Cisco Networking Academy

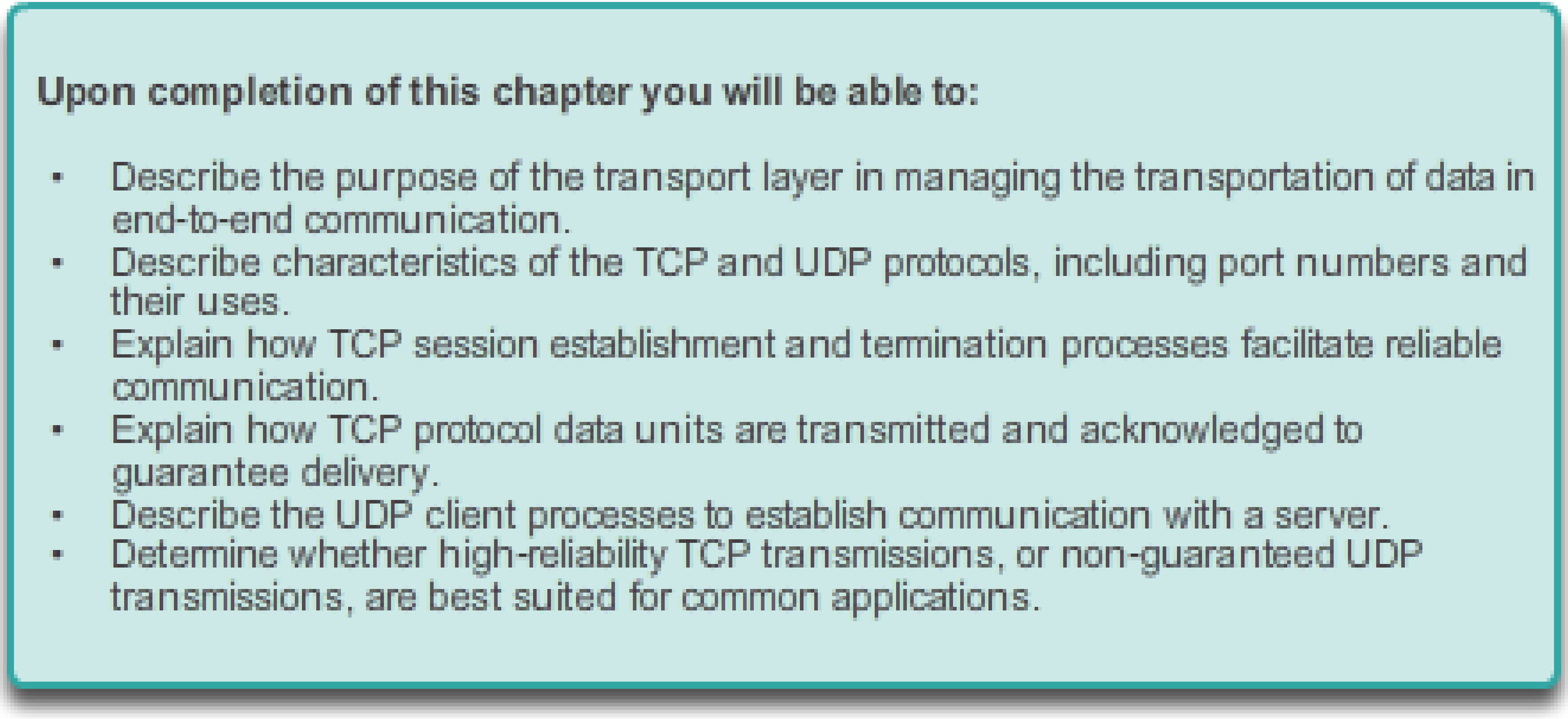
CCNA R&S: Introduction to Networks

Chapter 7:

The Transport Layer

Frank Schneemann

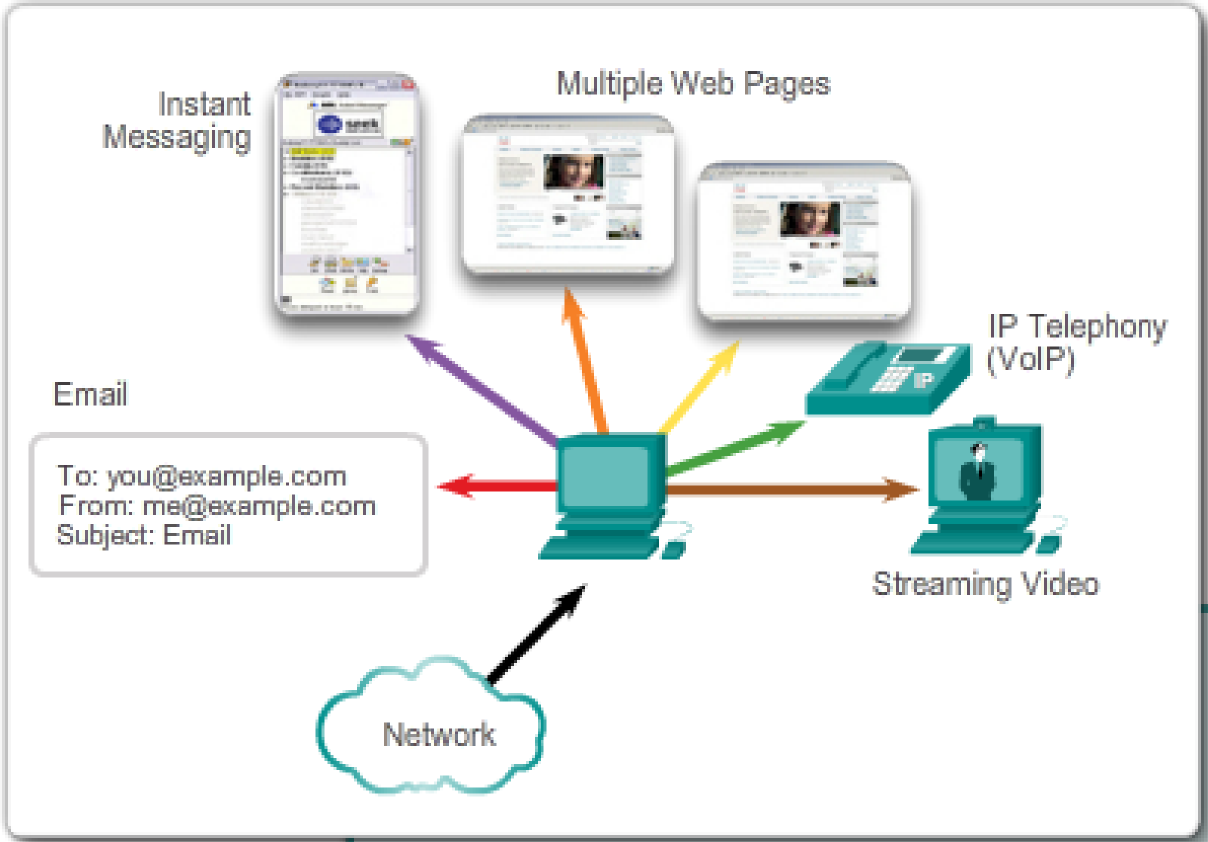
7.0.1.1 Introduction



Upon completion of this chapter you will 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.
- Explain how TCP session establishment and temination processes facilitate reliable communication.
- Explain how TCP protocol data units are transmitted and acknowledged to guarantee delivery.
- Describe the UDP client processes to establish communication with a server.
- Detemine whether high-reliability TCP transmissions, or non-guaranteed UDP transmissions, are best suited for common applications.

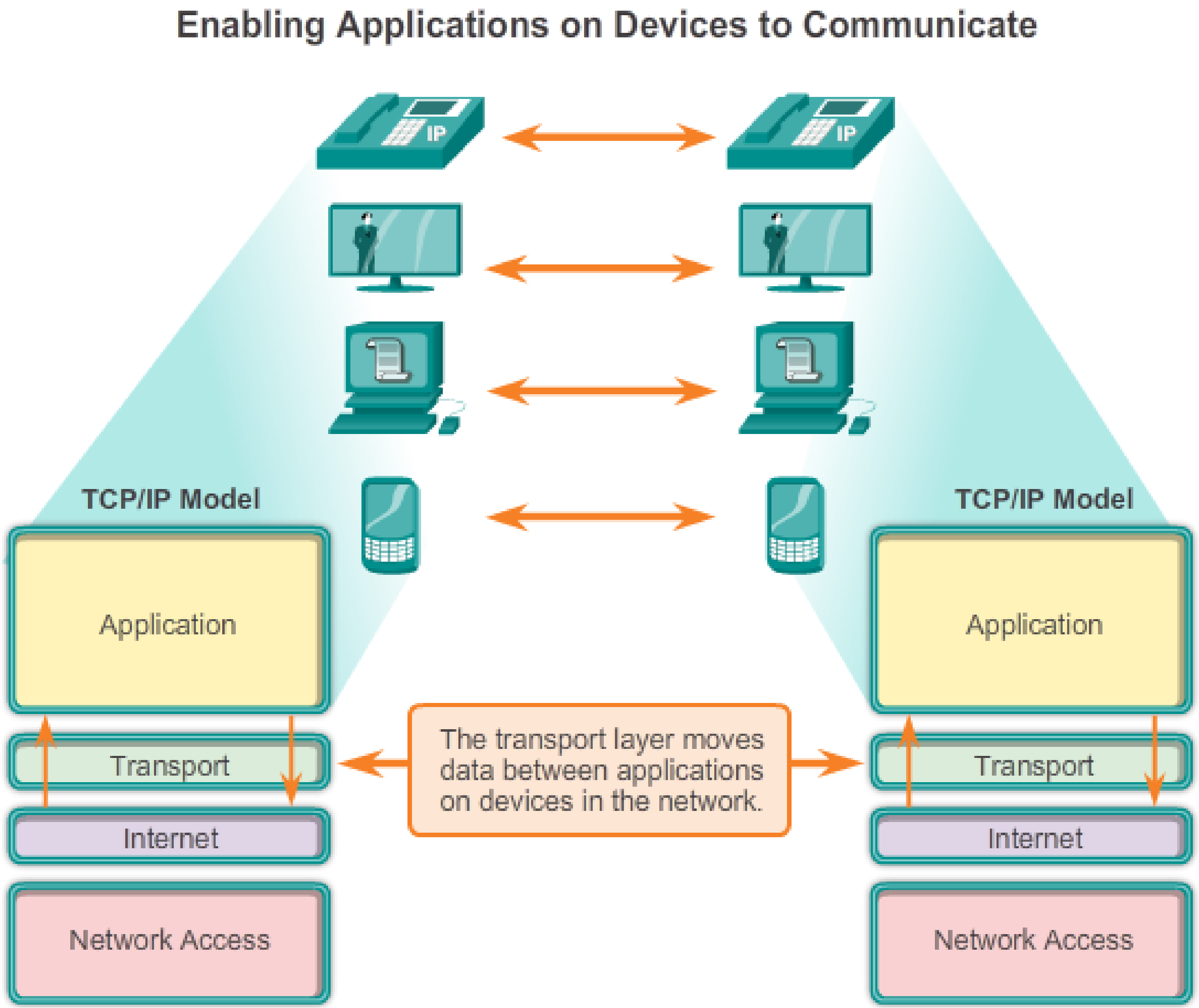
7.0.1.2 Class Activity - We Need to Talk – Game



While completing this activity, think about how:

- *Network communications have different levels of importance*
- *Important data must be accurate when sent and received*
- *Timing can be a factor when choosing a data delivery method*

7.1.1.1 Role of the Transport Layer

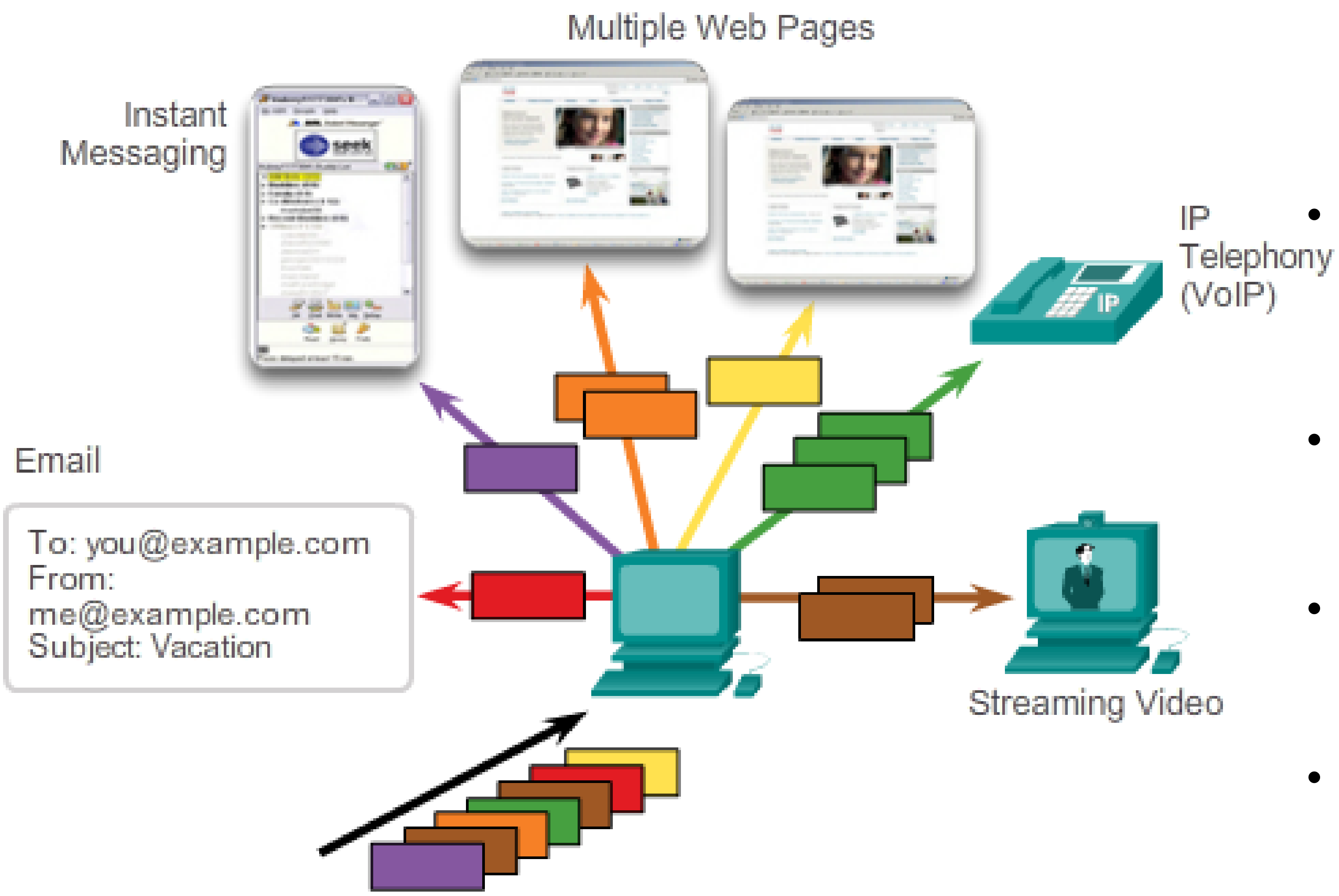

Enabling Applications on Devices to Communicate
IP
IP
TCP/IP Model
TCP/IP Model
Application
Transport
Internet
Network Access
Application
Transport
Internet
Network Access
The transport layer moves data between applications on devices in the network.

The primary responsibilities of transport layer protocols are:

- **Tracking** the 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
- **Identifying** the proper application for each communication stream

7.1.1.2 Role of the Transport Layer (Cont.)

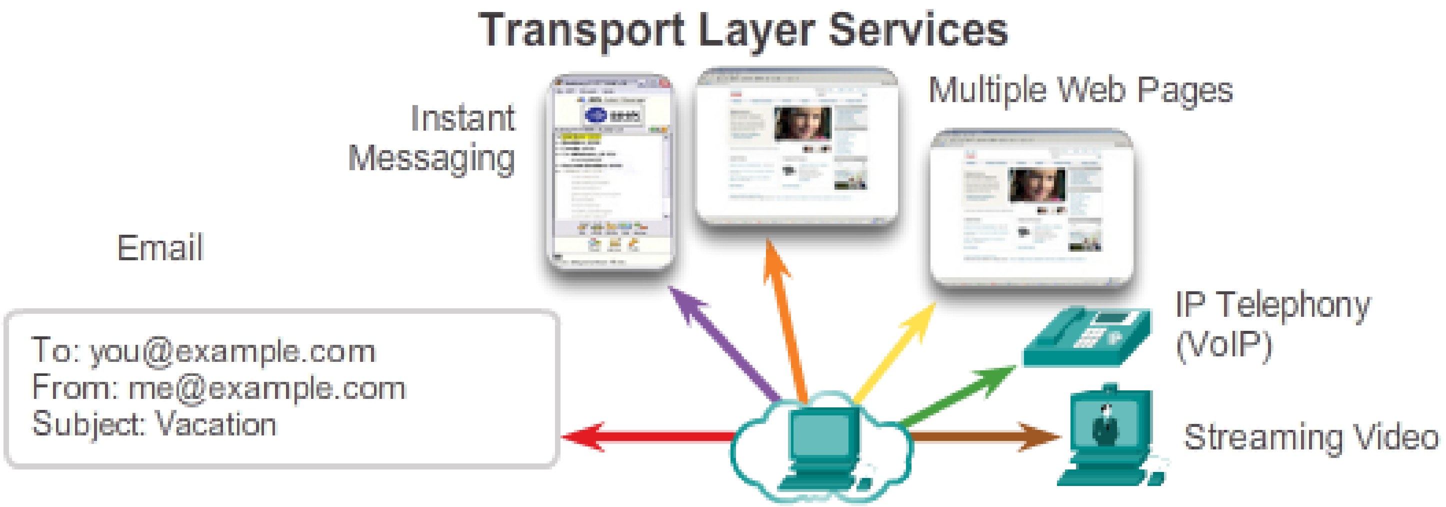
Identifying the Application

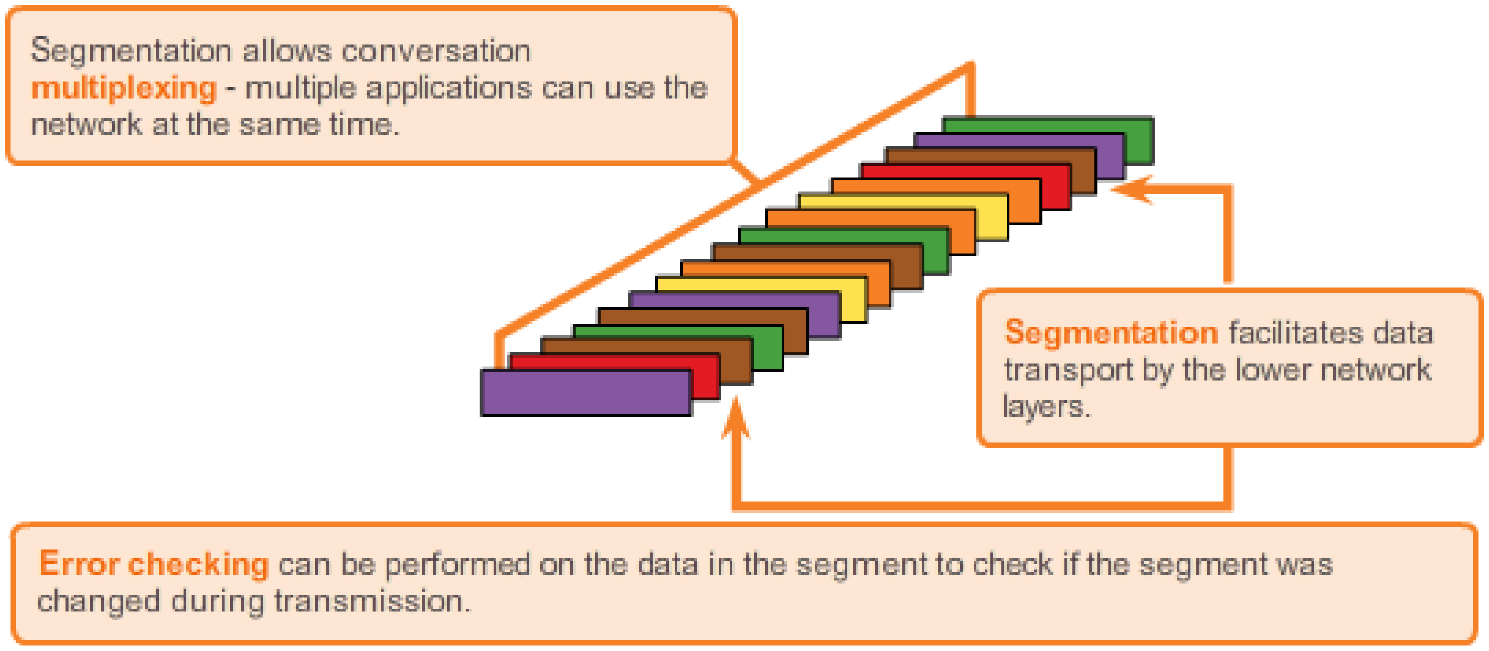


To: you@example.com
From:
me@example.com
Subject: Vacation

- Applications communicates with one or more applications on one or more remote hosts. It is the responsibility of the transport layer to maintain and track these multiple conversations.
- Transport layer protocols have services that segment the application data into blocks of data that are an appropriate size
- A header, used for reassembly, is added to each block of data. This header is used to track the data stream.
- the transport layer assigns each application an identifier. This identifier is called a port number.
- Each software process that needs to access the network is assigned a port number unique in that host

7.1.1.3 Conversation Multiplexing





Segmentation allows conversation **multiplexing** - multiple applications can use the network at the same time.

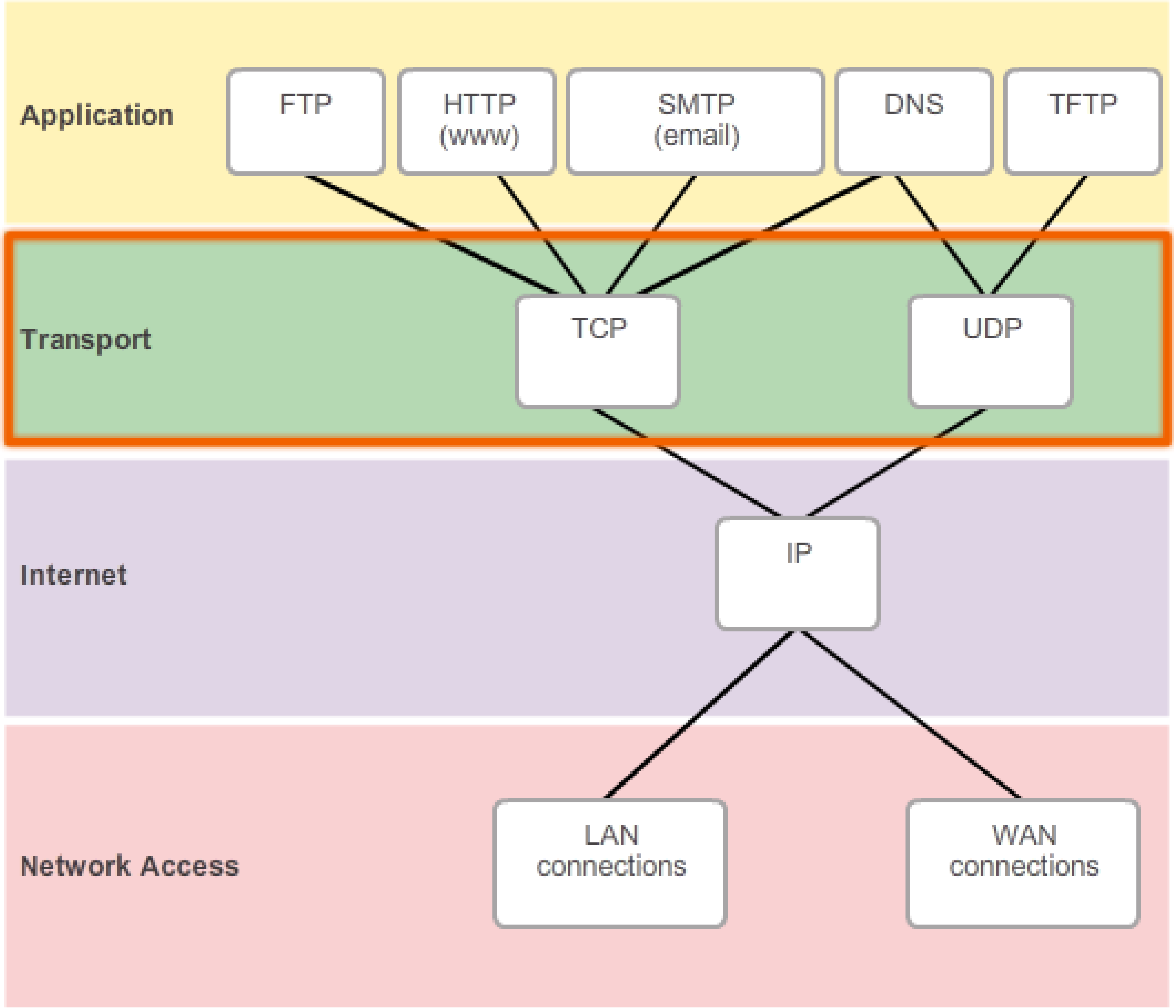
Segmentation facilitates data transport by the lower network layers.

Error checking can be performed on the data in the segment to check if the segment was changed during transmission.

- Segmentation of the data by transport layer protocols also provides the means to both send and receive data when running multiple applications concurrently on a computer.

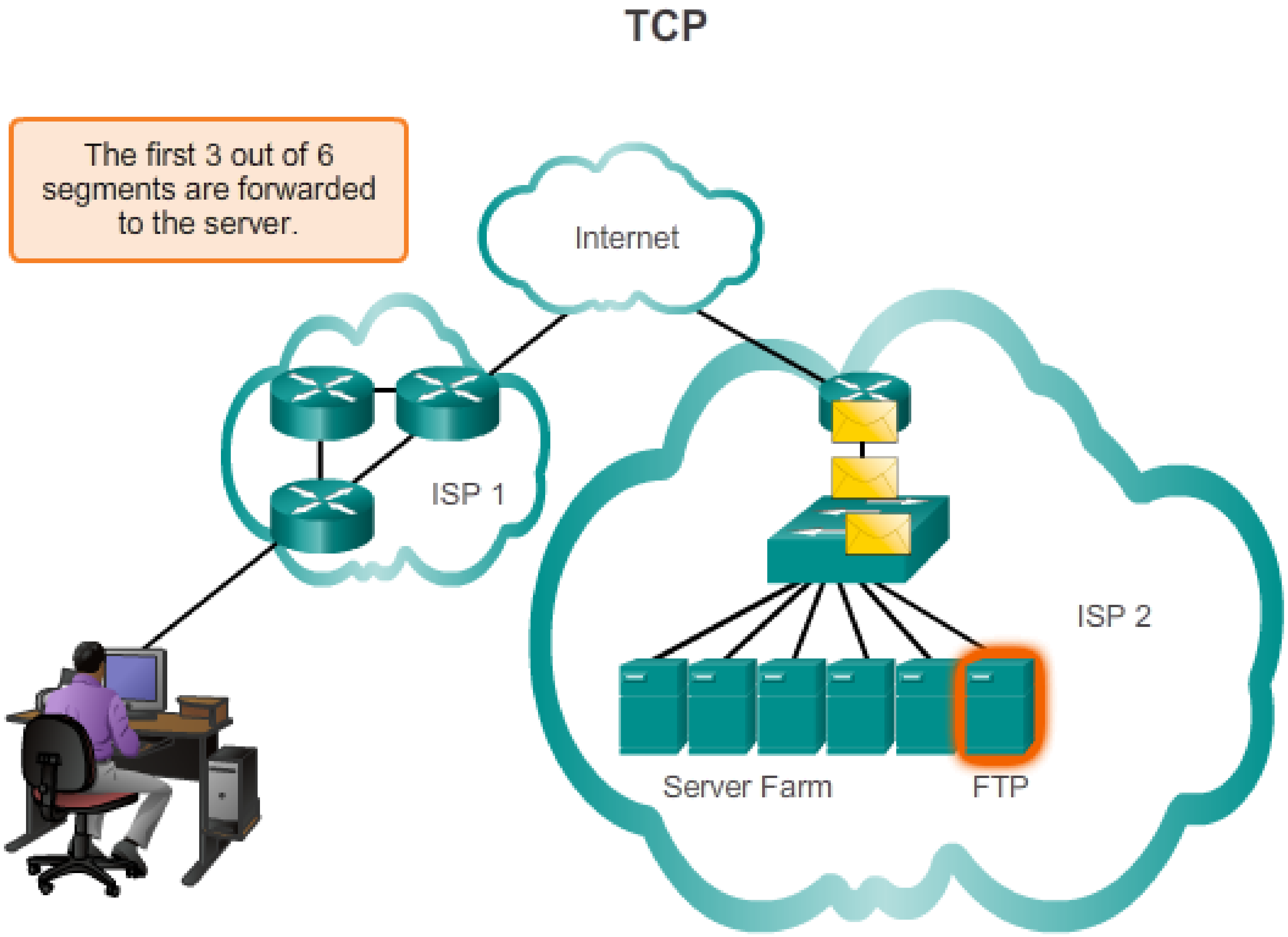
- To identify each segment of data, the transport layer adds to the segment a header containing binary data.
- This header contains fields of bits. It is the values in these fields that enable different transport layer protocols to perform different functions in managing data communication.

7.1.1.4 Transport Layer Reliability



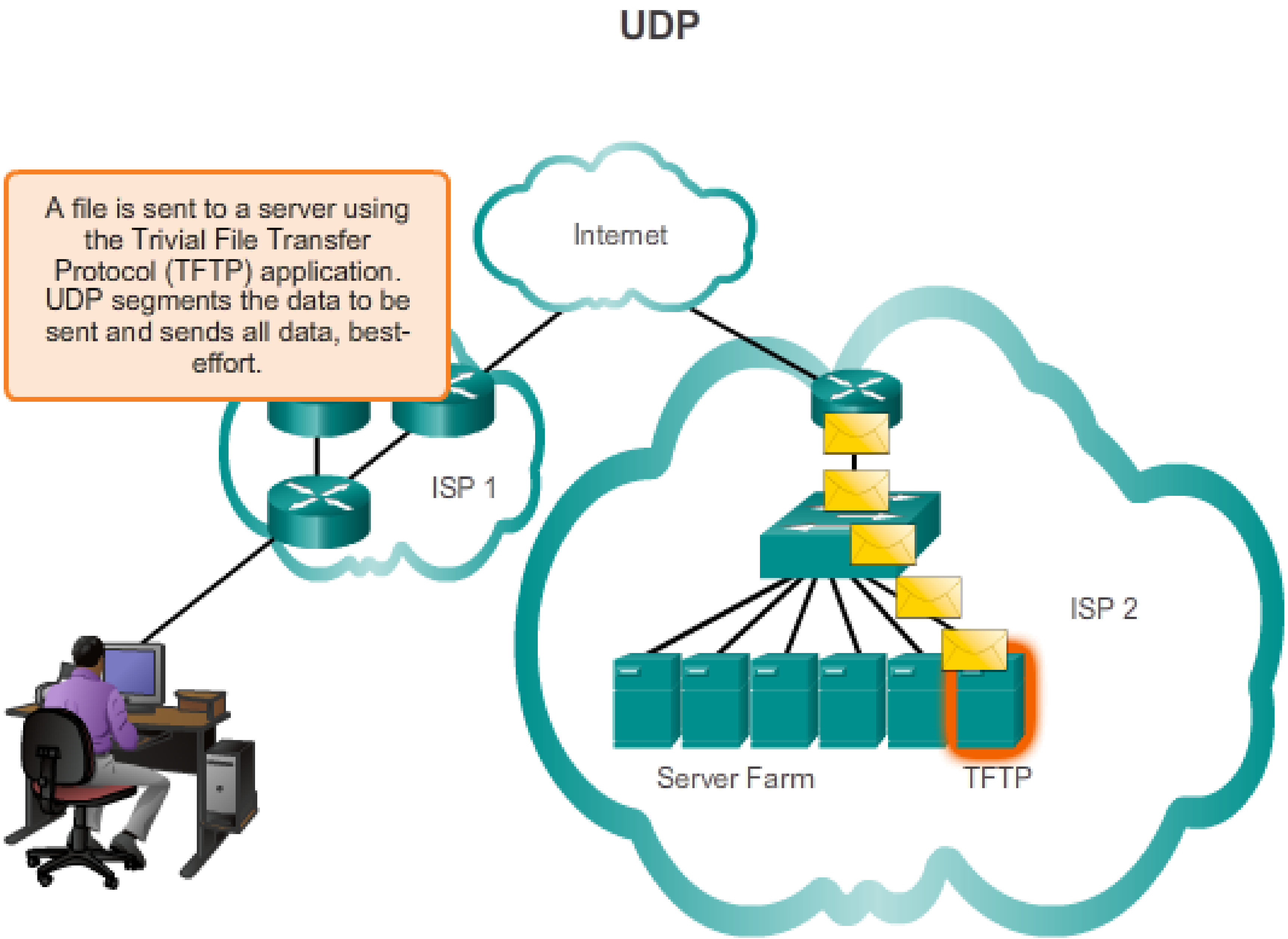
- IP is concerned only with the structure, addressing, and routing of packets.
- IP does not specify how the delivery or transportation of the packets takes place.
- Transport protocols specify how to transfer messages between hosts.
- TCP/IP provides two transport layer protocols, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP)
- IP uses these transport protocols to enable hosts to communicate and transfer data.

7.1.1.5 TCP



- With TCP, the three basic operations of reliability are:
- **Tracking** transmitted data segments
- **Acknowledging** received data
- **Retransmitting** any unacknowledged data
- These reliability processes place additional overhead on
- Control data is exchanged between the sending and receiving hosts. This control information is contained in a TCP header.

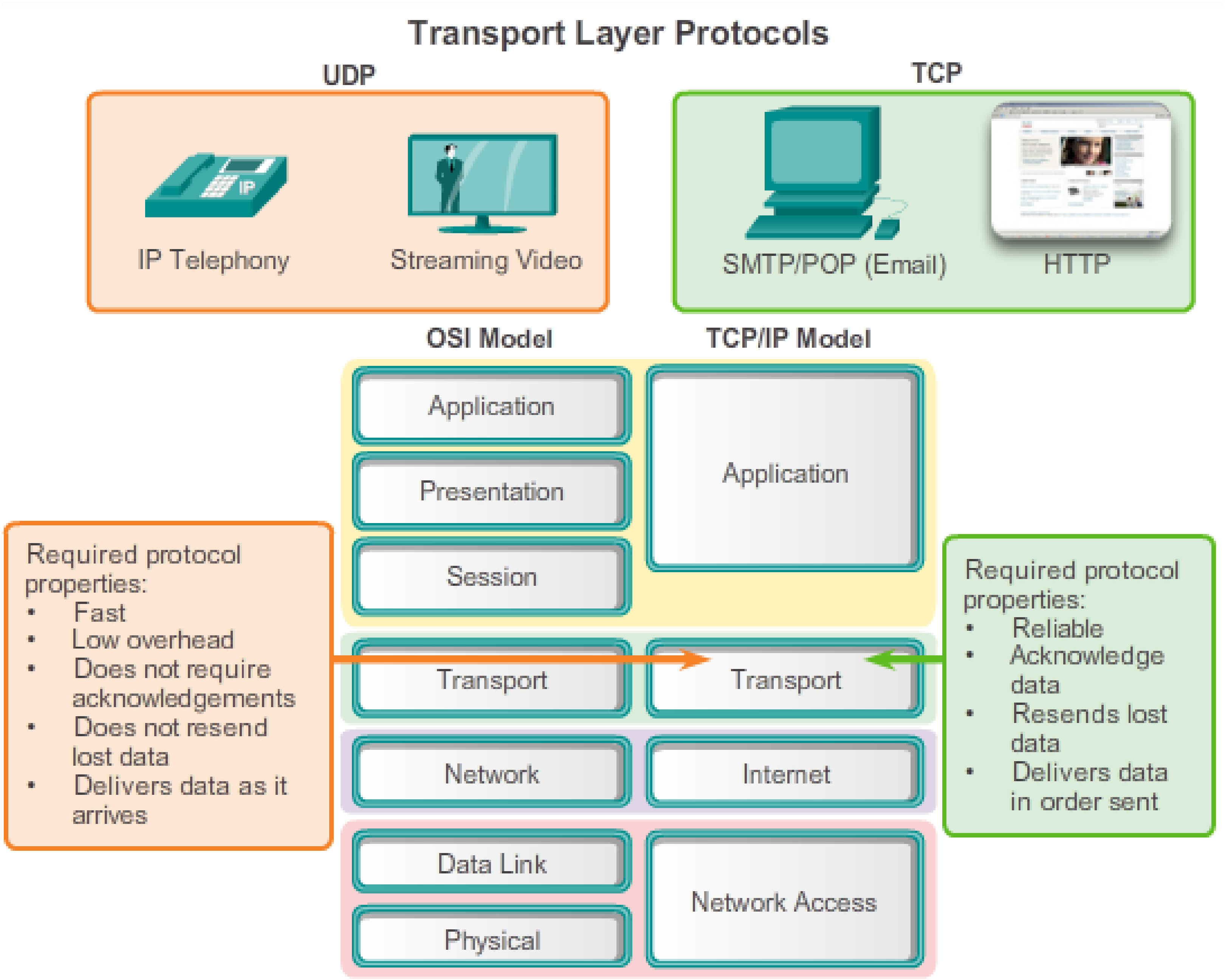
7.1.1.6 UDP



A file is sent to a server using the Trivial File Transfer Protocol (TFTP) application. UDP segments the data to be sent and sends all data, best-effort.

UDP provides just the basic functions for delivering data segments between the appropriate applications, with very little overhead and data checking. UDP is known as a **best-effort delivery protocol**.
In the context of networking, best-effort delivery is referred to as **unreliable**, because there is no acknowledgement that the data is received at the destination.
With UDP, there are no transport layer processes that inform the sender if successful delivery has occurred

7.1.1.7 The Right Transport Layer Protocol for the Right Application



Required protocol properties:

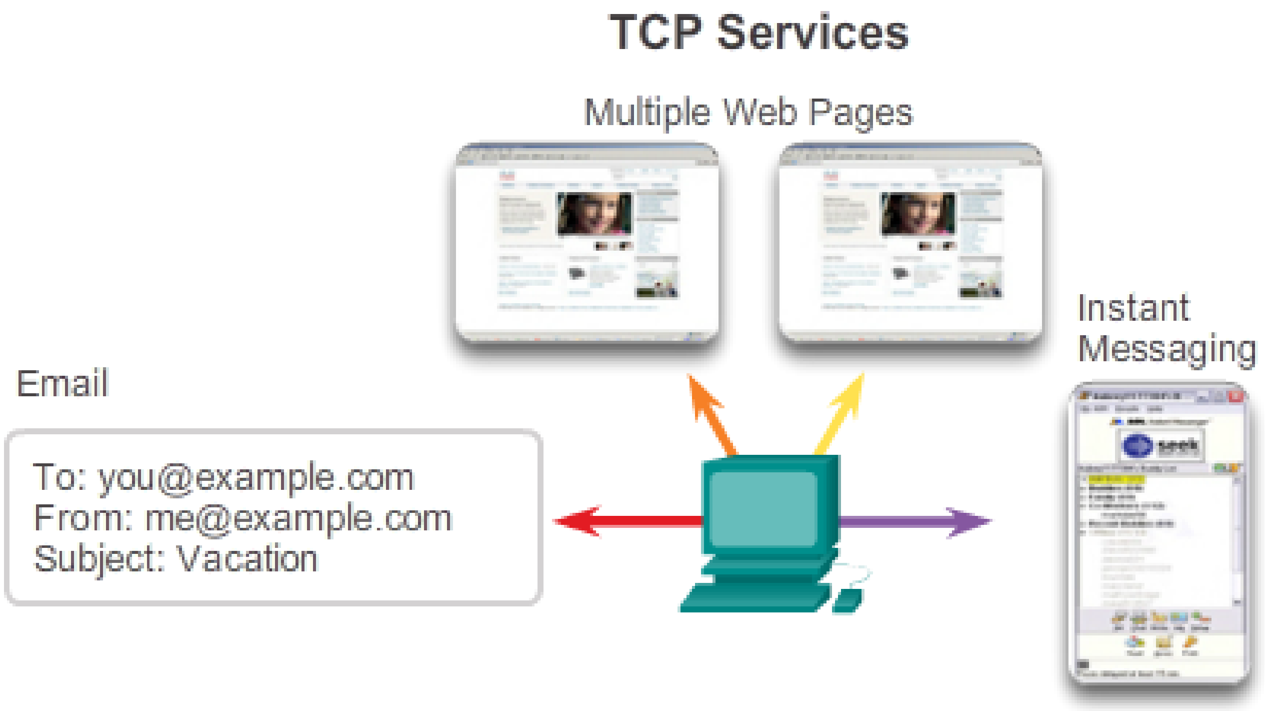
- Fast
- Low overhead
- Does not require acknowledgements
- Does not resend lost data
- Delivers data as it arrives

Required protocol properties:

- Reliable
- Acknowledge data
- Resends lost data
- Delivers data in order sent

Application developers choose the appropriate transport layer protocol based on the nature of the application

7.1.2.1 Introducing TCP



TCP Services

Multiple Web Pages

Instant Messaging

Email

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

Establishing a session ensures the application is ready to receive the data.

Same order delivery ensures that the segments are reassembled into the proper order.

Reliable delivery means lost segments are resent so the data is received complete.

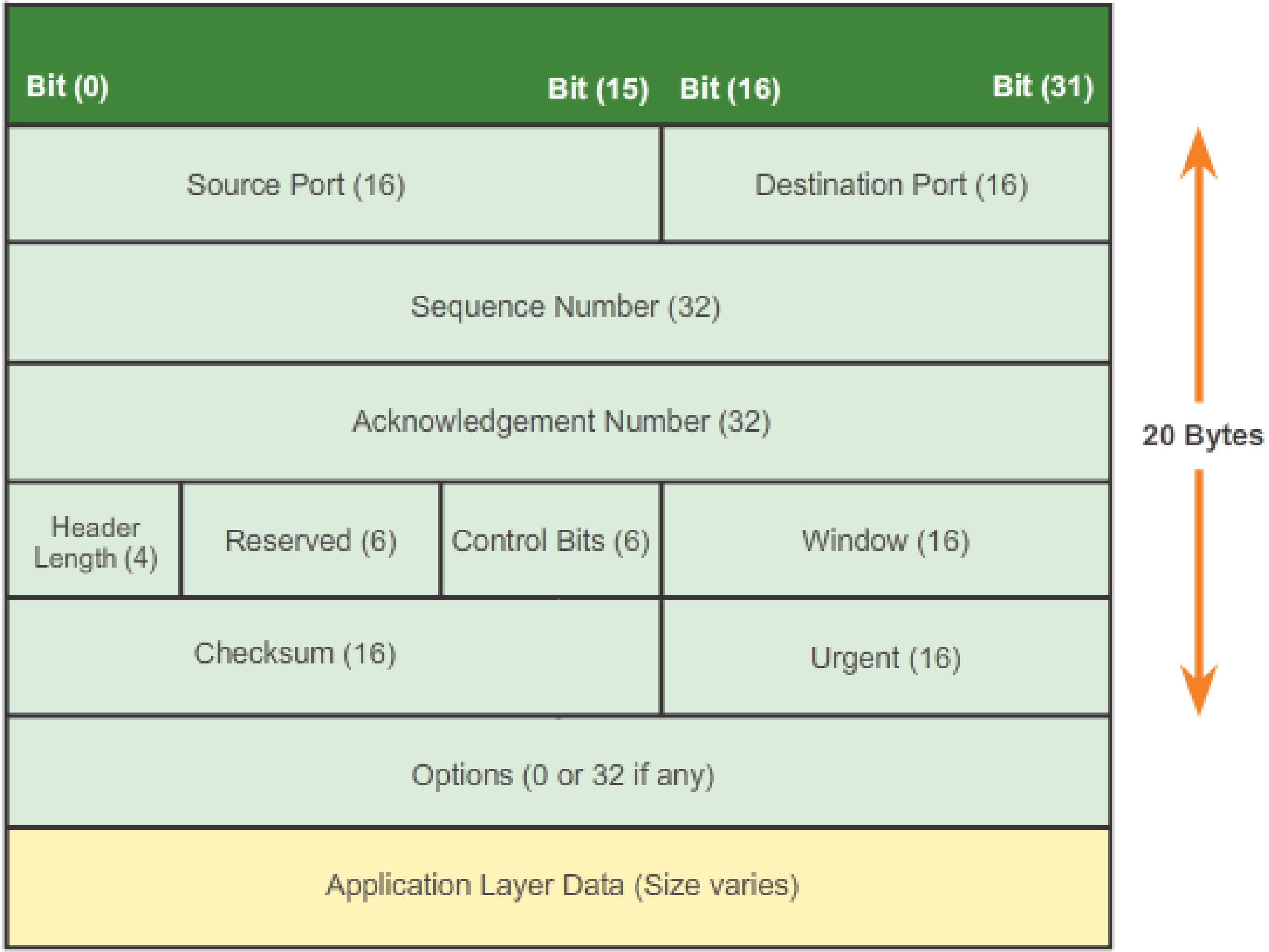
Flow control manages data delivery if there is congestion on the host.

TCP Provides

- Connection-oriented conversations by establishing sessions
- Reliable delivery
- Ordered data reconstruction
- Flow control

7.1.2.2 Role of TCP

TCP Segment



Bit (0) Bit (15) Bit (16) Bit (31)

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)

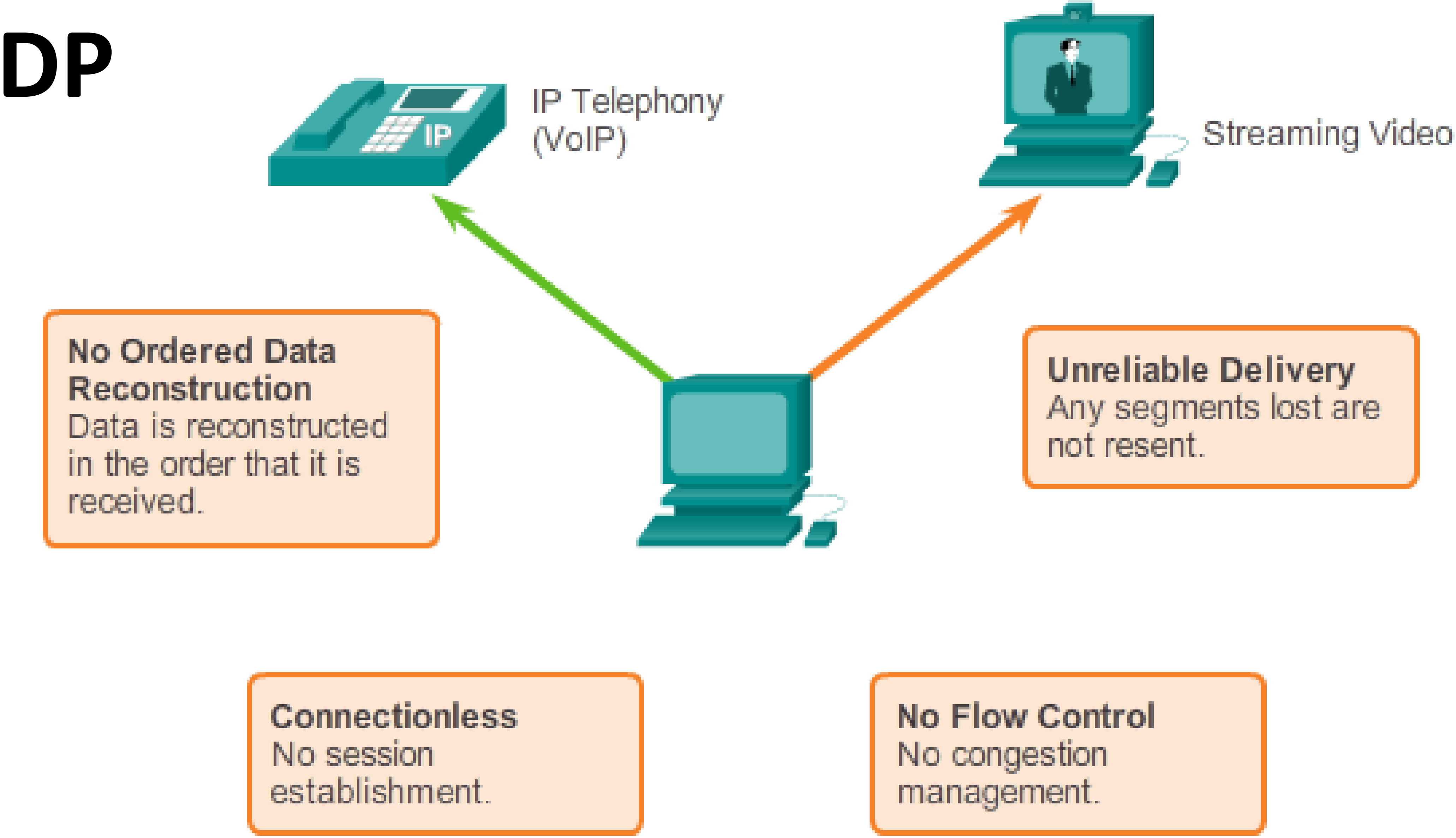
Options (0 or 32 if any)

Application Layer Data (Size varies)

- **Sequence number** (32 bits) - data reassembly purposes.
- **Acknowledgemen**t number (32 bits) - data that has been received.
- **Header length** (4 bits) - ”data offset”. length of the TCP segment header.
- **Reserved** (6 bits) - reserved for the future.
- **Control bits** (6 bits) - Includes bit codes, or flags, that indicate the purpose and function of the TCP segment.
- **Window size** (16 bits) - Indicates the number of segments that can be accepted at one time.
- **Checksum** (16 bits) - Used for error checking of the segment header and data.
- **Urgent** (16 bits) - Indicates if data is urgent.

7.1.2.3 Introducing UDP

UDP



No Ordered Data Reconstruction
Data is reconstructed in the order that it is received.

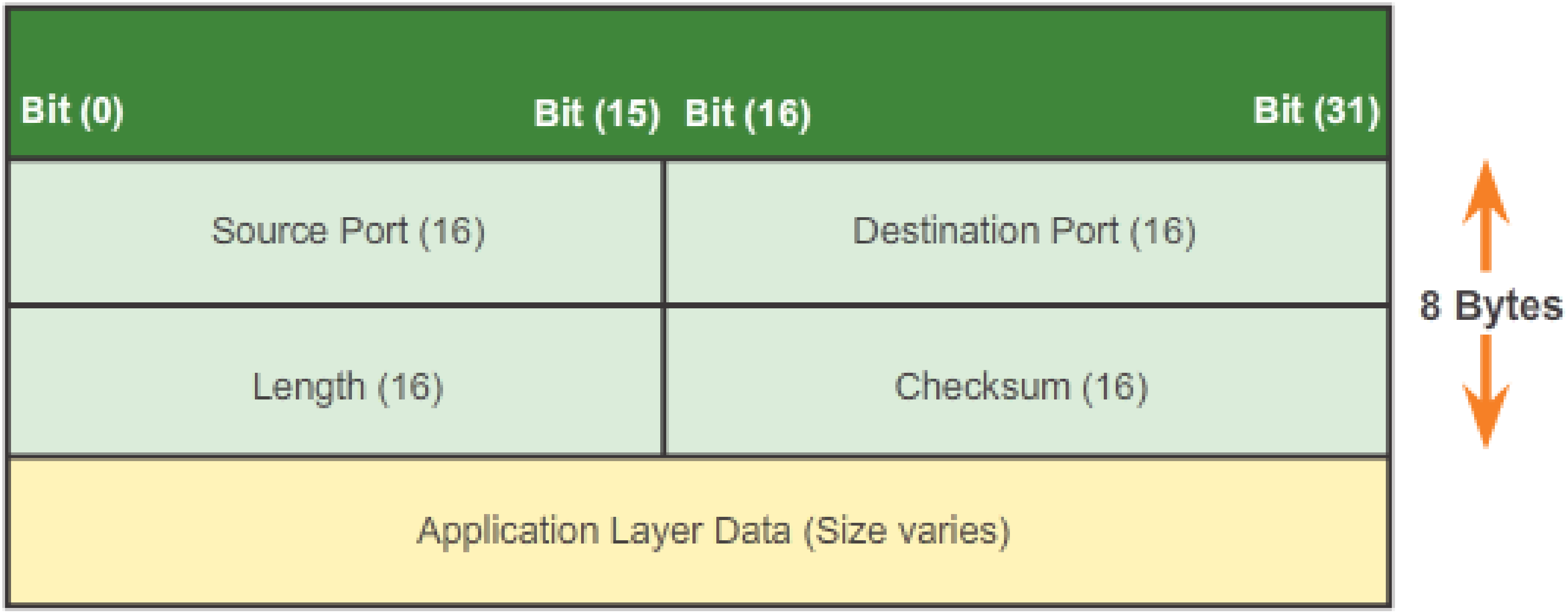
Unreliable Delivery
Any segments lost are not resent.

Connectionless
No session establishment.

No Flow Control
No congestion management.

7.1.2.4 Role of UDP

UDP Datagram

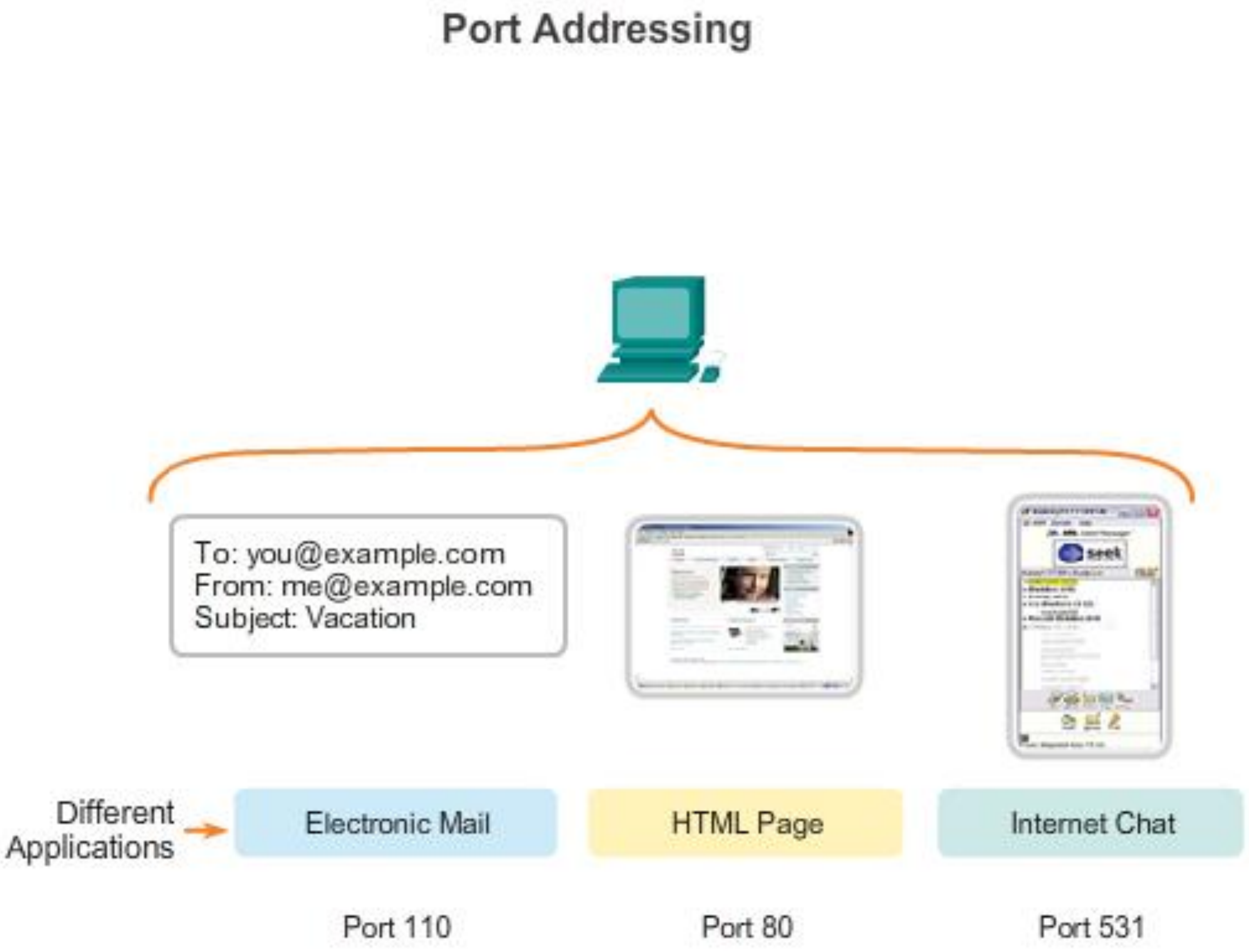


Bit (0) Bit (15)	Bit (16) Bit (31)
Source Port (16)	Destination Port (16)
Length (16)	Checksum (16)
Application Layer Data (Size varies)	

8 Bytes

UDP is a stateless protocol, meaning neither the client, nor the server, is obligated to keep track of the state of the communication session. As shown in the figure, UDP is not concerned with reliability or flow control. Data may be lost or received out of sequence without any UDP mechanisms to recover or reorder the data. **If reliability is required when using UDP as the transport protocol, it must be handled by the application.**

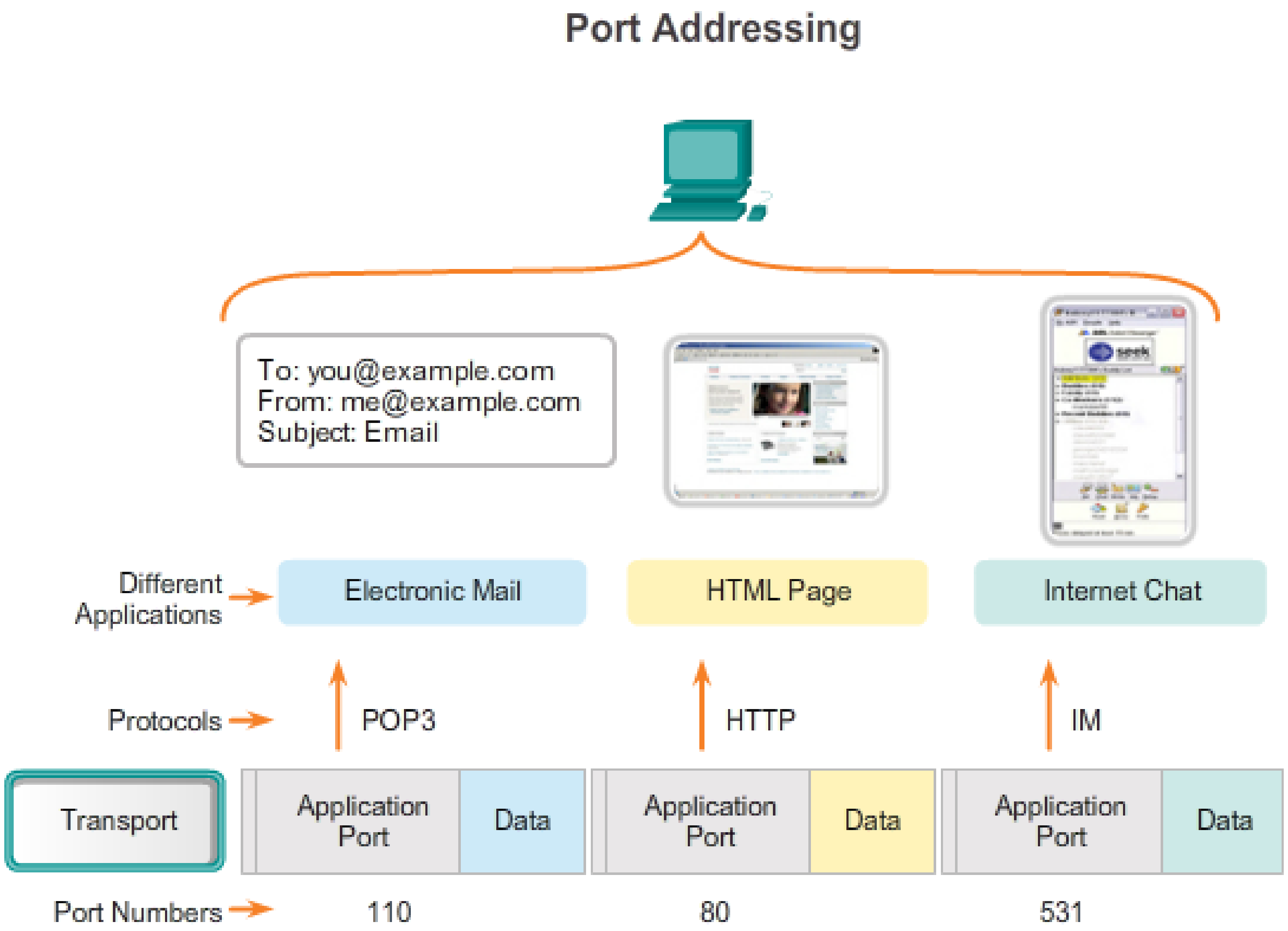
7.1.2.5 Separating Multiple Communications



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

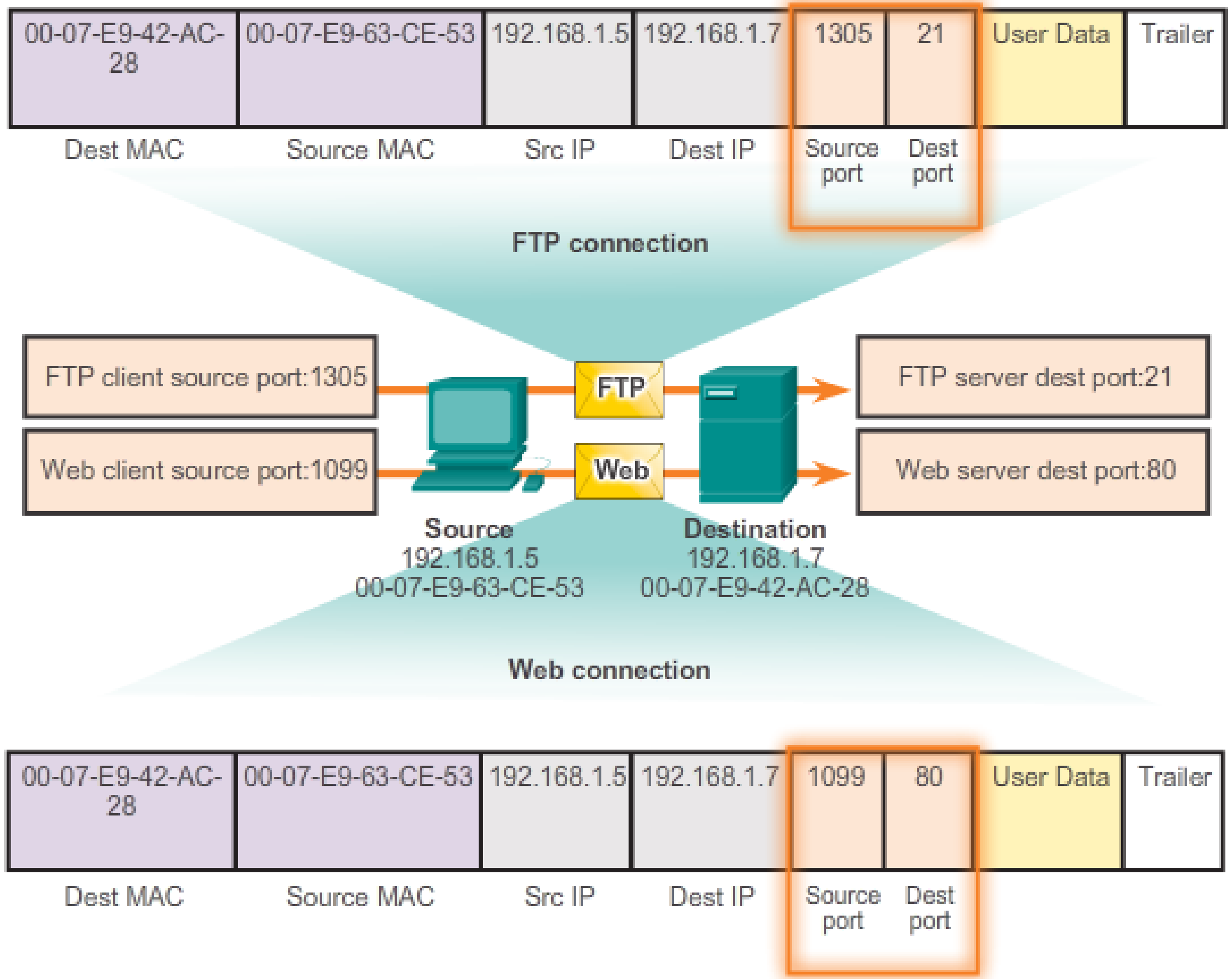
To differentiate the segments and datagrams for each application, both TCP and UDP have header fields that can uniquely identify these applications. These unique identifiers are the port numbers.

7.1.2.6 TCP and UDP Port Addressing



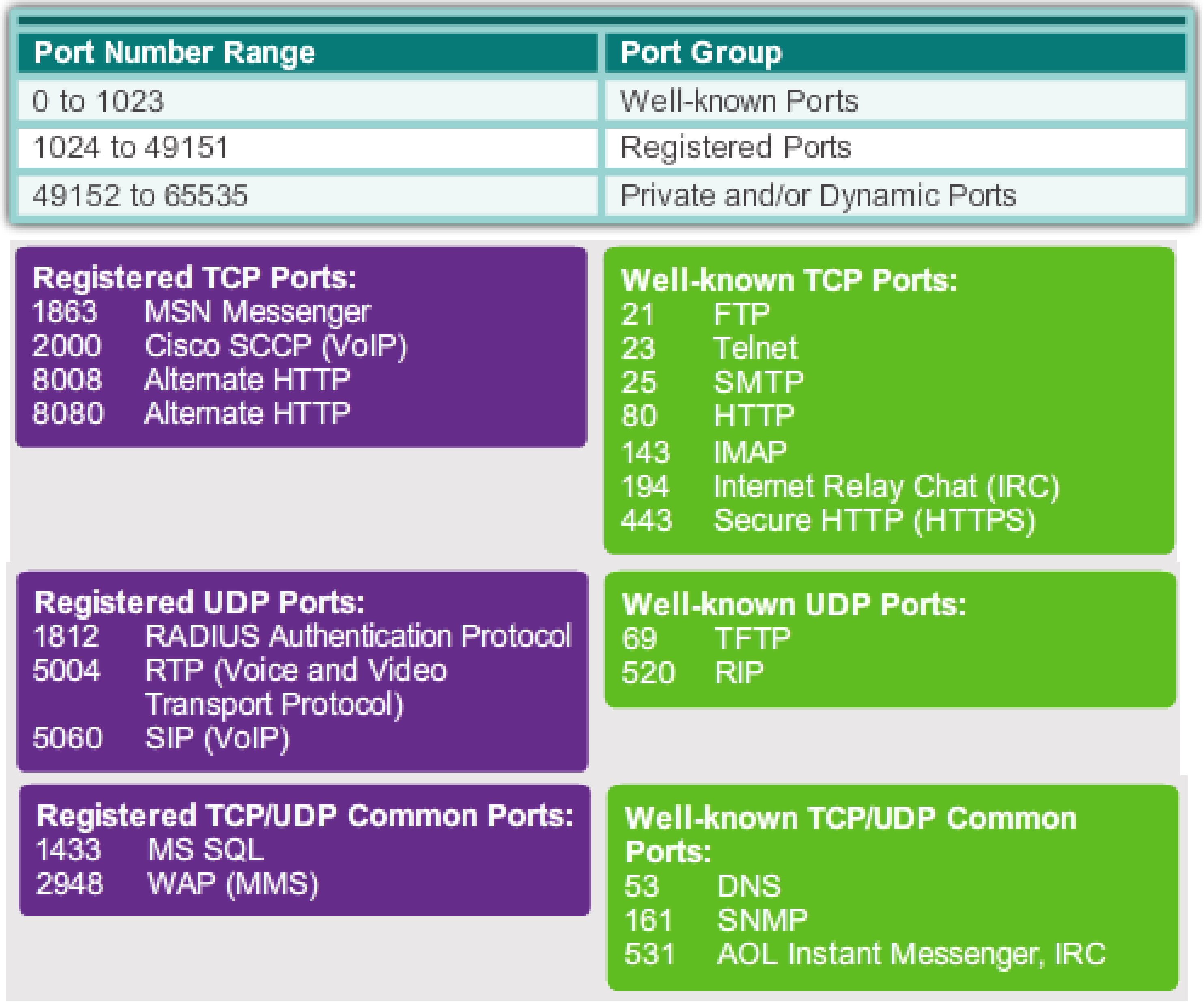
Data for different applications is directed to the correct application because each application has a unique port number.

7.1.2.7 TCP and UDP Port Addressing (Cont.)



- The source port of a client request is randomly generated.
- This port number acts like 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.
- The requesting application port number is used as the destination port number in the response coming back from the server.

7.1.2.8 TCP and UDP Port Addressing (Cont.)



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

Registered TCP Ports:

1863	MSN Messenger
2000	Cisco SCCP (VoIP)
8008	Alternate HTTP
8080	Alternate HTTP

Well-known TCP Ports:

21	FTP
23	Telnet
25	SMTP
80	HTTP
143	IMAP
194	Internet Relay Chat (IRC)
443	Secure HTTP (HTTPS)

Registered UDP Ports:

1812	RADIUS Authentication Protocol
5004	RTP (Voice and Video Transport Protocol)
5060	SIP (VoIP)

Well-known UDP Ports:

69	TFTP
520	RIP

Registered TCP/UDP Common Ports:

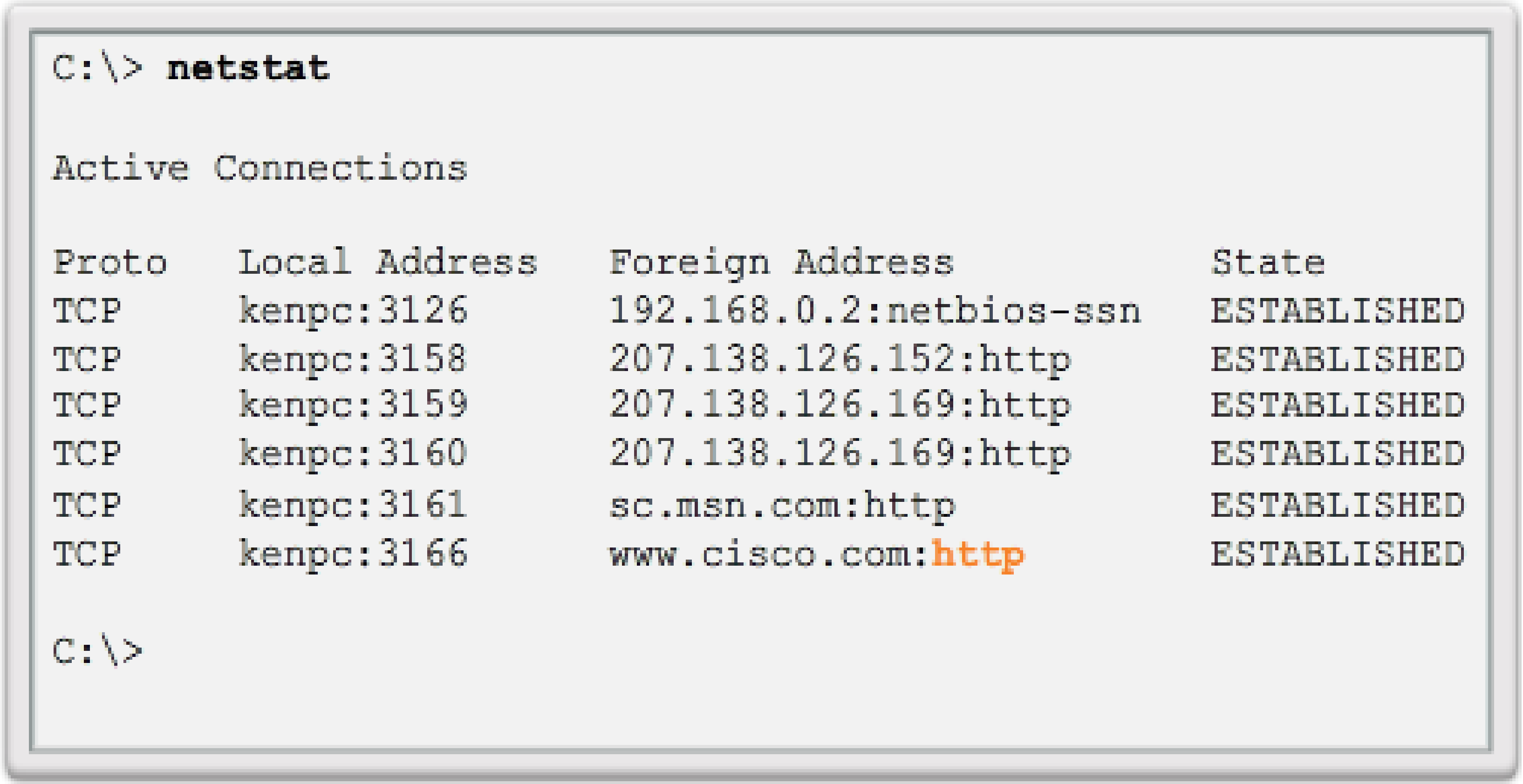
1433	MS SQL
2948	WAP (MMS)

Well-known TCP/UDP Common Ports:

53	DNS
161	SNMP
531	AOL Instant Messenger, IRC

7.1.2.9 TCP and UDP Port Addressing (Cont.)

Netstat Output



```
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:\>
```

Destination Port

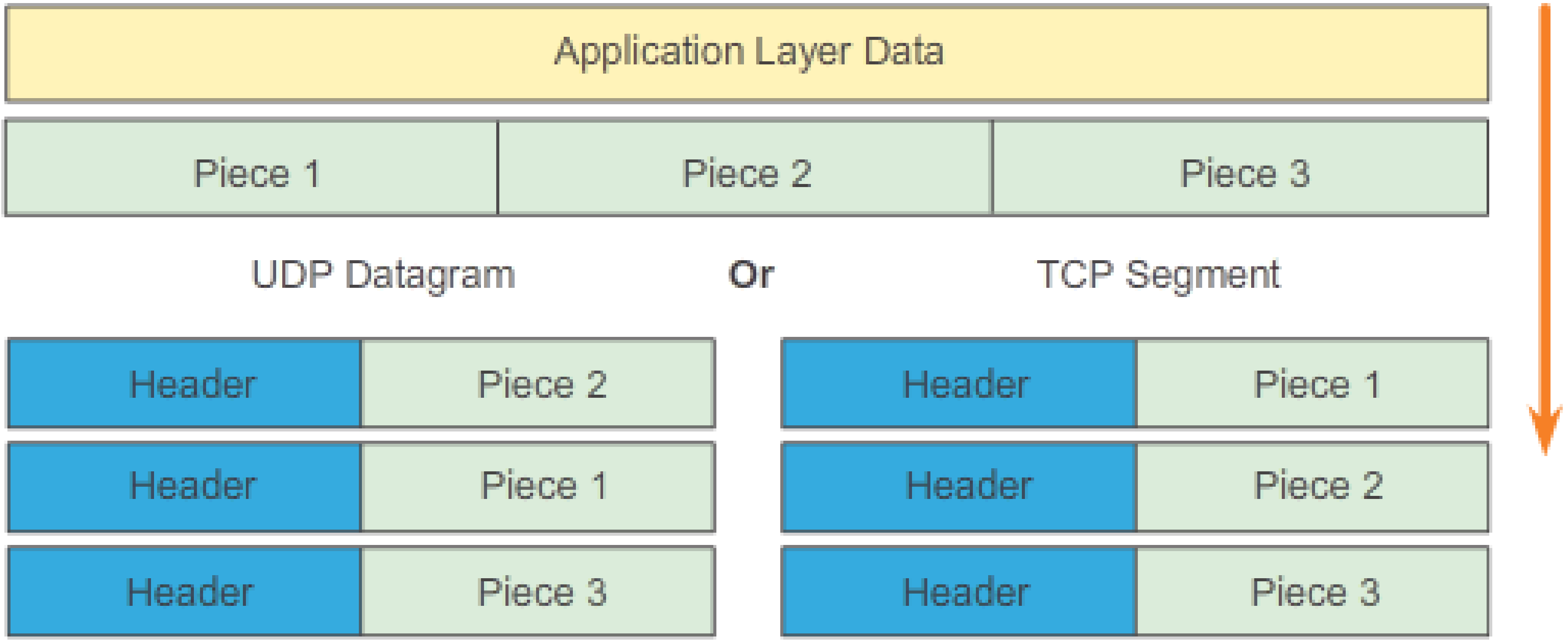
Sometimes it is necessary to know which active TCP connections are open and running on a networked host.

Netstat is an important network utility that can be used to verify those connections.

Netstat lists the protocol in use, the local address and port number, the foreign address and port number, and the connection state.

7.1.2.10 TCP and UDP Segmentation

Transport Layer Functions



Application Layer Data

Piece 1 | Piece 2 | Piece 3

UDP Datagram Or TCP Segment

Header | Piece 2

Header | Piece 1

Header | Piece 3

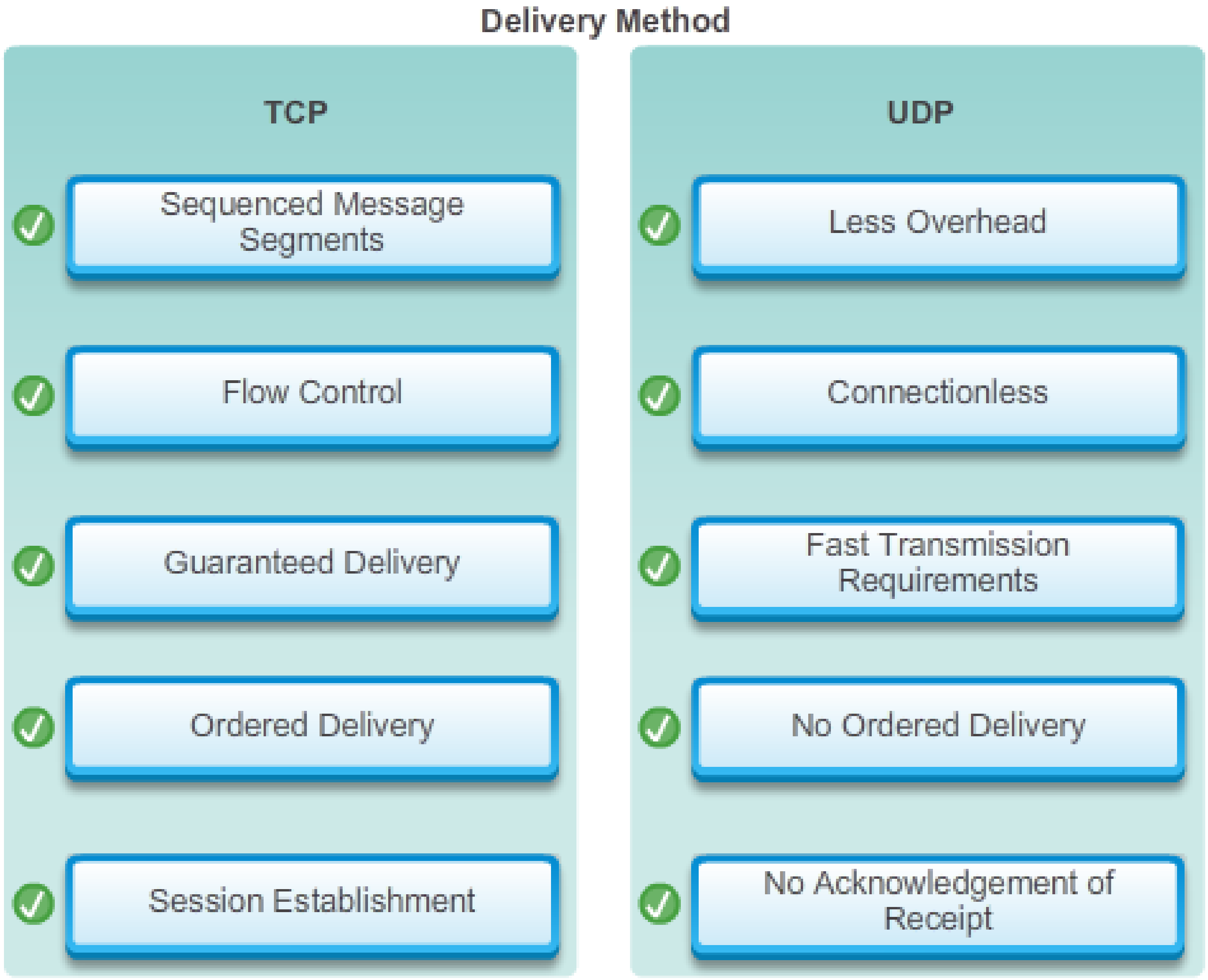
Header | Piece 1

Header | Piece 2

Header | Piece 3

Although services using UDP also track the conversations between applications; they are not concerned with the order in which the information was transmitted or concerned with maintaining a connection. There is no sequence number in the UDP header. UDP is a simpler design and generates less overhead than TCP, resulting in a faster transfer of data.

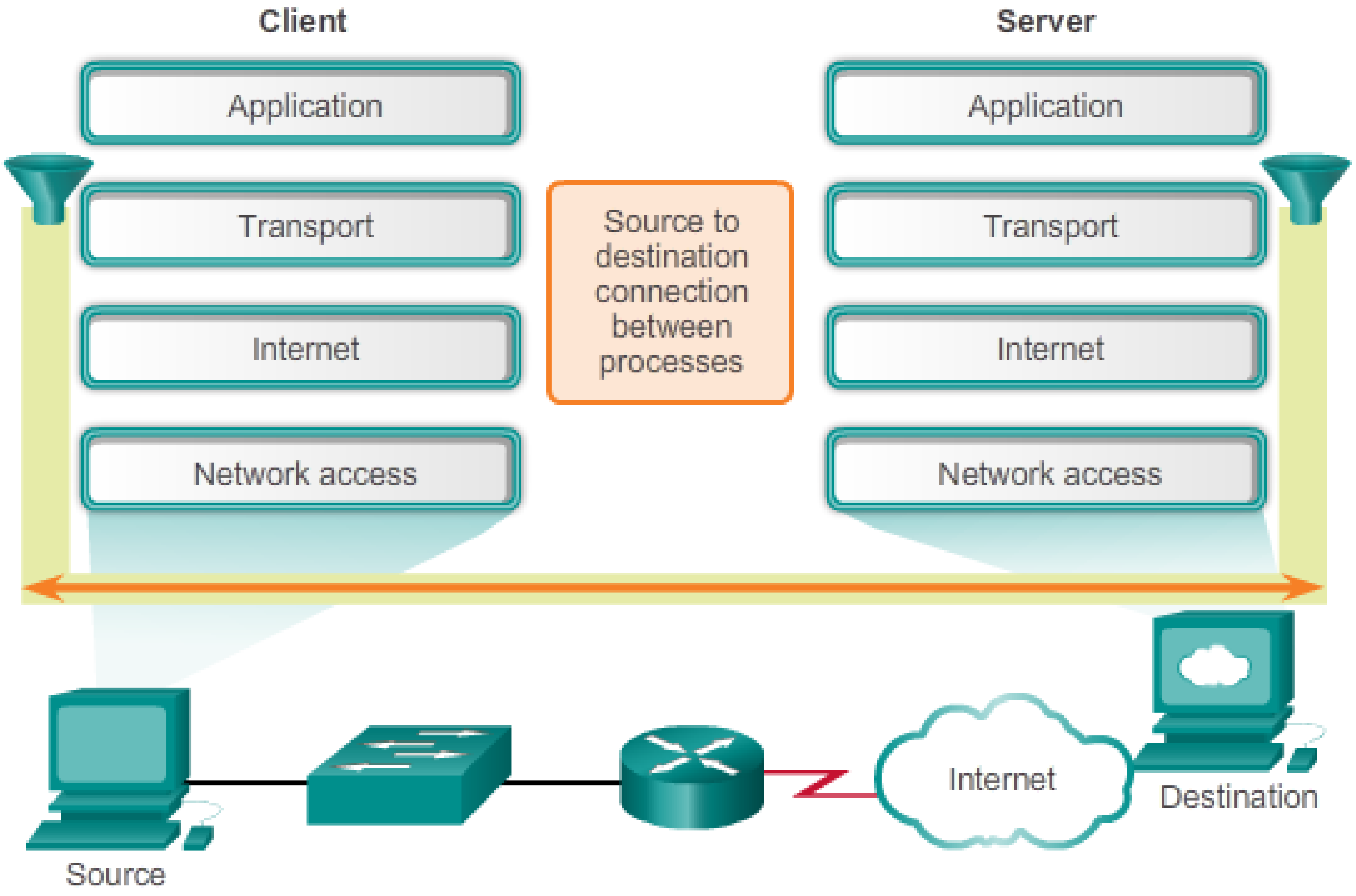
7.1.2.11 Activity - Compare TCP and UDP Characteristics



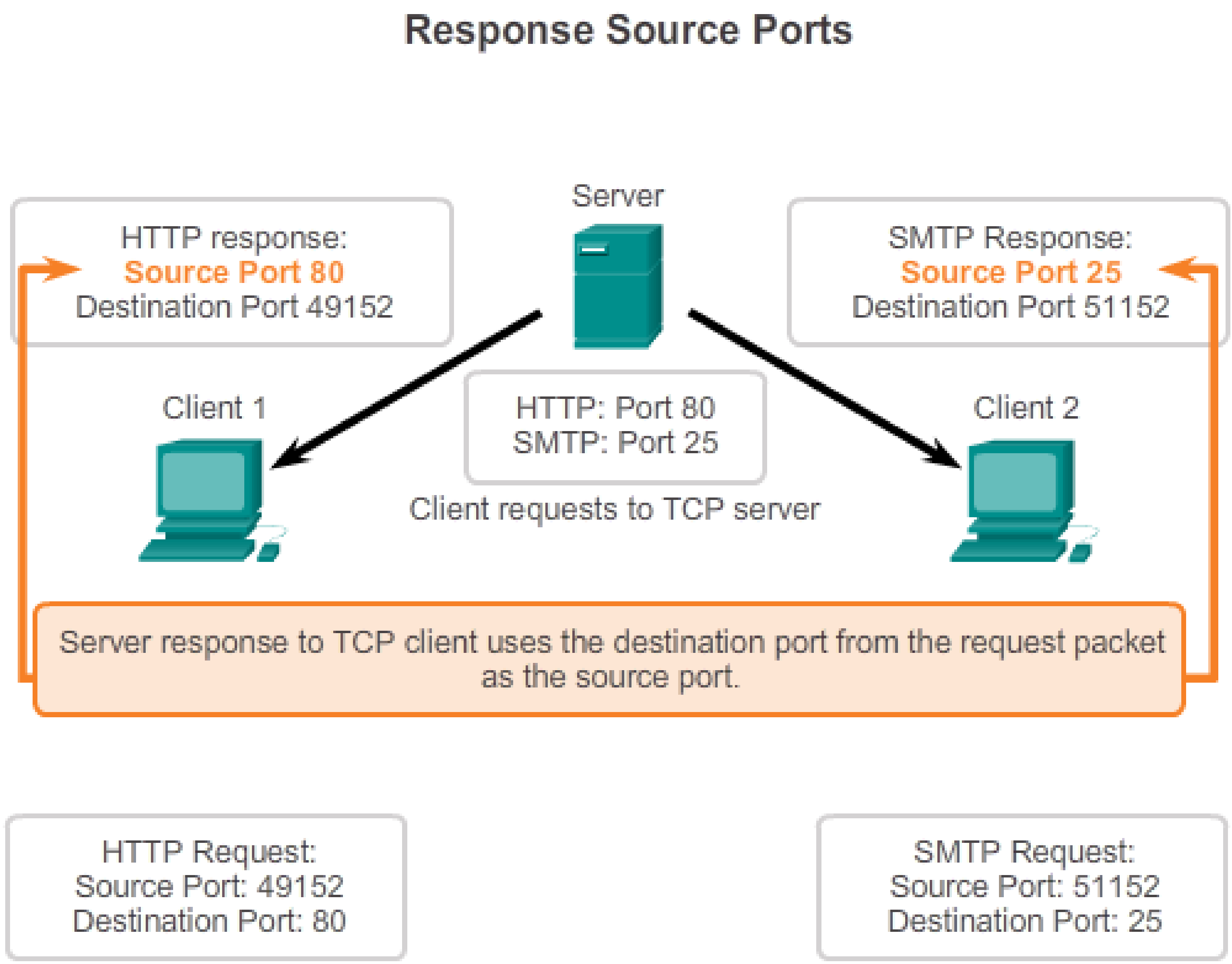
Delivery Method

TCP	UDP
✓ Sequenced Message Segments	✓ Less Overhead
✓ Flow Control	✓ Connectionless
✓ Guaranteed Delivery	✓ Fast Transmission Requirements
✓ Ordered Delivery	✓ No Ordered Delivery
✓ Session Establishment	✓ No Acknowledgement of Receipt

7.2.1.1 TCP Reliable Delivery



7.2.1.2 TCP Server Processes

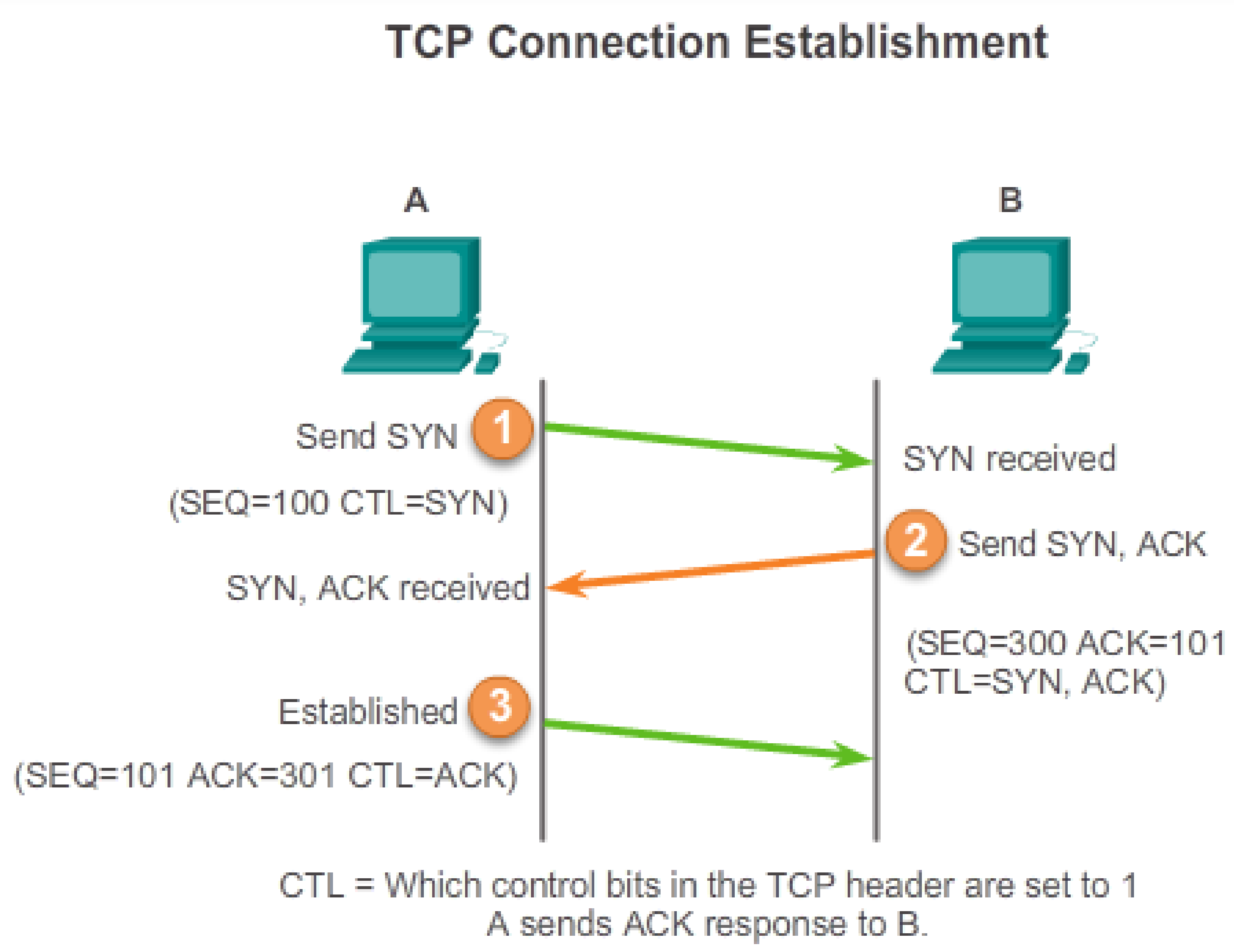


. Any incoming client request addressed to the correct socket is accepted and the data is passed to the server application.

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

It is common for a server to provide more than one service at the same time, such as a web server and an FTP server

7.2.1.3 TCP Connection Establishment and Termination

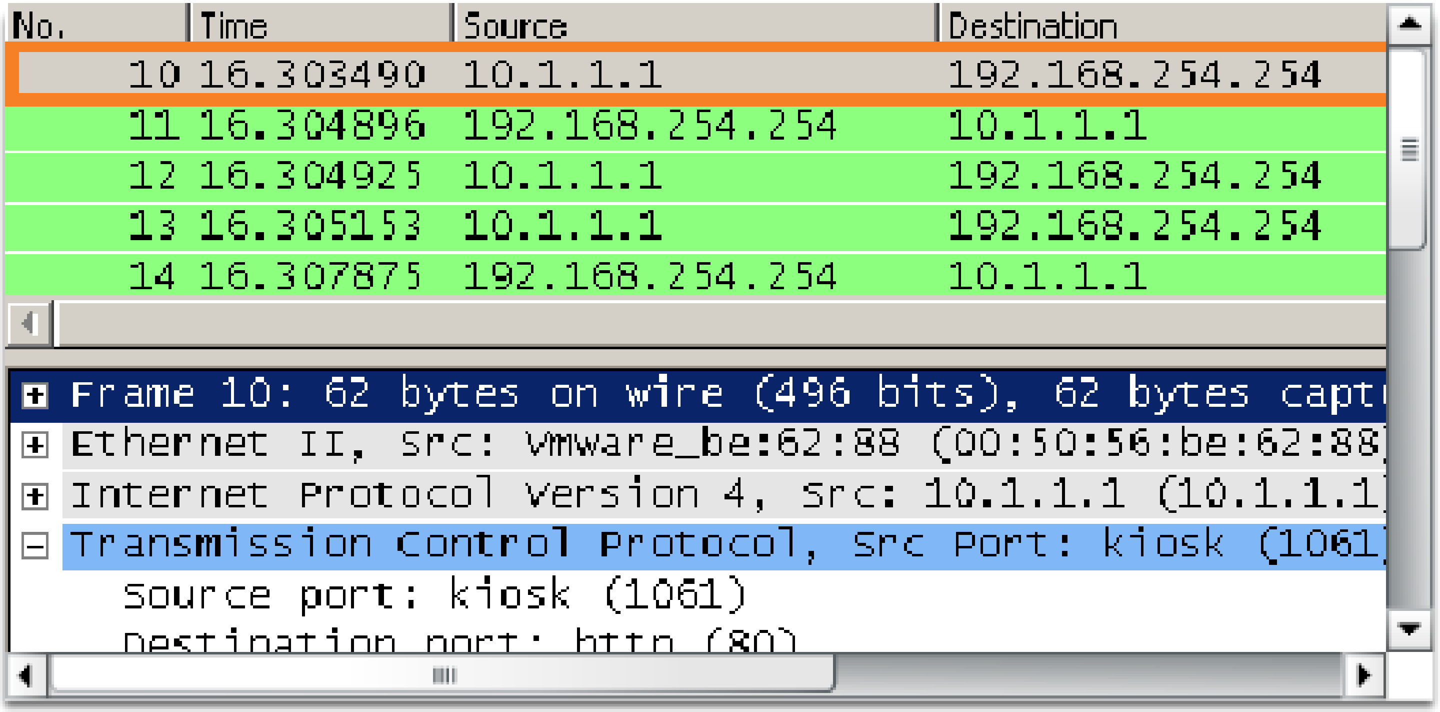


CTL = Which control bits in the TCP header are set to 1
A sends ACK response to B.

- URG - Urgent pointer field significant
- ACK - Acknowledgement field significant
- PSH - Push function
- RST - Reset the connection
- SYN - Synchronize sequence numbers
- FIN - No more data from sender
- The ACK and SYN fields are relevant to our analysis of the three-way handshake

7.2.1.4 TCP Three-way Handshake Analysis - Step 1

TCP 3-Way Handshake (SYN)



```
No.     Time        Source            Destination
     10 16.303490   10.1.1.1          192.168.254.254
     11 16.304896   192.168.254.254   10.1.1.1
     12 16.304925   10.1.1.1          192.168.254.254
     13 16.305153   10.1.1.1          192.168.254.254
     14 16.307875   192.168.254.254   10.1.1.1

⊞ Frame 10: 62 bytes on wire (496 bits), 62 bytes captu
⊞ Ethernet II, Src: Vmware_be:62:88 (00:50:56:be:62:88)
⊞ Internet Protocol Version 4, Src: 10.1.1.1 (10.1.1.1)
⊟ Transmission Control Protocol, Src Port: kiosk (1061)
    Source port: kiosk (1061)
    Destination port: http (80)
```

A protocol analyzer shows initial client request for session in frame 10

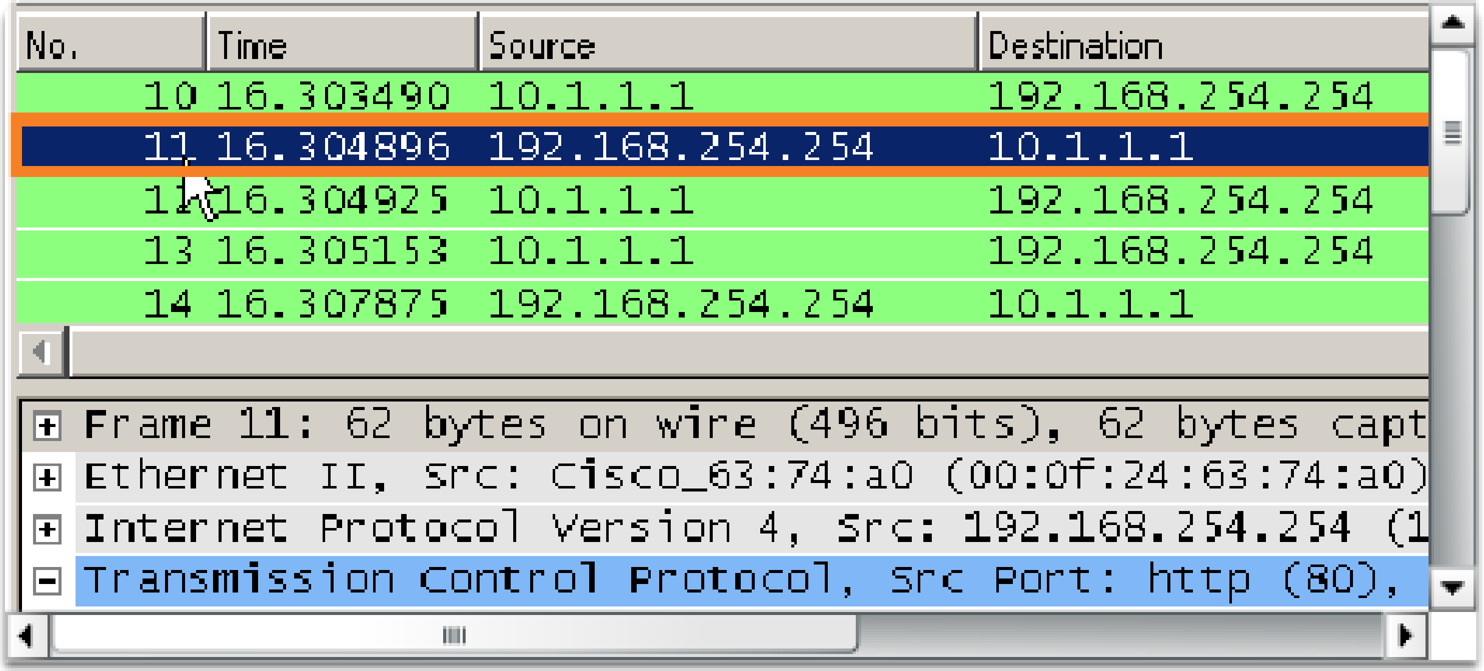
TCP segment in this frame shows:

- SYN flag set to validate an Initial Sequence Number
- Randomized sequence number valid (relative value is 0)
- Random source port 1061
- Well-known destination port is 80 (HTTP port) indicates web server (httpd)

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

7.2.1.5 TCP Three-way Handshake Analysis - Step 2

TCP 3-Way Handshake (SYN, ACK)



No.	Time	Source	Destination
10	16.303490	10.1.1.1	192.168.254.254
11	16.304896	192.168.254.254	10.1.1.1
12	16.304925	10.1.1.1	192.168.254.254
13	16.305153	10.1.1.1	192.168.254.254
14	16.307875	192.168.254.254	10.1.1.1

```
⊞ Frame 11: 62 bytes on wire (496 bits), 62 bytes capt
⊞ Ethernet II, Src: Cisco_63:74:a0 (00:0f:24:63:74:a0)
⊞ Internet Protocol Version 4, Src: 192.168.254.254 (1
⊟ Transmission Control Protocol, Src Port: http (80),
```

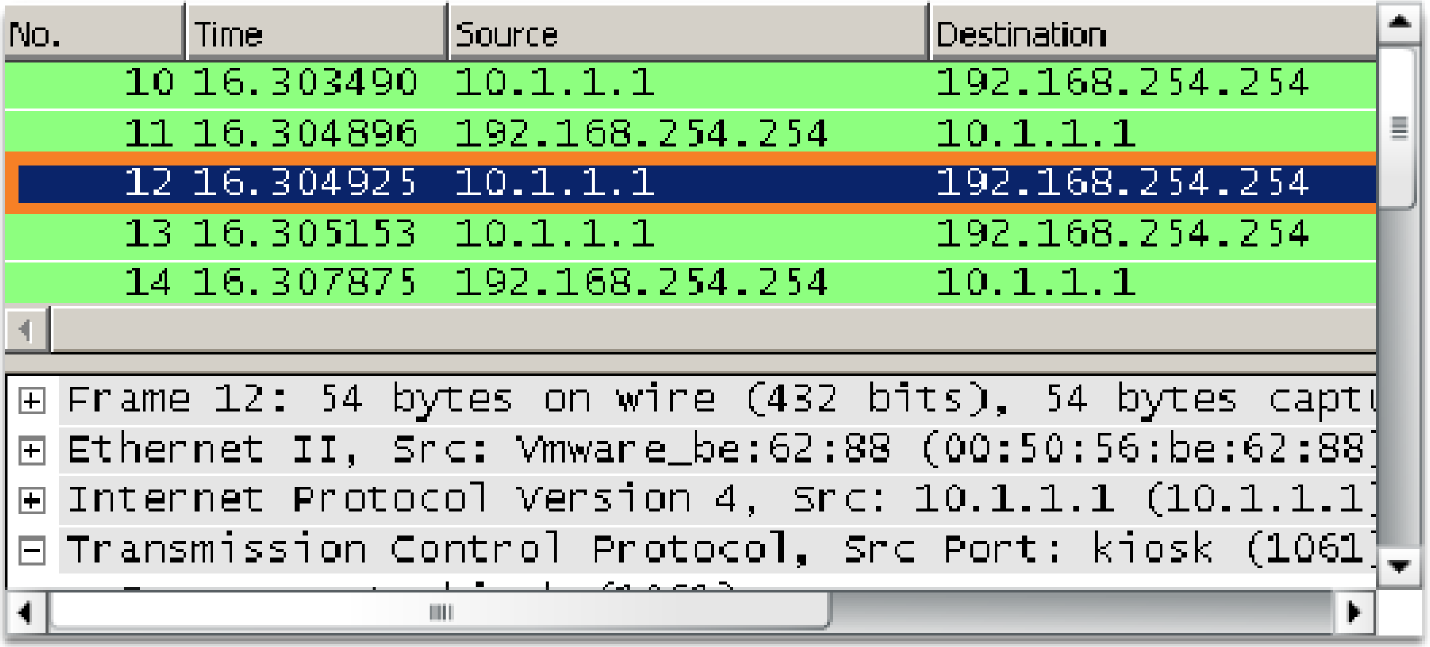
A protocol analyzer shows server response in frame 11

- ACK flag set to indicate a valid Acknowledgement number
- Acknowledgement number response to initial sequence number as relative value of 1
- SYN flag set to indicate the Initial Sequence Number for the server to client session
- Destination port number of 1061 to corresponding to the clients source port
- Source port number of 80 (HTTP) indicating the web server service (httpd)

Step 2: The server acknowledges the client-to-server communication session and requests a server-to-client communication session.

7.2.1.6 TCP Three-way Handshake Analysis - Step 3

TCP 3-Way Handshake (ACK)



```
No.   Time        Source            Destination
   10 16.303490   10.1.1.1          192.168.254.254
   11 16.304896   192.168.254.254   10.1.1.1
   12 16.304925   10.1.1.1          192.168.254.254
   13 16.305153   10.1.1.1          192.168.254.254
   14 16.307875   192.168.254.254   10.1.1.1
```

```
⊞ Frame 12: 54 bytes on wire (432 bits), 54 bytes captu
⊞ Ethernet II, Src: Vmware_be:62:88 (00:50:56:be:62:88]
⊞ Internet Protocol Version 4, Src: 10.1.1.1 (10.1.1.1]
⊟ Transmission Control Protocol, Src Port: kiosk (1061]
```

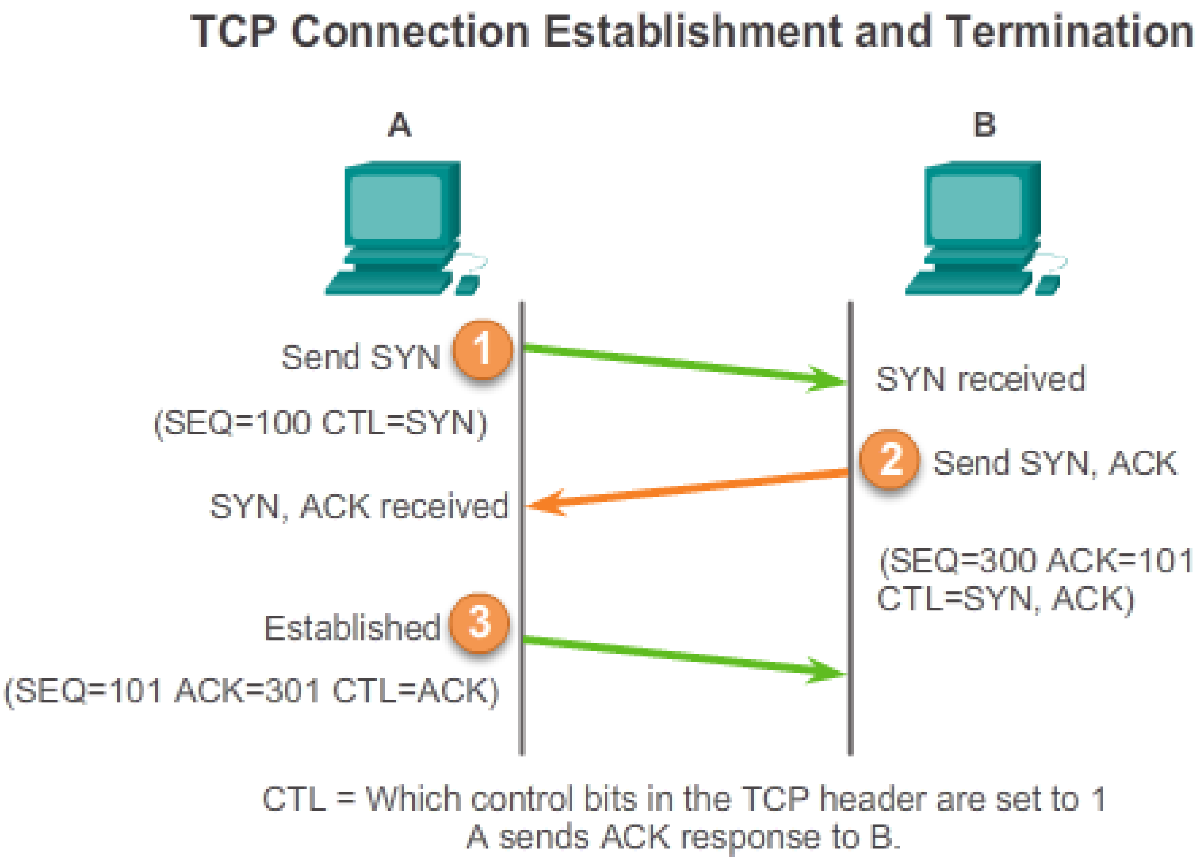
A protocol analyzer shows client response to session in frame 12

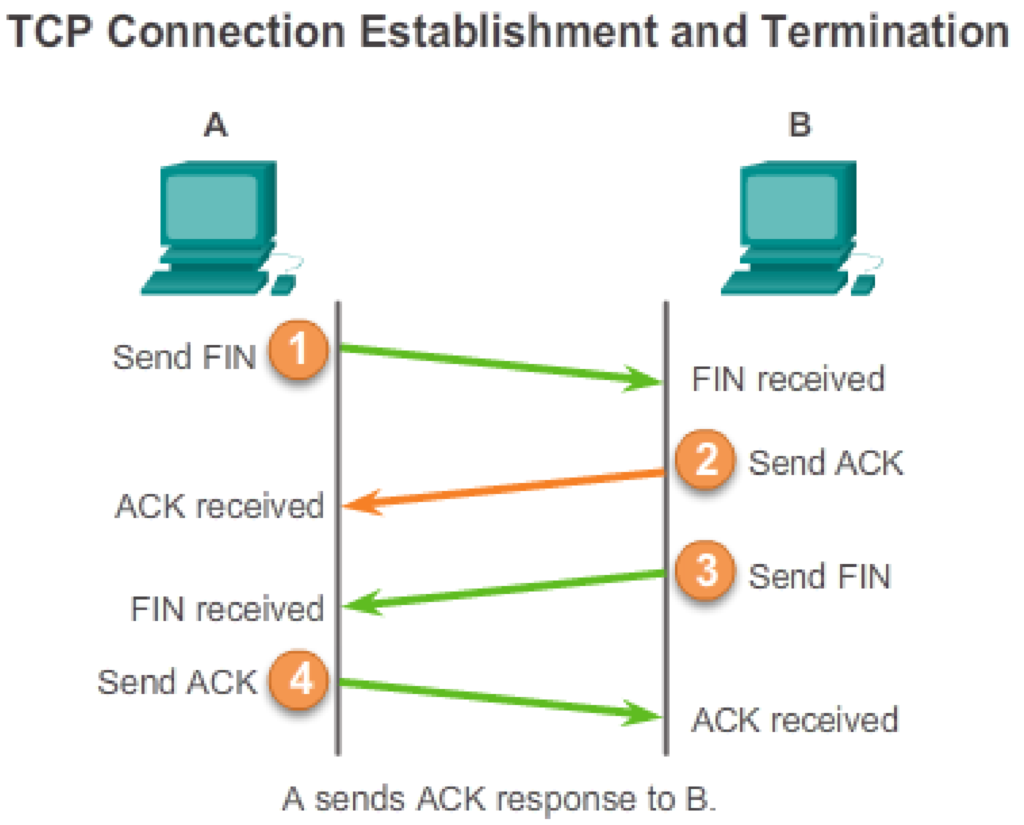
The TCP segment in this frame shows:

- ACK flag set to indicate a valid Acknowledgement number
- Acknowledgement number response to initial sequence number as relative value of 1
- Source port number of 1061 to corresponding
- Destination port number of 80 (HTTP) indicating the web server service (httpd)

Step 3: The initiating client acknowledges the server-to-client communication session.

7.2.1.7 TCP Session Termination Analysis


TCP Connection Establishment and Termination
A
B
Send SYN
1
(SEQ=100 CTL=SYN)
SYN received
2
Send SYN, ACK
SYN, ACK received
(SEQ=300 ACK=101
CTL=SYN, ACK)
Established
3
(SEQ=101 ACK=301 CTL=ACK)
CTL = Which control bits in the TCP header are set to 1
A sends ACK response to B.

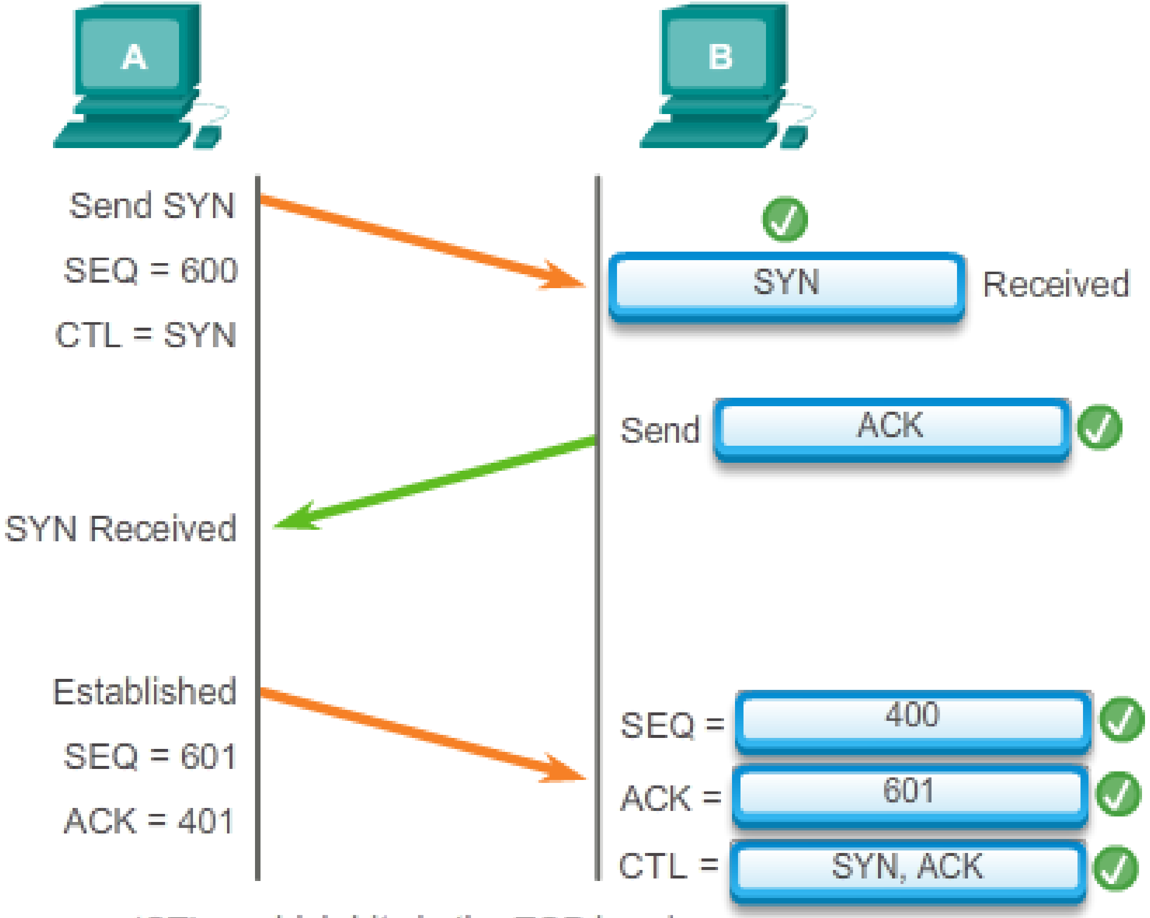


To close a connection, the Finish (FIN) control flag must be set in the segment header. To end each one-way TCP session, a two-way handshake is used, consisting of a FIN segment and an ACK segment. Therefore, to terminate a single conversation supported by TCP, four exchanges are needed to end both sessions, as shown in Figure 1

7.2.1.8 Lab - Using Wireshark to Observe the TCP 3-Way Handshake

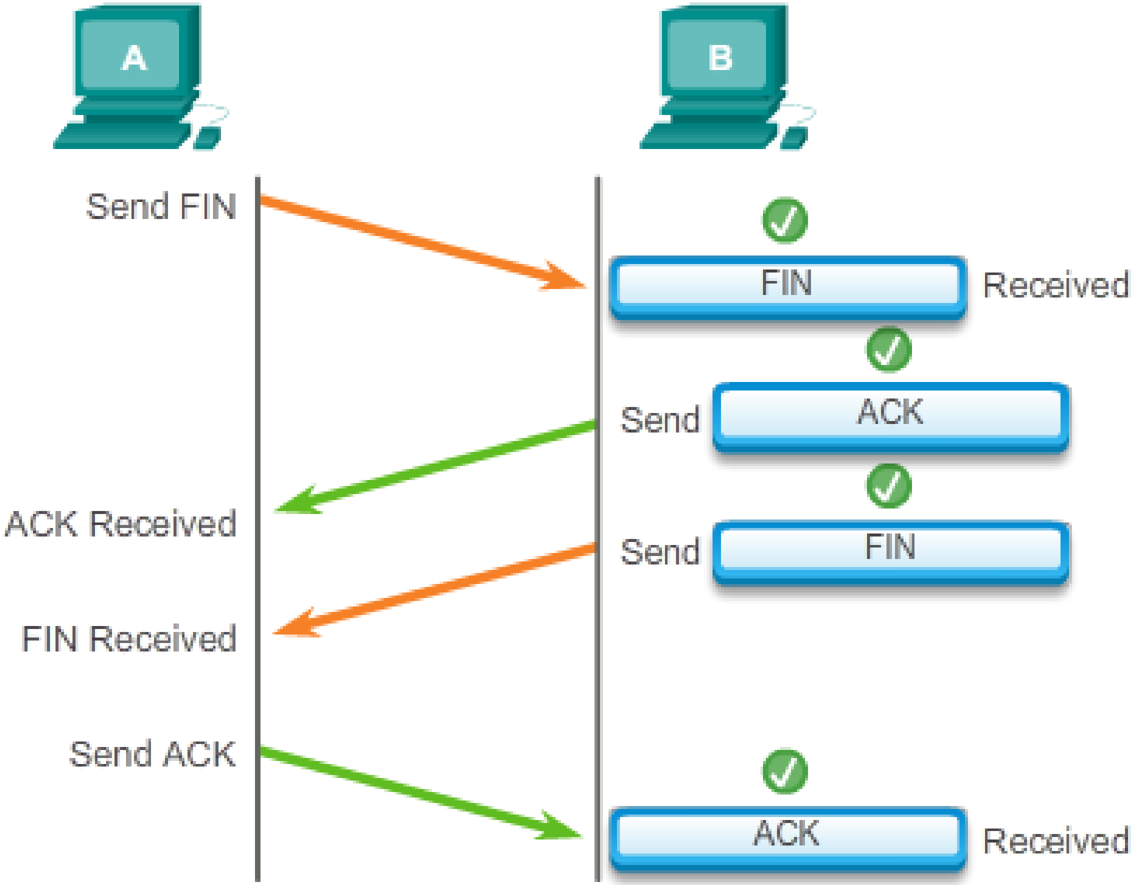
7.2.1.9 Activity - TCP Connection and Termination Process

3-Way Handshake of TCP Establishment Session

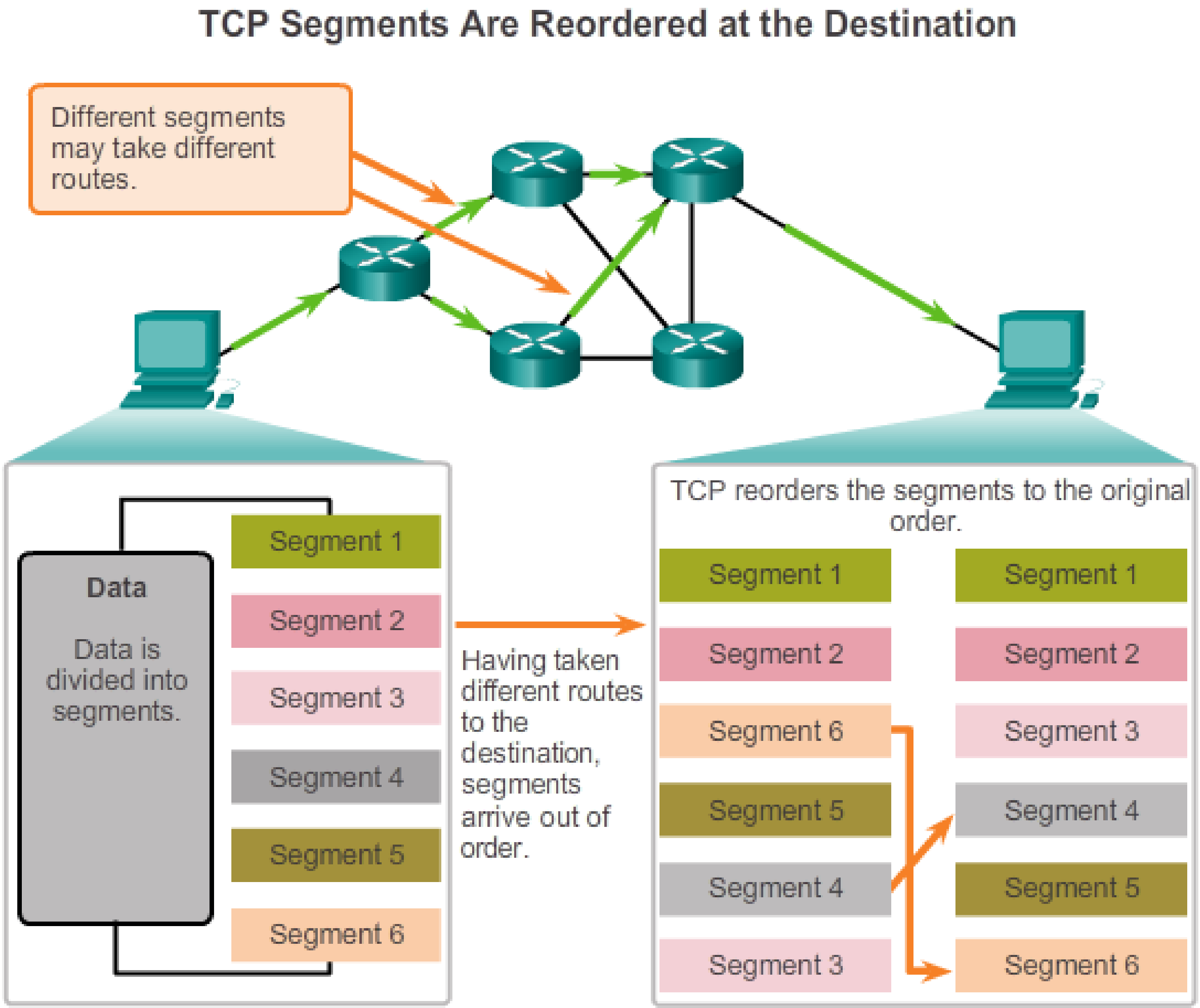


(CTL = which bits in the TCP header are set to 1)

4 Step Process of TCP Termination Session

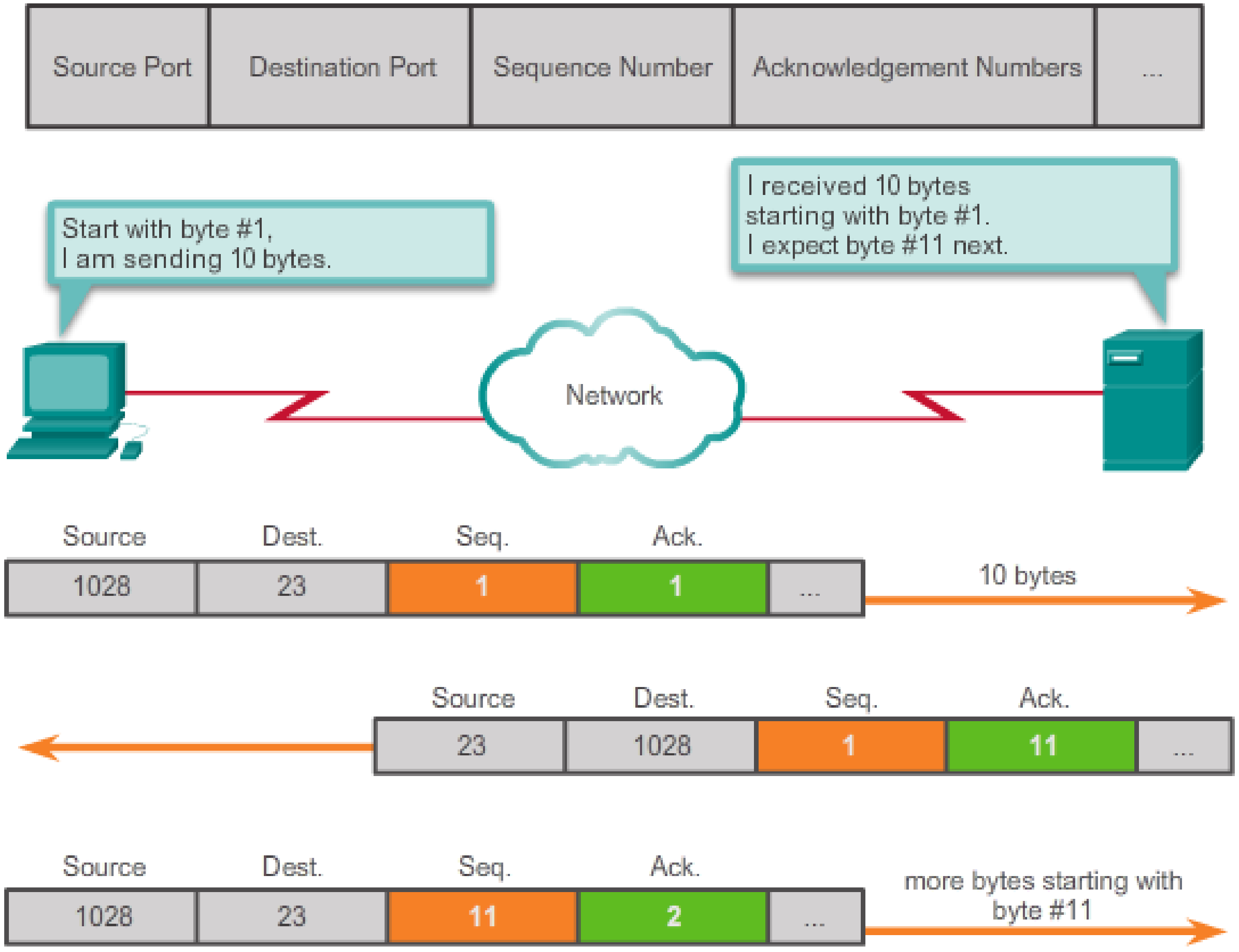


7.2.2.1 TCP Reliability – Ordered Delivery


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

7.2.2.2 TCP Reliability – Acknowledgement and Window Size

Acknowledgement of TCP Segments



Source Port	Destination Port	Sequence Number	Acknowledgement Numbers	...

Start with byte #1,
I am sending 10 bytes.

I received 10 bytes
starting with byte #1.
I expect byte #11 next.

Source	Dest.	Seq.	Ack.	
1028	23	1	1	...

10 bytes

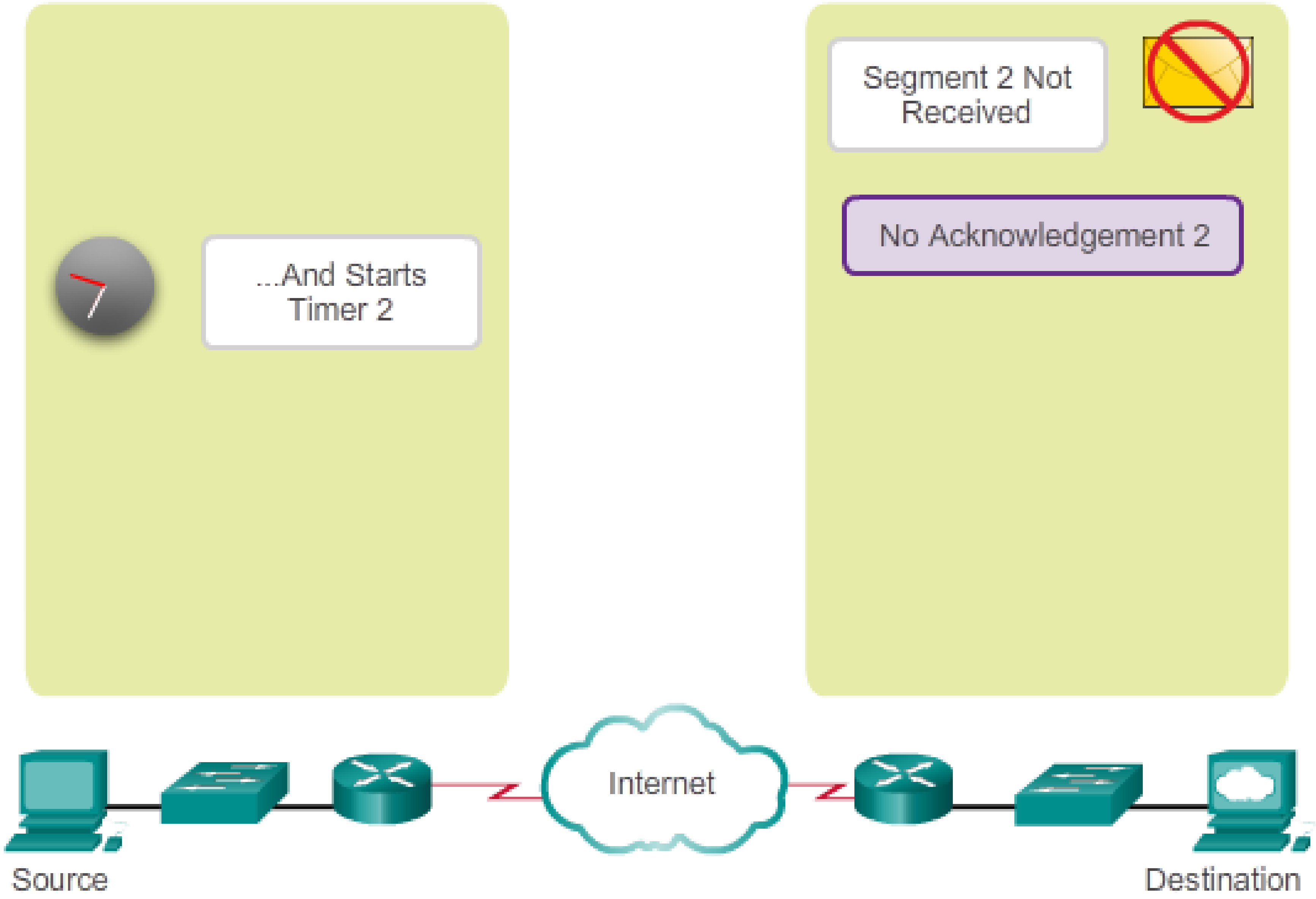
Source	Dest.	Seq.	Ack.	
23	1028	1	11	...

Source	Dest.	Seq.	Ack.	
1028	23	11	2	...

more bytes starting with
byte #11

The amount of data that a source can transmit before an acknowledgement must be received is called the window size, which is a field in the TCP header that enables the management of lost data and flow control.

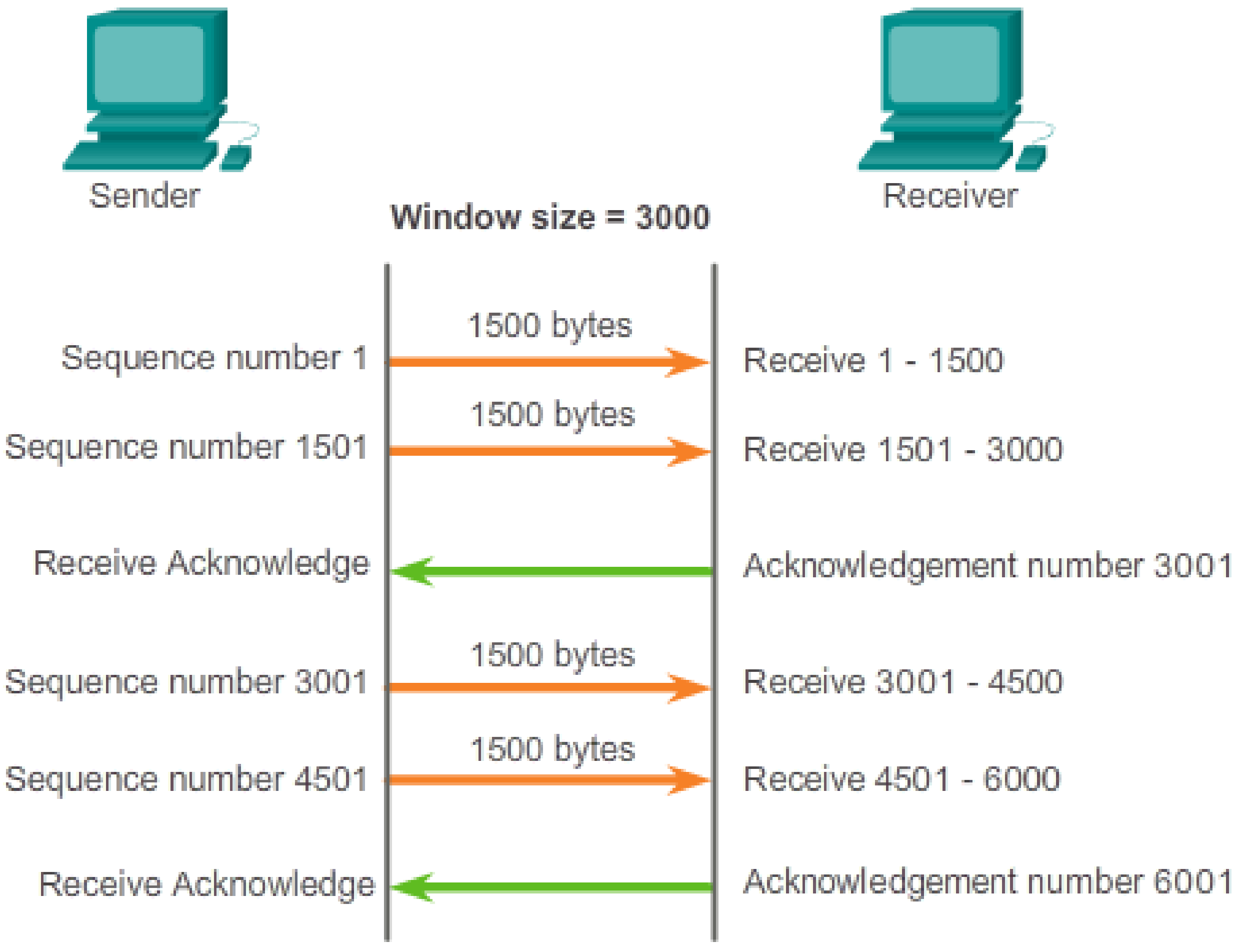
7.2.2.3 TCP Reliability - Data Loss and Retransmission



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

7.2.2.4 TCP Flow Control – Window Size and Acknowledgements

TCP Segment Acknowledgement and Window Size



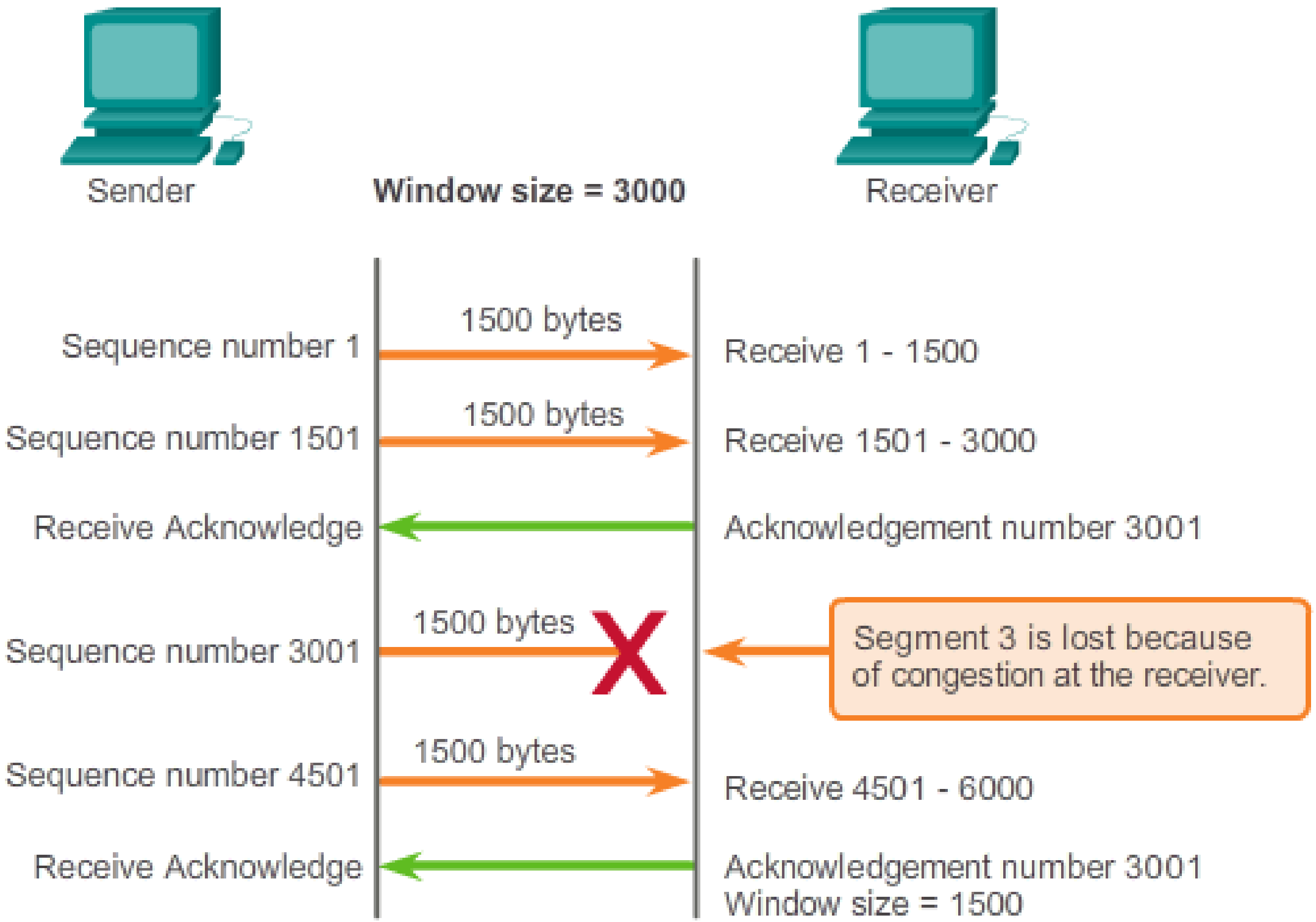
The window size determines the number of bytes sent before an acknowledgment is expected.

The acknowledgement number is the number of the next expected byte.

TCP uses window sizes to attempt to manage the rate of transmission to the maximum flow that the network and destination device can support, while minimizing loss and retransmissions.

7.2.2.5 TCP Flow Control - Congestion Avoidance

TCP Congestion and Flow Control



Sender

Window size = 3000

Receiver

Sequence number 1 — 1500 bytes → Receive 1 - 1500

Sequence number 1501 — 1500 bytes → Receive 1501 - 3000

Receive Acknowledge ← Acknowledgement number 3001

Sequence number 3001 — 1500 bytes X

Segment 3 is lost because of congestion at the receiver.

Sequence number 4501 — 1500 bytes → Receive 4501 - 6000

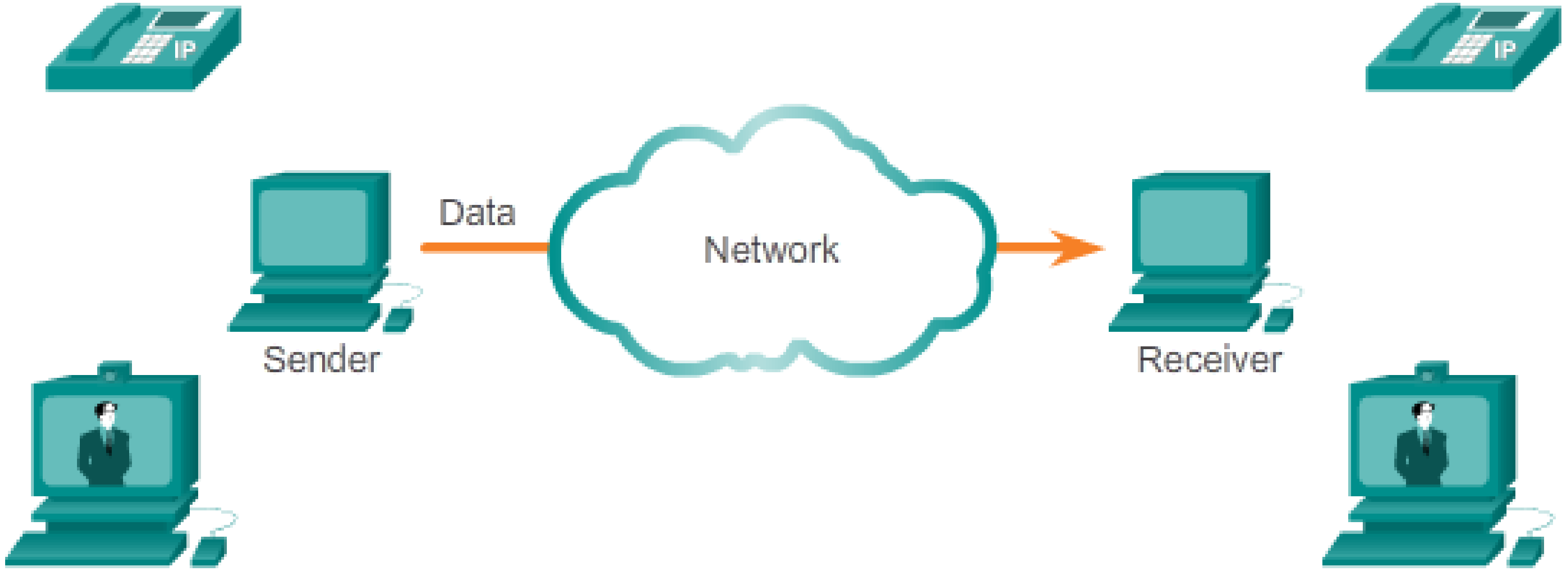
Receive Acknowledge ← Acknowledgement number 3001
Window size = 1500

If segments are lost because of congestion, the receiver will acknowledge the last received sequential segment and reply with a reduced window size.

This dynamic increasing and decreasing of window size is a continuous process in TCP. In highly efficient networks, window sizes may become very large because data is not lost. In networks where the underlying infrastructure is under stress, the window size likely remains small.

7.2.3.1 UDP Low Overhead versus Reliability

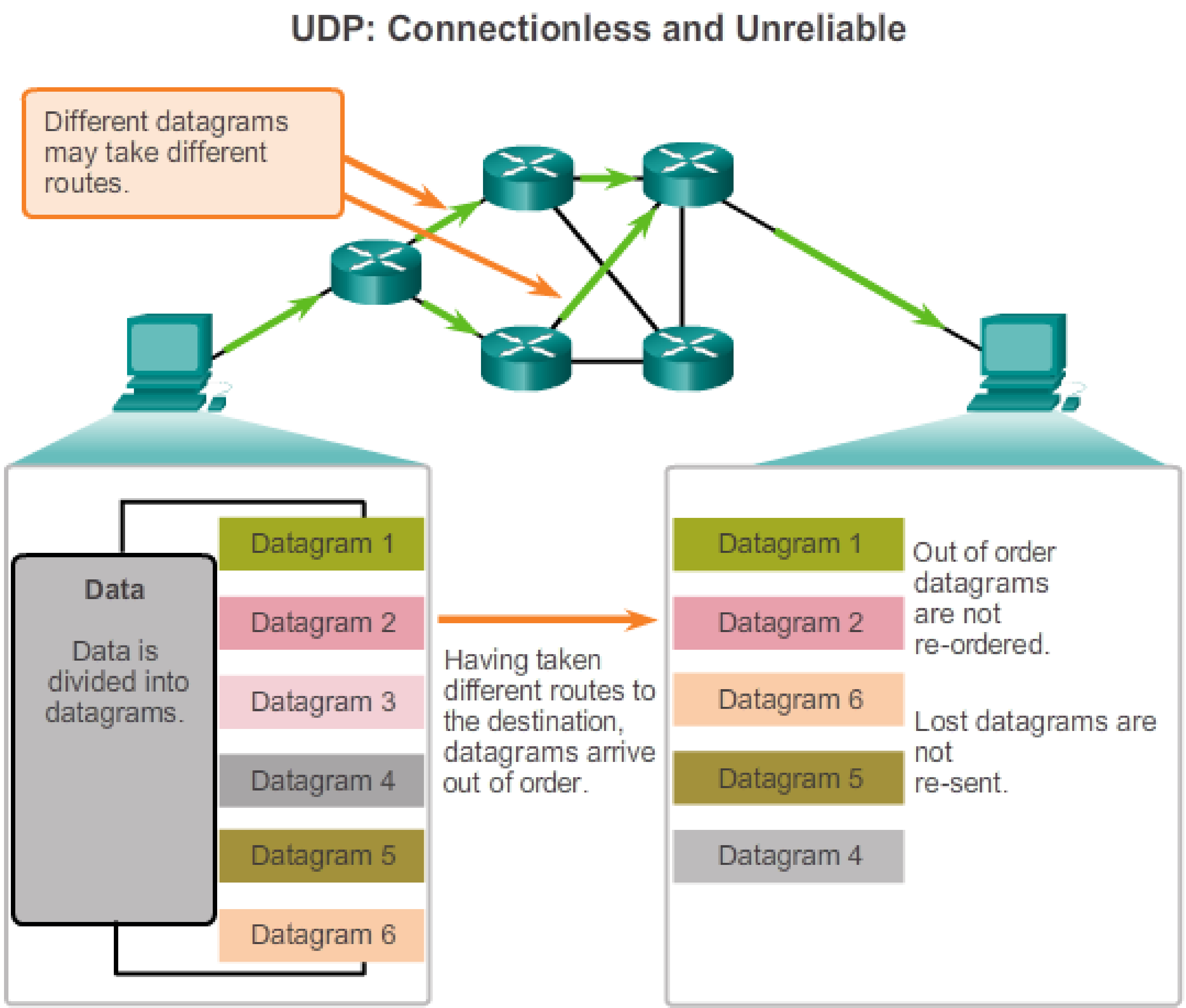
UDP Low Overhead Data Transport



UDP does not establish a connection before sending data.

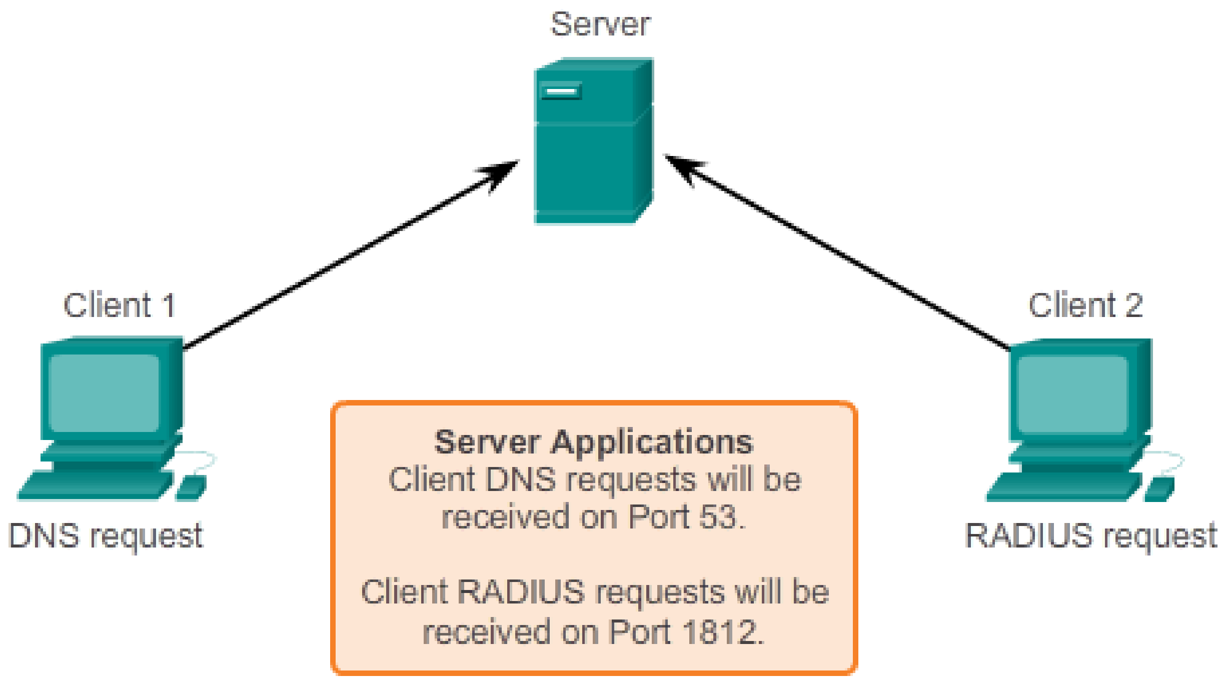
UDP provides low overhead data transport because it has a small datagram header and no network management traffic.

7.2.3.2 UDP Datagram Reassembly


UDP: Connectionless and Unreliable
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.

7.2.3.3 UDP Server Processes and Requests

UDP Server Listening for Requests

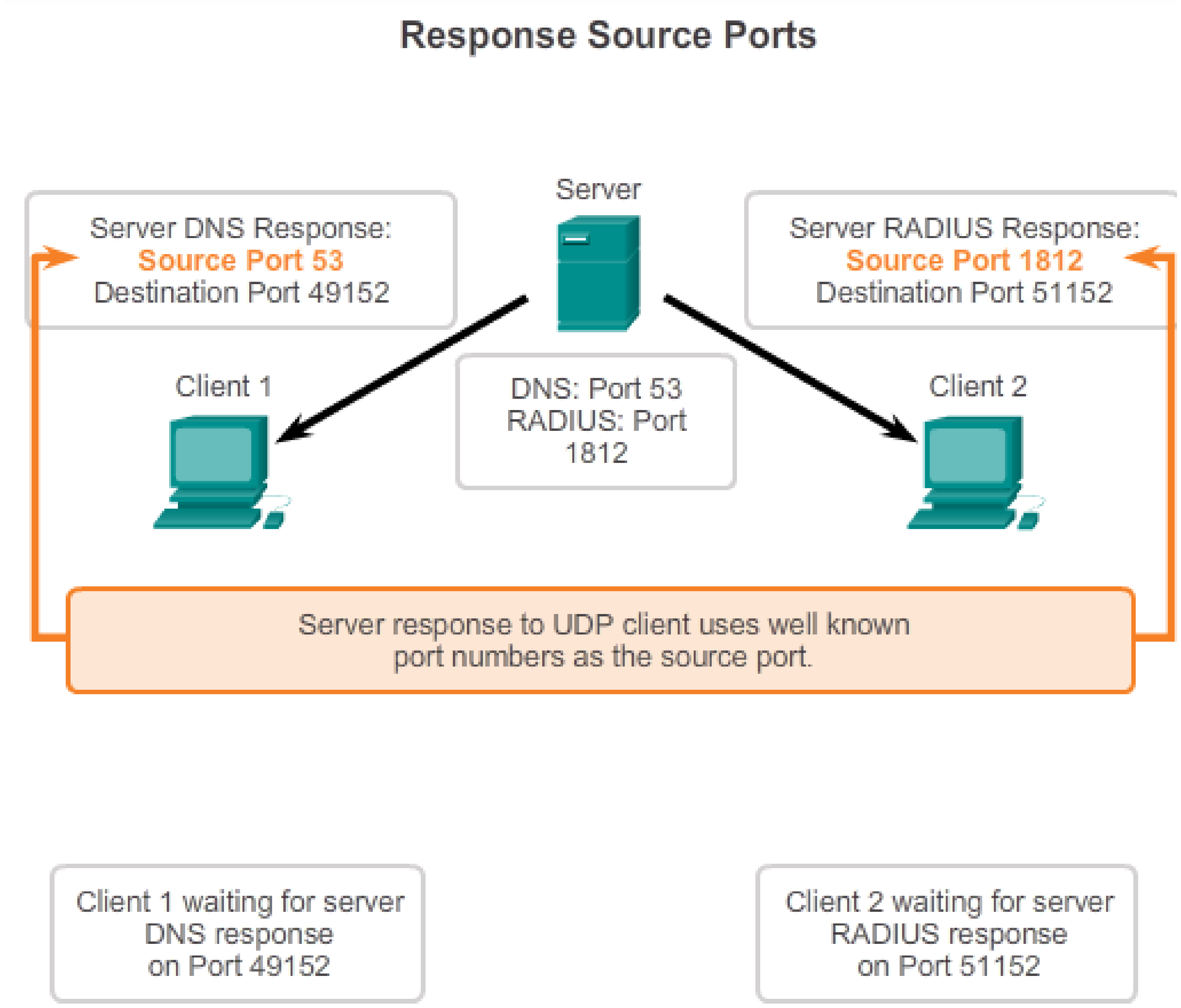


Server Applications
Client DNS requests will be received on Port 53.

Client RADIUS requests will be received on Port 1812.

Client requests to servers have well known port numbers as the destination port.

7.2.3.4 UDP Client Processes



Response Source Ports

Server DNS Response:
Source Port 53
Destination Port 49152

Server RADIUS Response:
Source Port 1812
Destination Port 51152

DNS: Port 53
RADIUS: Port 1812

Server response to UDP client uses well known port numbers as the source port.

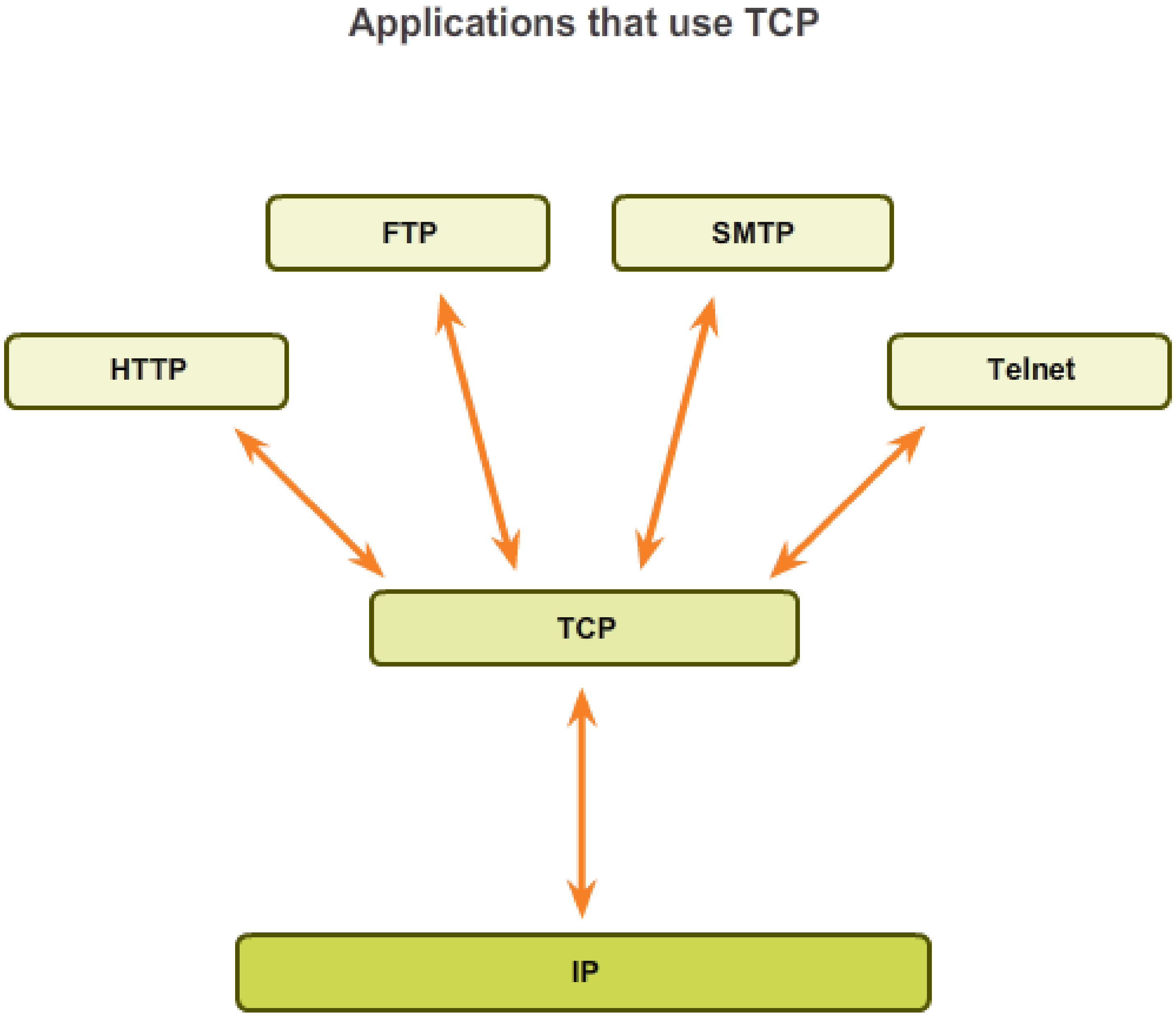
Client 1 waiting for server DNS response on Port 49152

Client 2 waiting for server RADIUS response on Port 51152

7.2.3.5 Lab - Using Wireshark to Examine a UDP DNS Capture

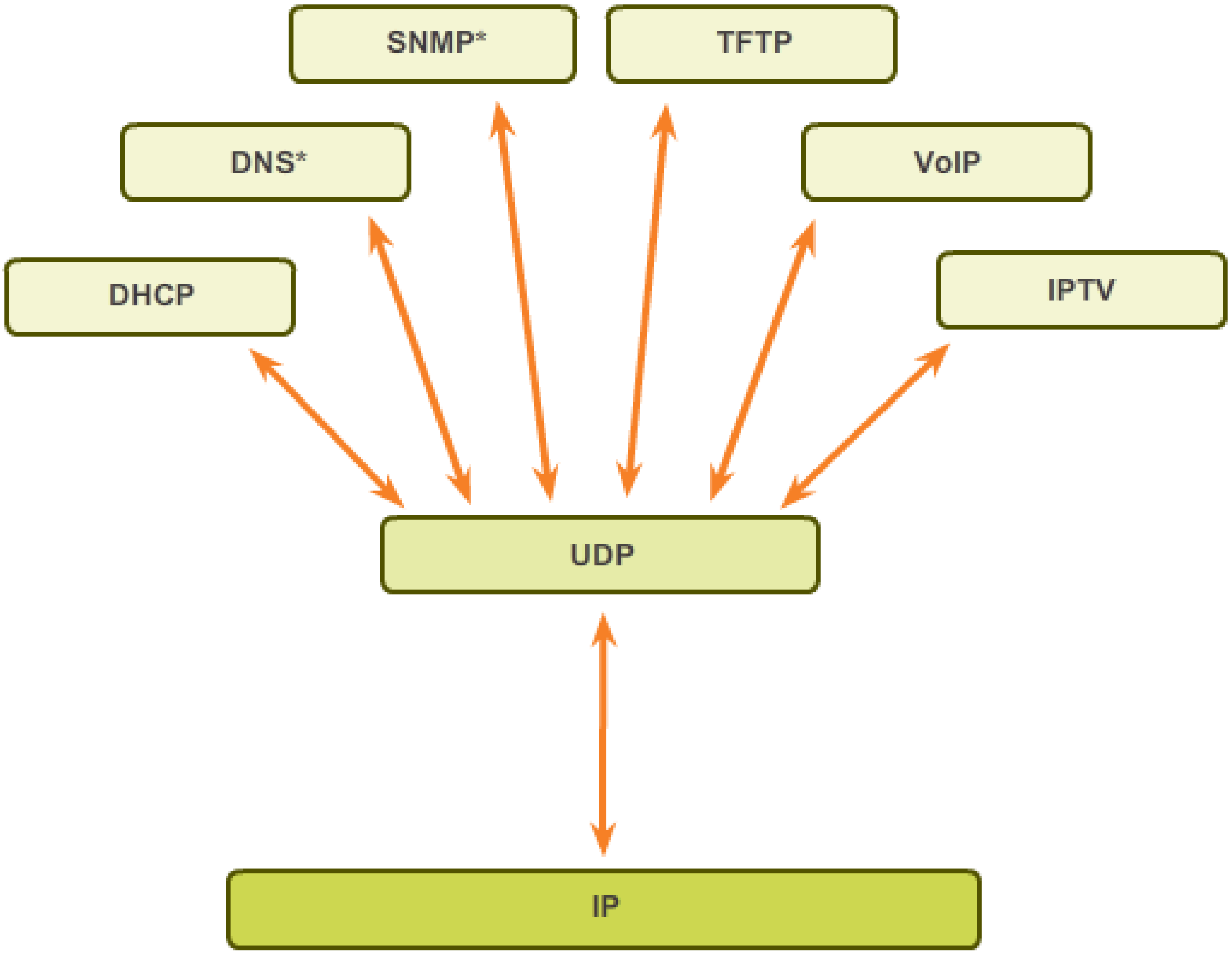
Using Wireshark to Examine a UDP DNS Capture

7.2.4.1 Applications that use TCP



7.2.4.2 Applications that use UDP

Applications that use UDP



SNMP*

TFTP

DNS*

VoIP

DHCP

IPTV

UDP

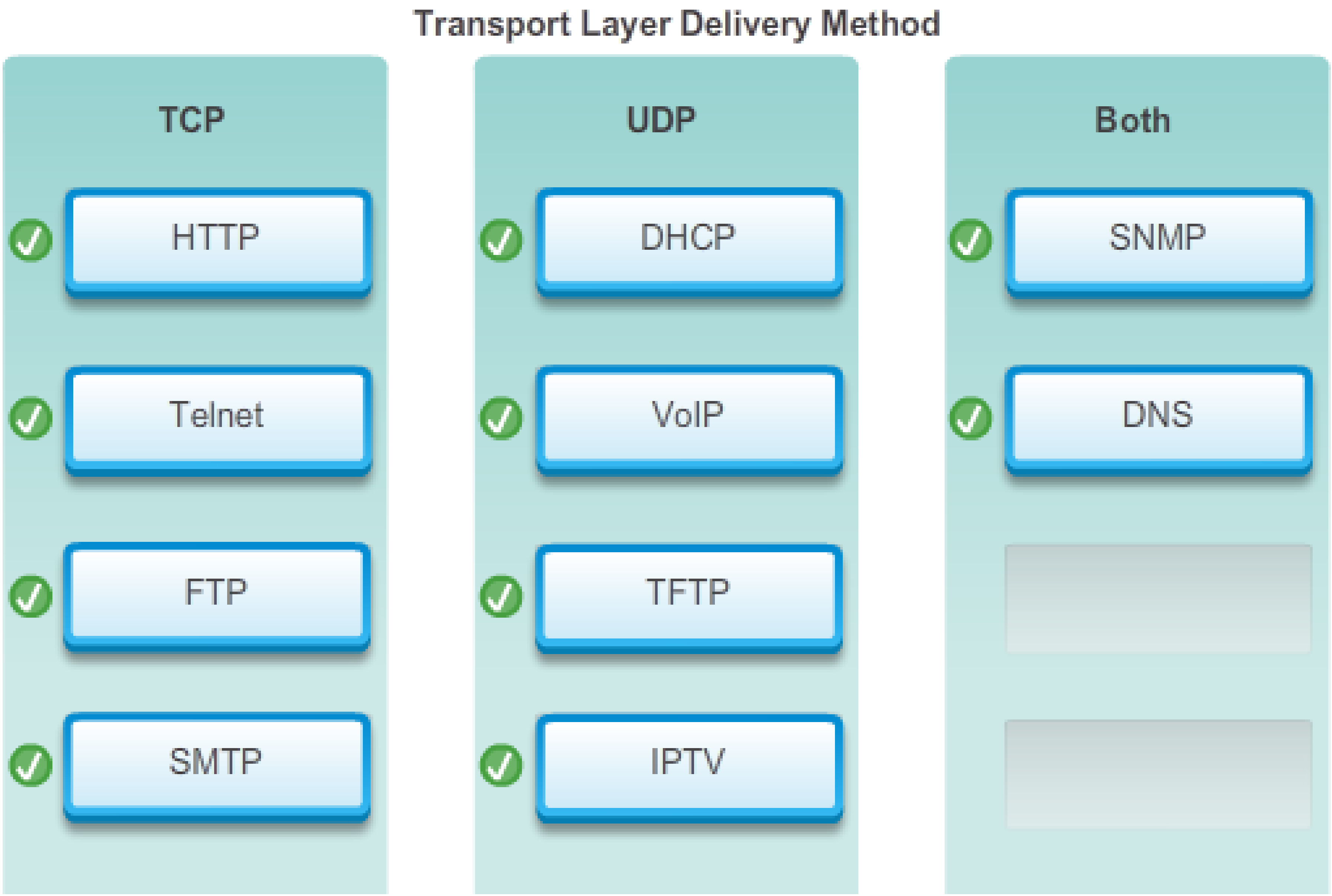
IP

*These applications can also use TCP.

7.2.4.3 Lab - Using Wireshark to Examine FTP and TFTP Captures

Using Wireshark to Examine FTP and TFTP Captures

7.2.4.4 Activity - TCP, UDP or Both



Transport Layer Delivery Method

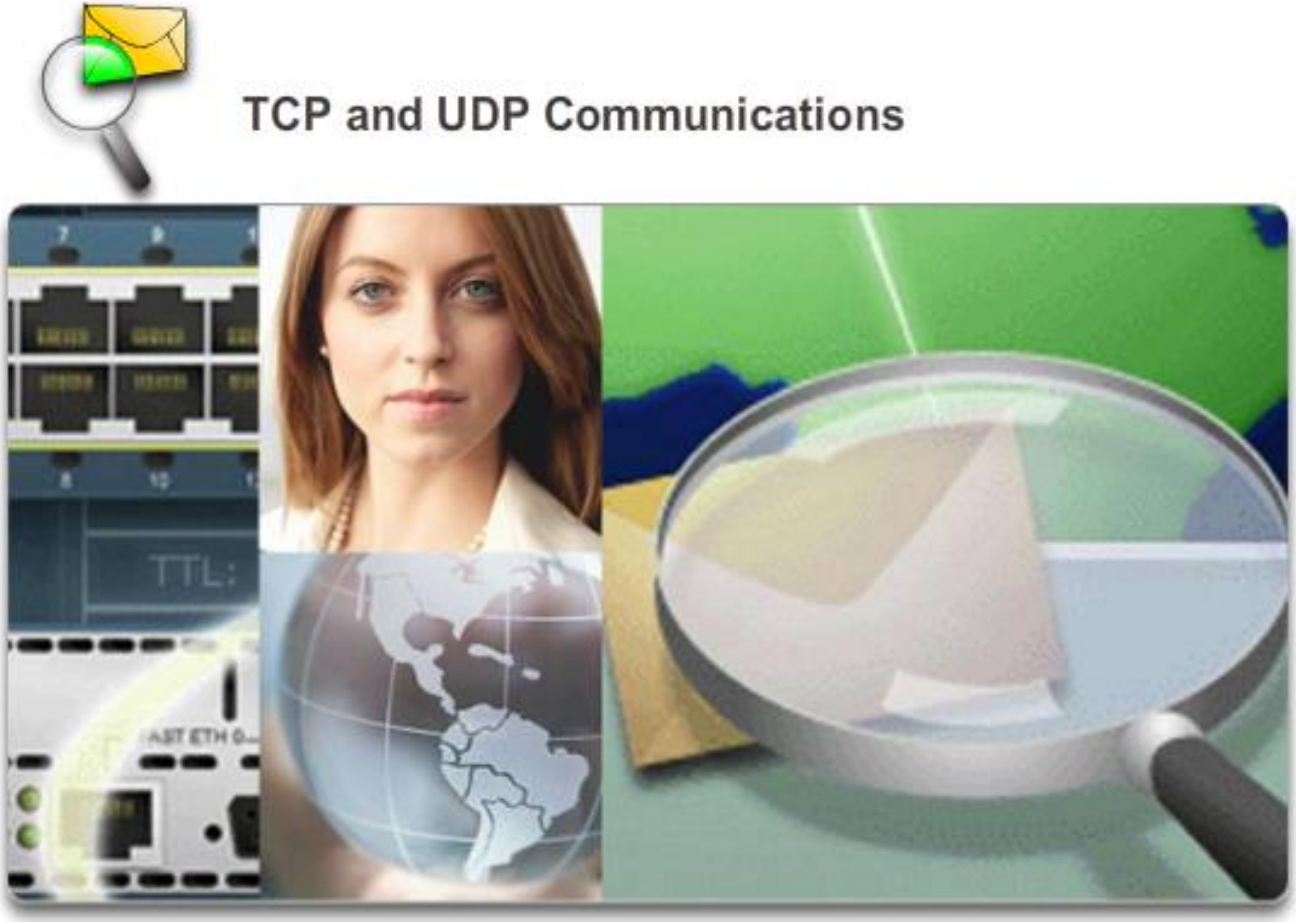
TCP	UDP	Both
✓ HTTP	✓ DHCP	✓ SNMP
✓ Telnet	✓ VoIP	✓ DNS
✓ FTP	✓ TFTP	
✓ SMTP	✓ IPTV	

7.3.1.1 Class Activity - We Need to Talk, Again – Game

TCP and UDP are transport layer protocols instrumental in ensuring that…

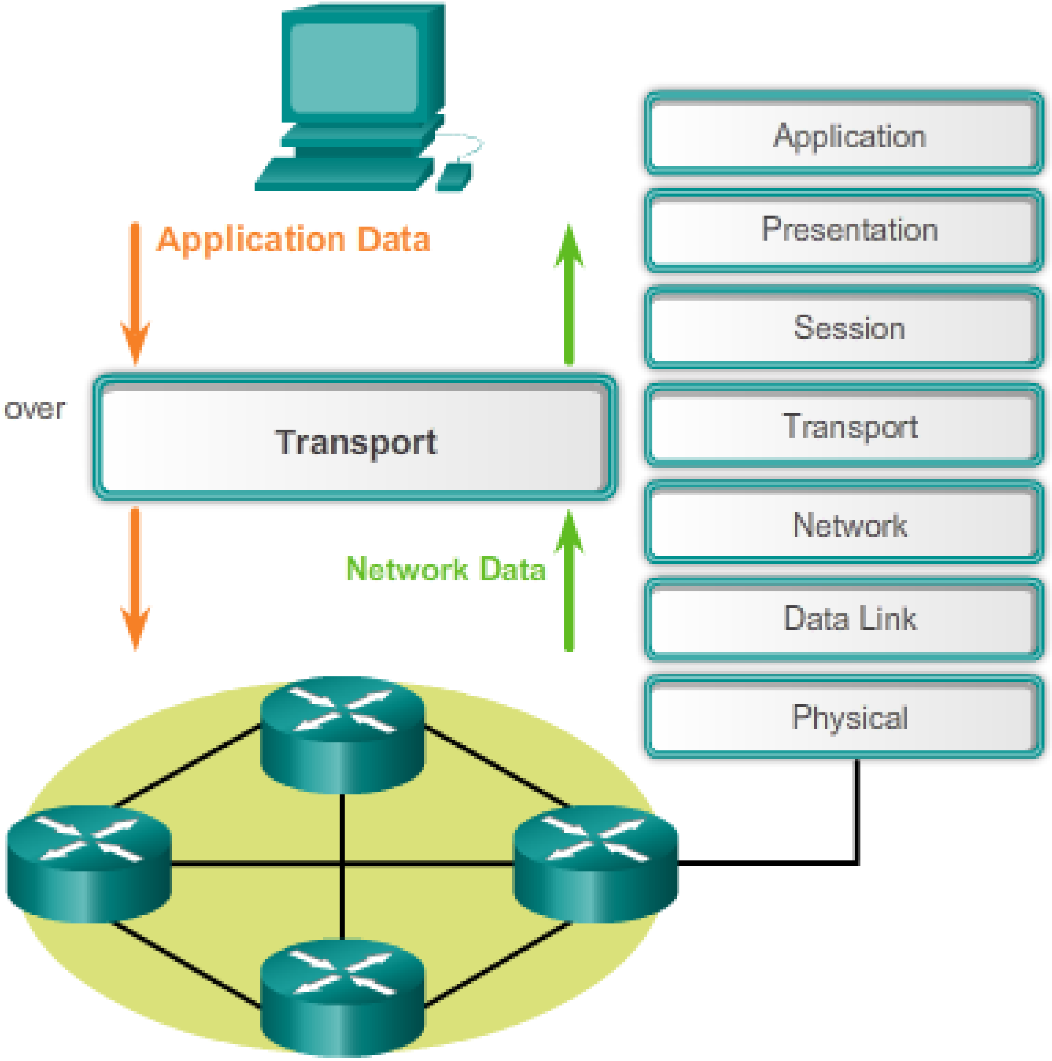
- *Network communications with different levels of importance are sent/received according to their levels of importance.*
- *The type of data will affect whether TCP or UDP will be used as the method of delivery.*
- *Timing is a factor and will affect how long it takes to send/receive TCP/UDP data transmissions.*

7.3.1.2 Packet Tracer Simulation - TCP and UDP Communications



7.3.1.3 Summary

The OSI Transport Layer



The transport layer prepares application data for transport over the network and processes network data for use by applications.

Thanks

Thanks for your attention!!

CISCO SYSTEMS®