow control ensures the TCP endpoints can receive and process data reliably.
- TCP handles flow control 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. The window size is the number of bytes that the destination device of a TCP session can accept and process at one time.
- 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.



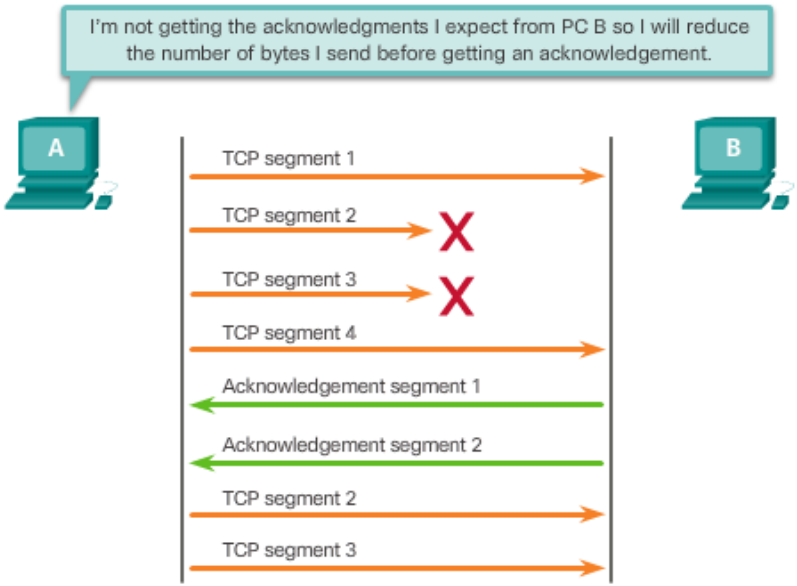
The **window size** determines the number of bytes that can be sent before expecting an acknowledgment.
The **acknowledgement** number is the number of the next expected byte.

TCP Flow Control – Window Size and Acknowledgements



TCP Flow Control – Congestion Avoidance

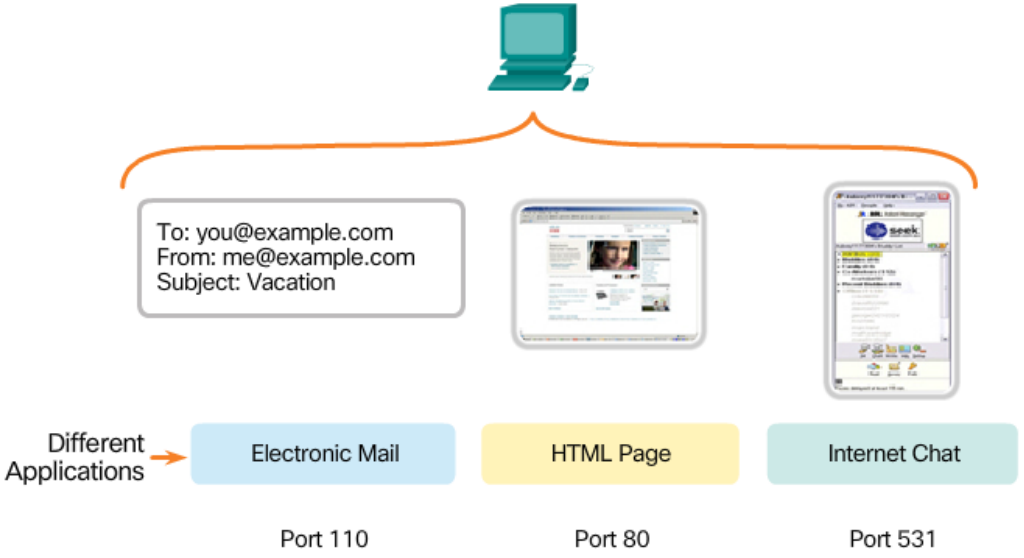
- Network congestion usually results in discarded packets.
- Undelivered TCP segments trigger re-transmission. TCP segment retransmission can make the congestion even worse.
- The source can estimate a certain level of network congestion by looking at the rate at which TCP segments are sent but not acknowledged.
- 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.



Acknowledgement numbers are for the next expected byte and not for a segment. Segment number are only used here for simplicity.

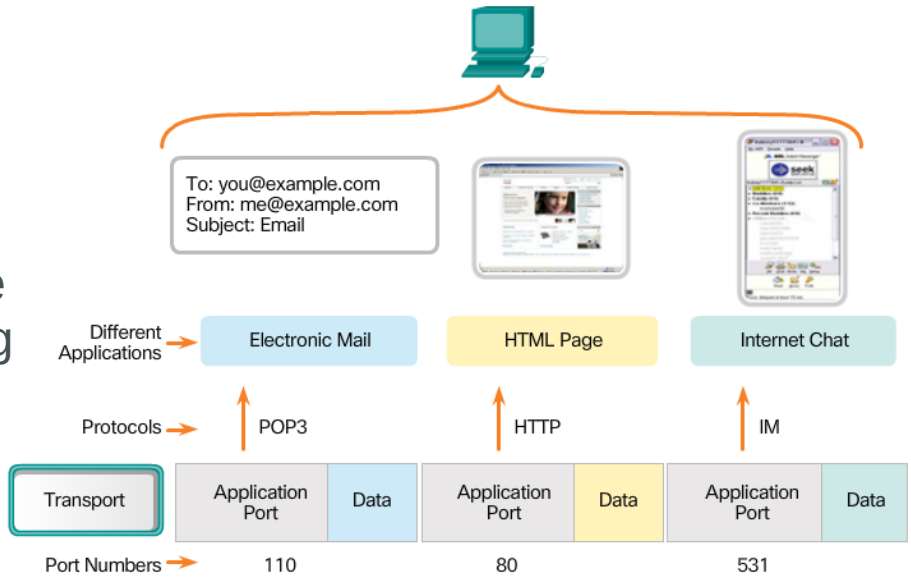
Multiple Separate Conversations

- The transport layer must separate and manage multiple communications with different transport requirements.
- Different applications are sending and receiving data over the network simultaneously.
- Unique header values allow TCP and UDP to manage these multiple and simultaneous conversations by identifying these applications.
- These unique identifiers are the port numbers.



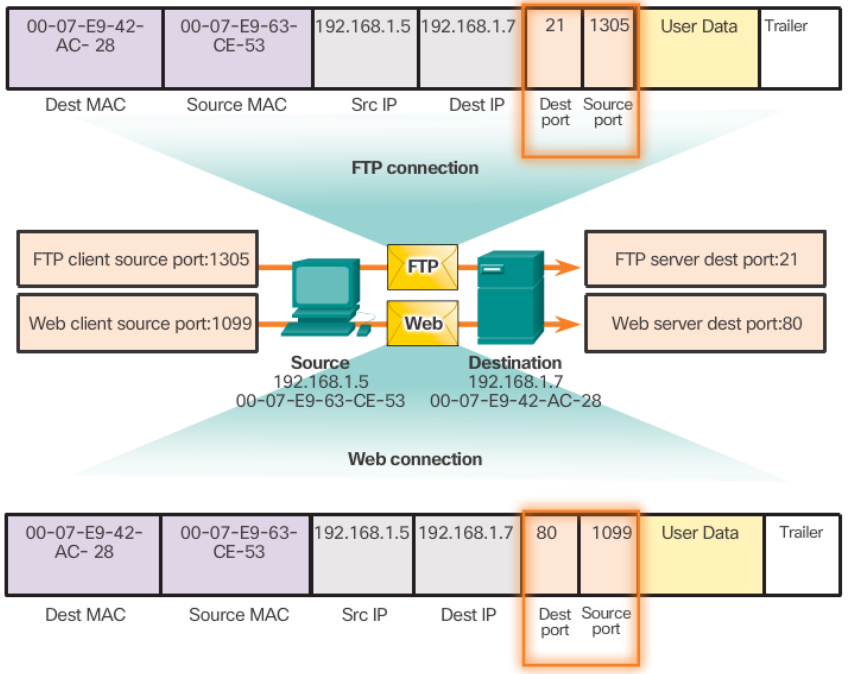
Port Numbers

- Source Port
  - The source port number is dynamically chosen by the sending device to identify a conversation between two devices.
  - An HTTP client usually sends multiple HTTP requests to a web server at the same time. Each separate HTTP conversation is tracked based on the source ports.
- Destination Port
  - Used to identify an application or service running in the server.
  - A server can offer more than one service at the same time, offering a web service on port 80 and FTP on port 21 simultaneously.



Socket Pairs

- 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.
- The socket is used to identify the server and service being requested by the client.
- Two sockets combine to form a socket pair: (192.168.1.5:1099, 192.168.1.7:80)
- Sockets enable multiple processes running on a client 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.
- It is the transport layer's job to keeps track of active sockets.



Port Number Groups

The Internet Assigned Numbers Authority (IANA) is the standards body responsible for assigning various addressing standards, including port numbers.

Port Numbers

Port Number Range	Port Group
0 to 1023	Well-known Ports
1024 to 49151	Registered Ports
49152 to 65535	Private and/or Dynamic Ports

Well-Known Port Numbers

Port Number	Protocol	Application	Acronym
20	TCP	File Transfer Protocol (data)	FTP
21	TCP	File Transfer Protocol (control)	FTP
22	TCP	Secure Shell	SSH
23	TCP	Telnet	–
25	TCP	Simple Mail Transfer Protocol	SMTP
53	UDP, TCP	Domain Name Service	DNS
67, 68	UDP	Dynamic Host Configuration Protocol	DHCP
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 indicate a major security threat.
- Netstat is an important network utility that can be used to verify the active connections in a host.
- Use **netstat** to list the protocols in use, the local address and port numbers, the foreign address and port numbers, and the connection state.
- By default, the **netstat** command will attempt to resolve IP addresses to domain names and port numbers to well-known applications.
- The **-n** option can be used to display IP addresses and port numbers in their numerical form.

```
C:\> netstat

Active Connections

Proto   Local Address    Foreign Address          State
TCP     kenpc:3126       192.168.0.2:netbios-ssn  ESTABLISHED
TCP     kenpc:3158       207.138.126.152:http     ESTABLISHED
TCP     kenpc:3159       207.138.126.169:http     ESTABLISHED
TCP     kenpc:3160       207.138.126.169:http     ESTABLISHED
TCP     kenpc:3161       sc.msn.com:http          ESTABLISHED
TCP     kenpc:3166       www.cisco.com:http       ESTABLISHED

C:\>
```

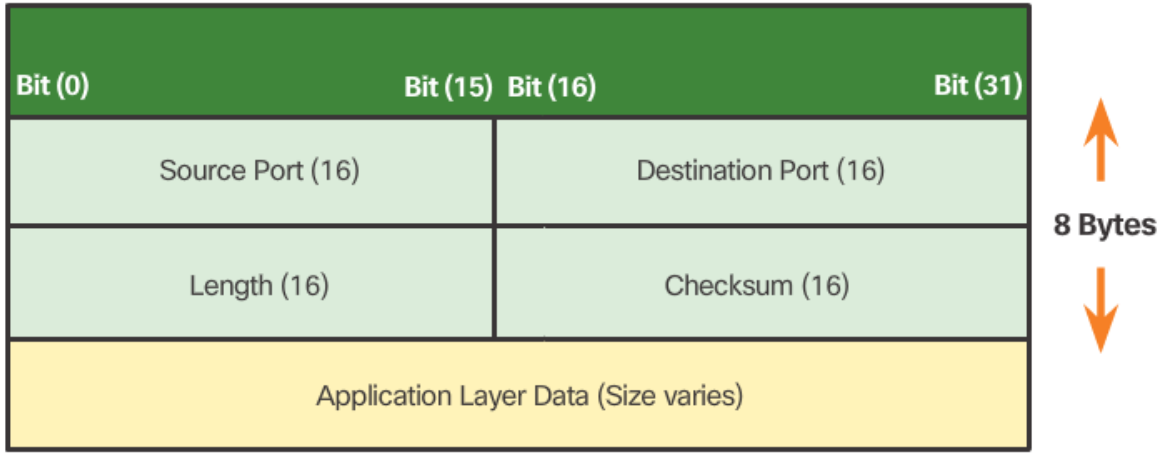
TCP Features

- **Segmentation**
- **Connection-Oriented**
  - Syn-Syn/Ack-Ack; Fin-Ack-Fin-Ack
- **Ordered Delivery**
  - Sequence Numbers
- **Reliable Service**
  - Acknowledgement
- **Flow and Congestion Control**
  - Window and Rate Control
- **Multiplexing**
  - Ports

User Datagram Protocol - UDP

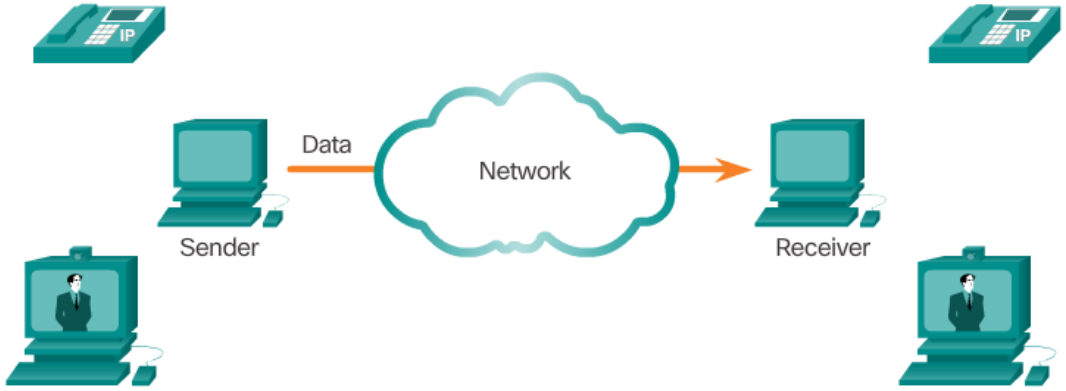
UDP Header

- UDP is a stateless protocol. Neither the sender or the receiver is obligated to keep track of the state of the communication session.
- Reliability must be handled by the application.
- Live video and voice applications must quickly deliver data and can tolerate some data loss; they are perfectly suited to UDP.
- The pieces of communication in UDP are called datagrams.
- These datagrams are sent as best-effort by the transport layer protocol.
- UDP has a low overhead of 8 bytes.



UDP Low Overhead versus Reliability

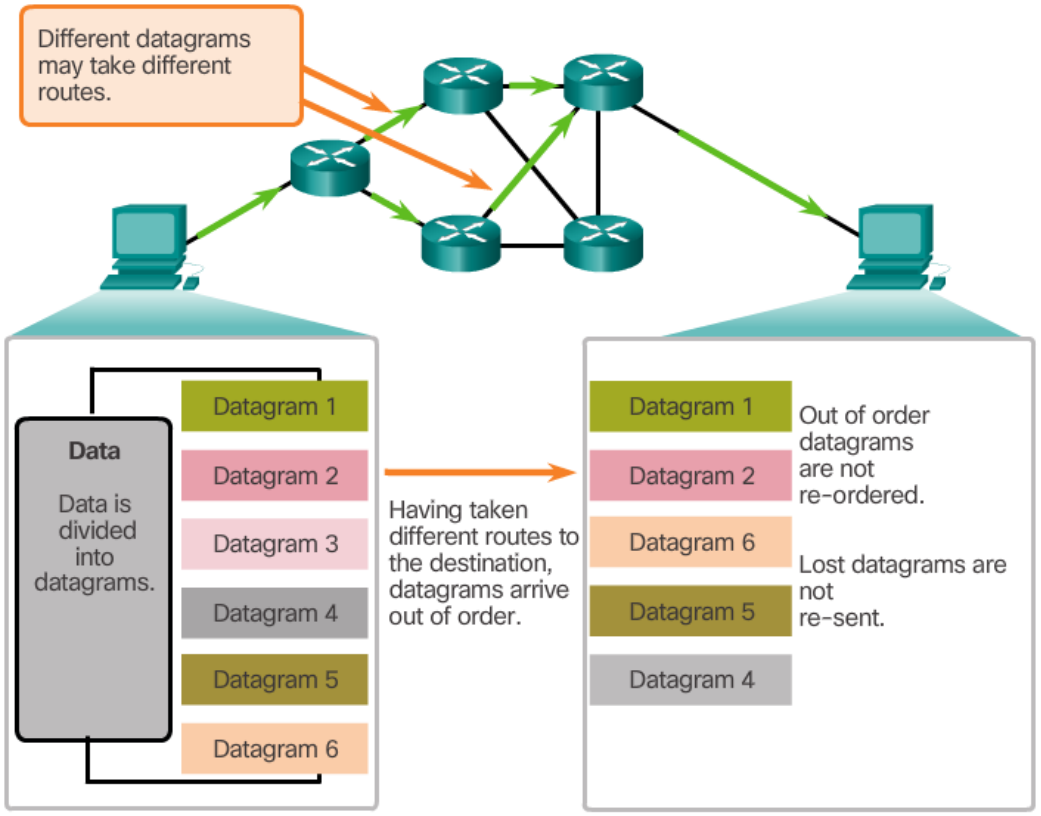
- UDP is a simple protocol.
- UDP provides the basic transport layer functions.
- UDP has much lower overhead than TCP.
- UDP is not connection-oriented and does not offer the sophisticated retransmission, sequencing, and flow control mechanisms.
- Applications running UDP can still use reliability, but it must be implemented in the application layer.
- However, UDP is not inferior. It is designed to be simpler and faster than TCP at the expense of reliability.



UDP does not establish a connection before sending data.

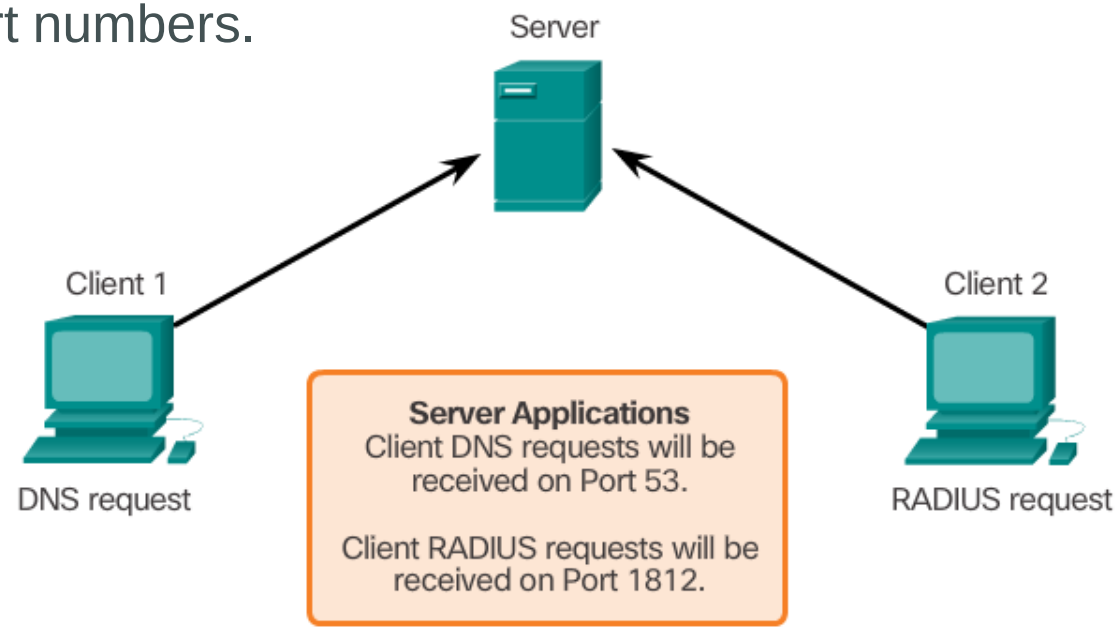
UDP Datagram Reassembly

- 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 in which it was received.
- The application must identify the proper sequence, if necessary.



UDP Server Processes

- UDP-based server applications are also assigned well-known or registered port numbers.
- UDP applications and services running on a server accept UDP client requests.
- Requests received on a specific port are forwarded to the proper application based on port numbers.



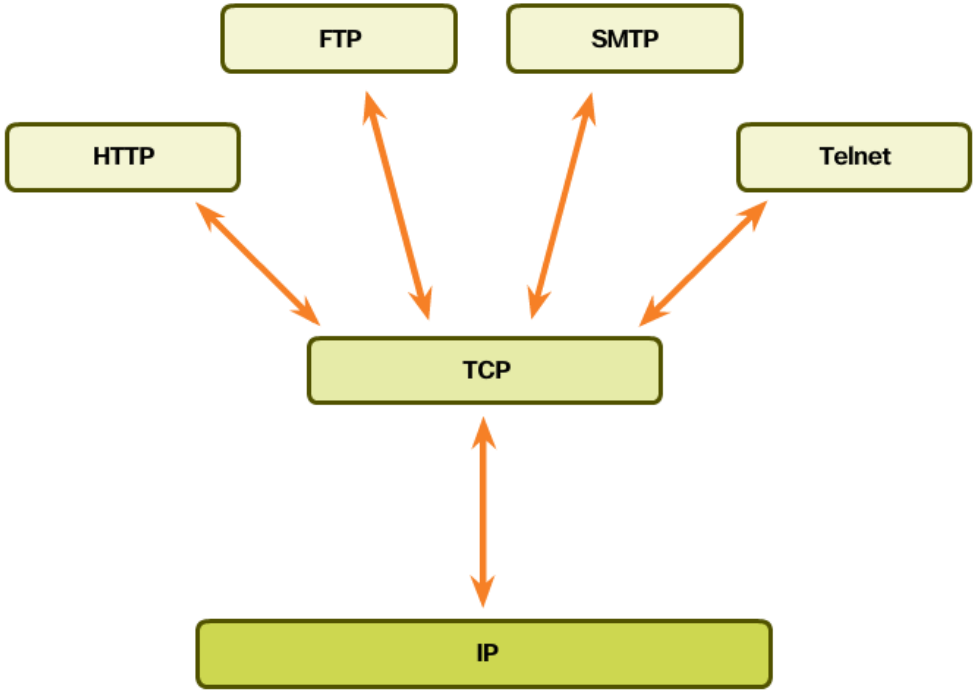
UDP Server Processes

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

TCP or UDP

Applications that use TCP

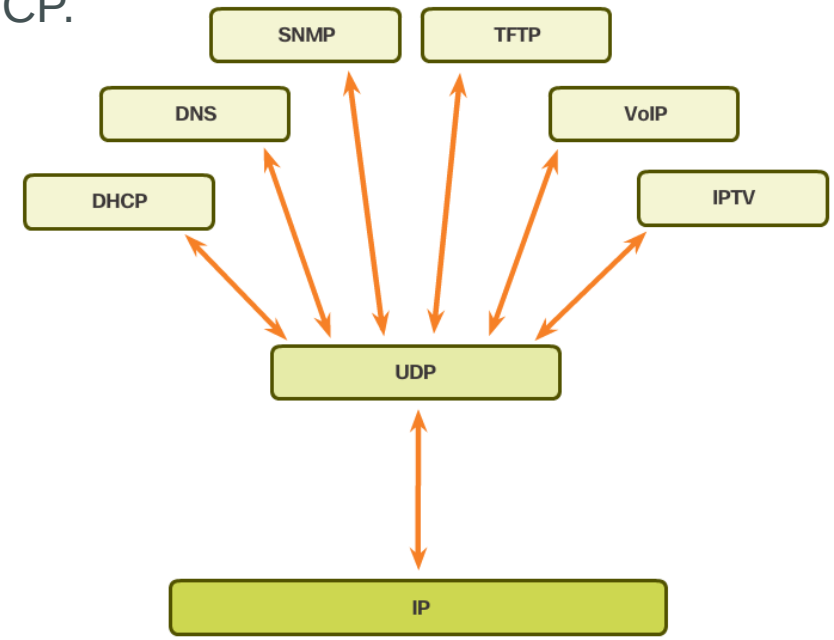
- TCP handles all transport layer related tasks.
- This frees the application from having to manage any of these tasks.
- Applications can simply send the data stream to the transport layer and use the services of TCP.



Applications that use UDP

There are three types of applications that are best suited for UDP:

- **Live video and multimedia 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.
- **Applications that handle reliability themselves** - Unidirectional communications where flow control, error detection, acknowledgements, and error recovery is not required or can be handled by the application. Examples include SNMP and TFTP.



Thank you.

Cisco Networking Academy

Mind Wide Open