Module 14: Transport Layer (Taşıma Katmanı)

CCNA1

Introduction to Networks v7.0 (ITN)

Gökhan AKIN - CCIE
gokhan@agyoneticileri.org

Ozan BÜK - CCIE
ozan@agyoneticileri.org

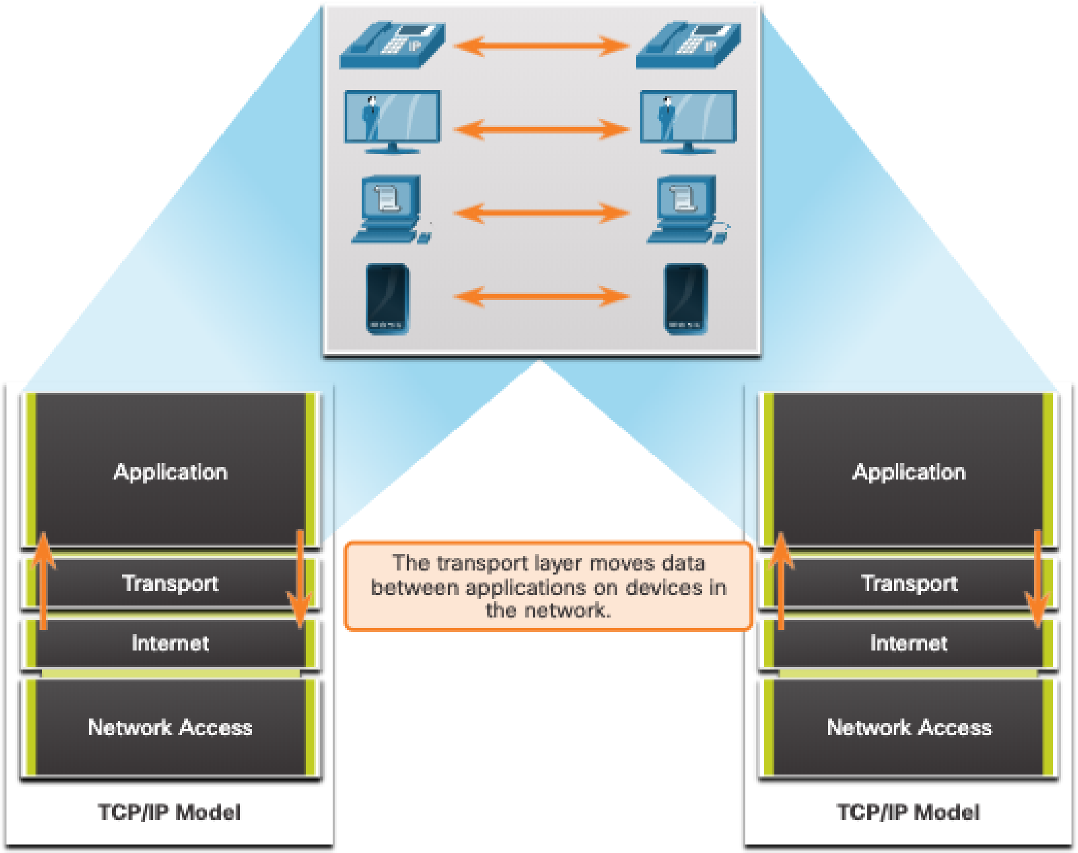
14.1 Transportation of Data (Verilerin Taşınması)

Transportation of Data

Role of the Transport Layer

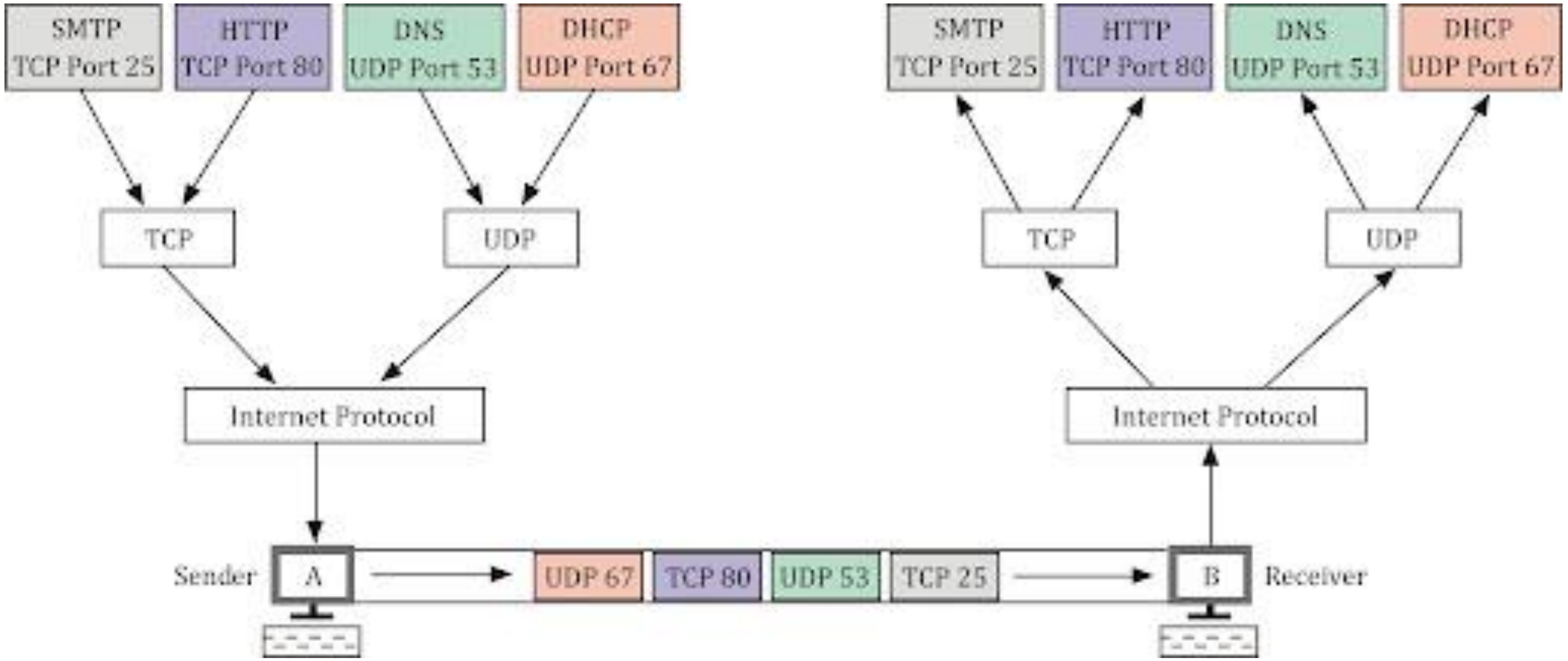
The transport layer is:

- responsible for logical communications between applications running on different hosts.
- The link between the application layer and the lower layers that are responsible for network transmission.



Transportation of Data

Role of the Transport Layer

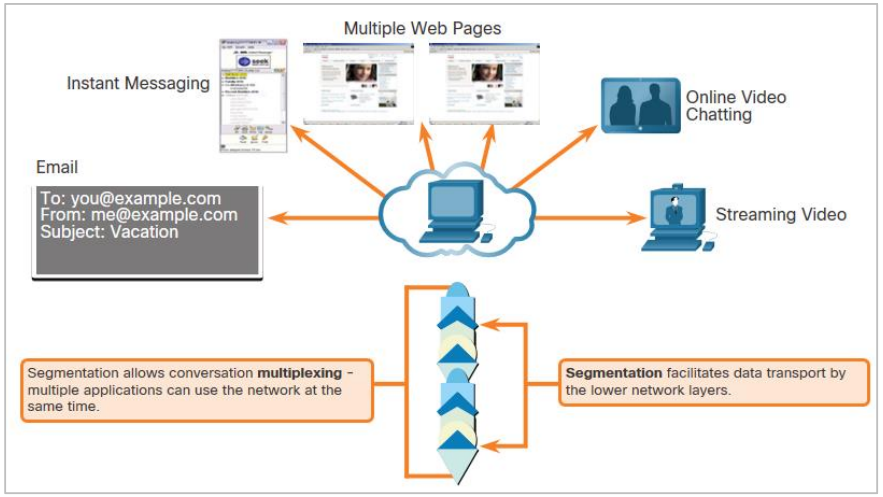


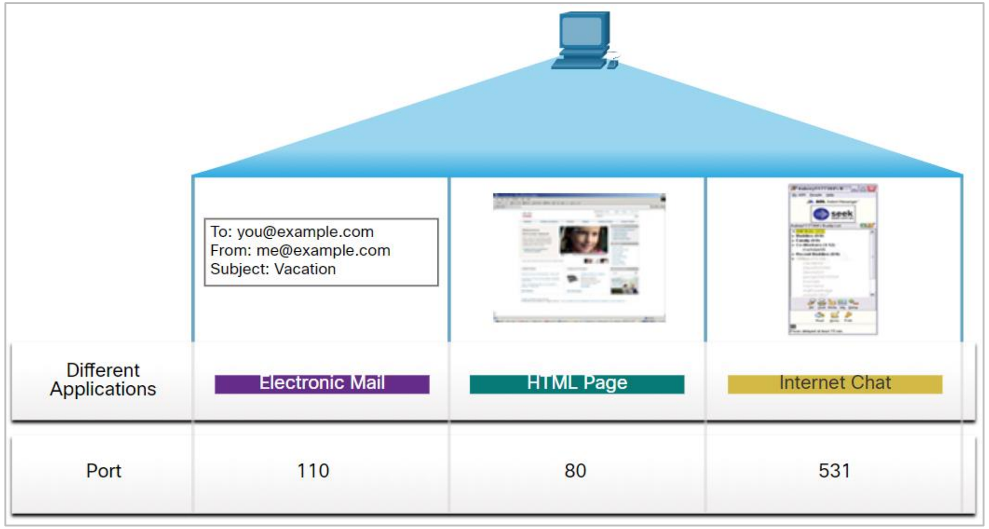
The Transport Layer

Transport Layer Responsibilities

The transport layer has many responsibilities.

- **Tracking Individual Conversations**
- **Segmenting Data and Reassembling Segments**
- **Add Header Information**
- **Identifying the Applications**
- **Conversation Multiplexing**


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.

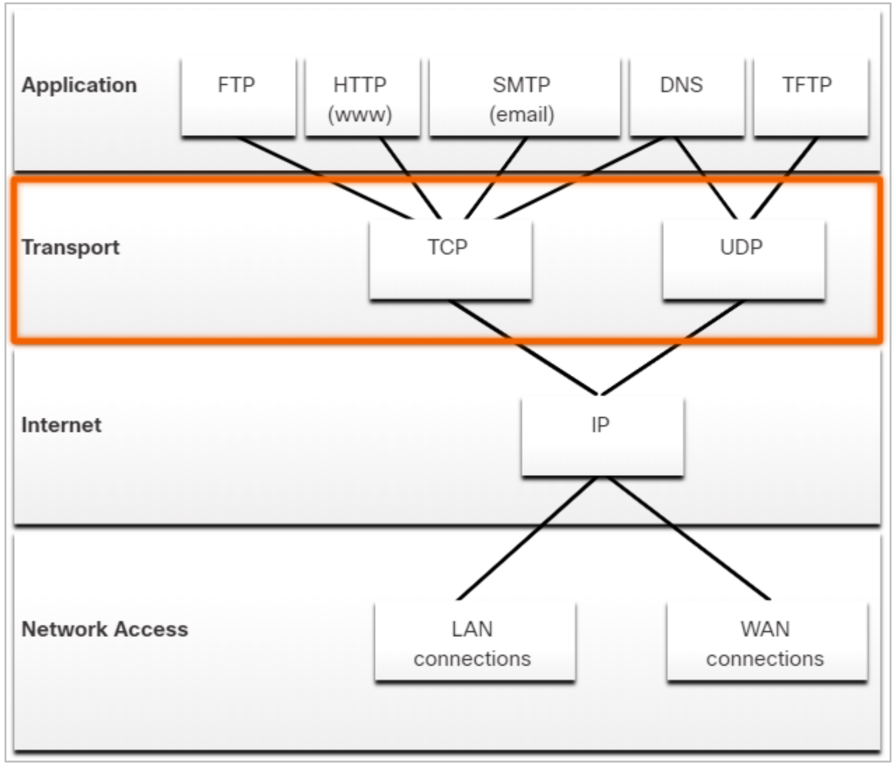


Different Applications	Electronic Mail	HTML Page	Internet Chat
Port	110	80	531

The Transport Layer

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.

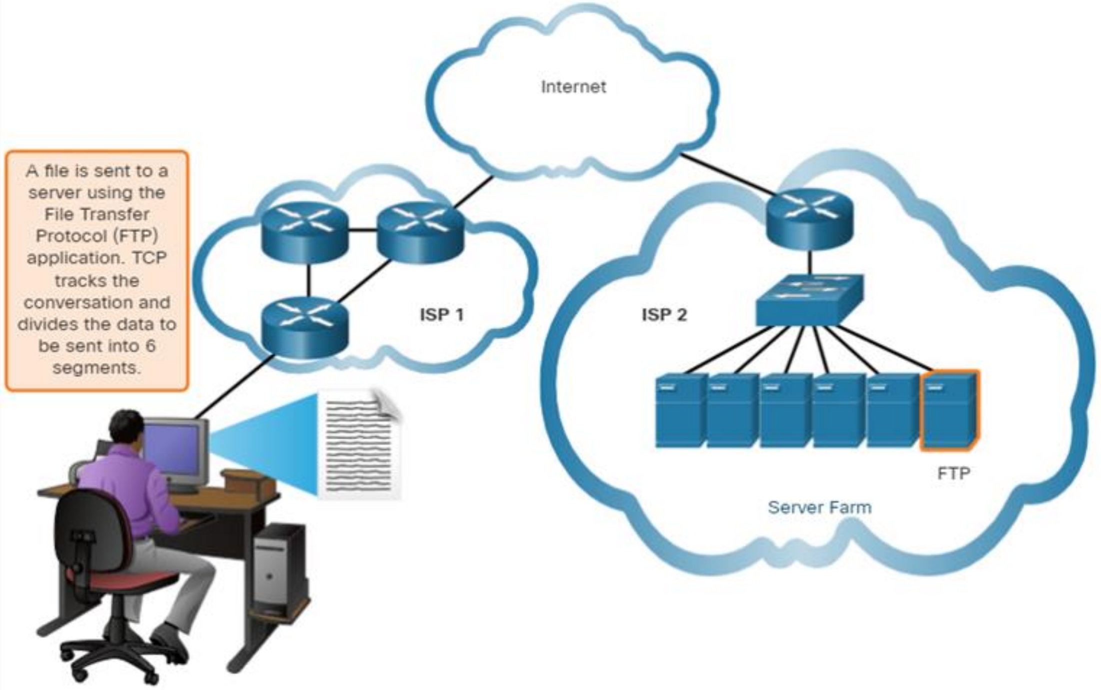


Transportation of Data

Transmission Control Protocol

TCP provides **reliability** and **flow control**. TCP basic operations:

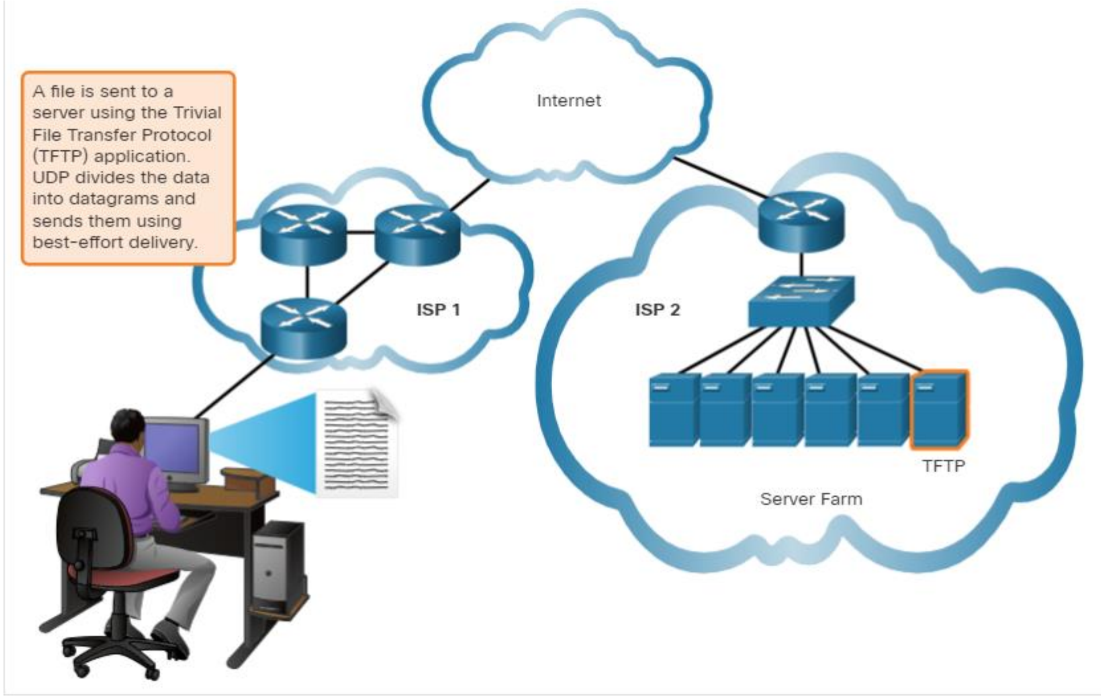
- Number and track data segments transmitted to a specific host from a specific application
- Acknowledge received data
- Retransmit any unacknowledged data after a certain amount of time
- Sequence data that might arrive in wrong order
- Send data at an efficient rate that is acceptable by the receiver


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

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

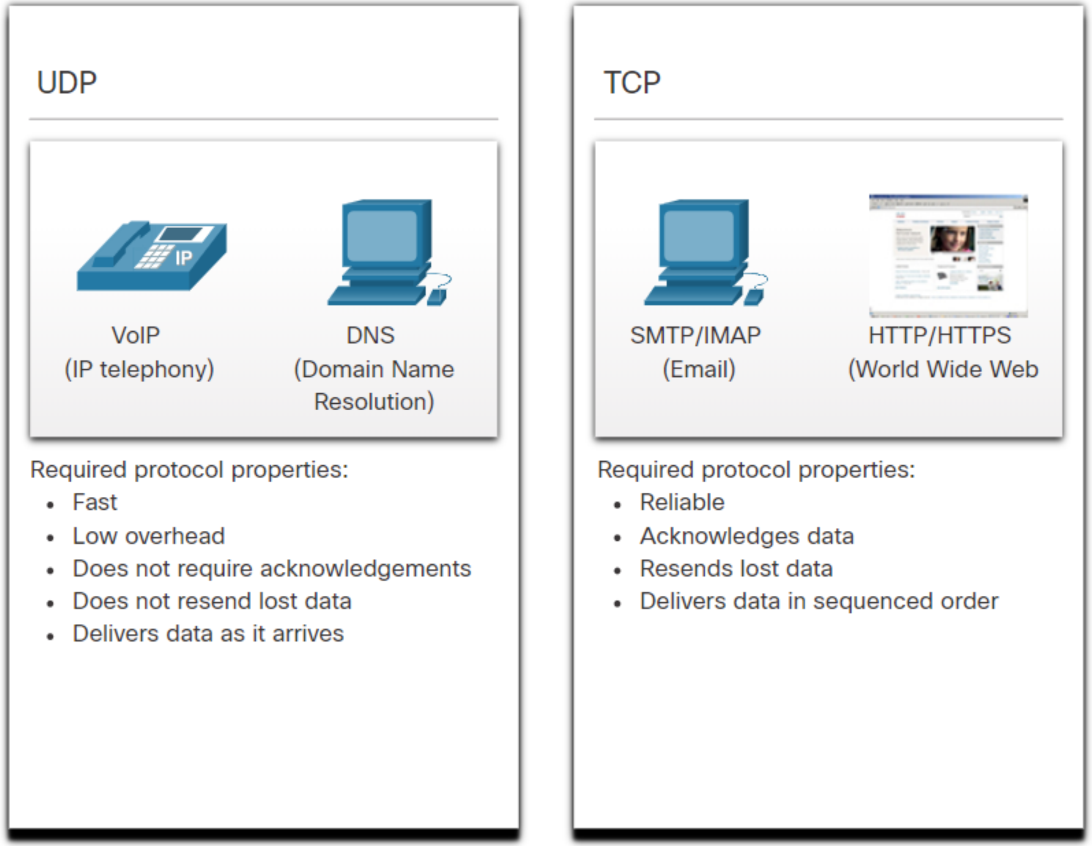

A file is sent to a server using the Trivial File Transfer Protocol (TFTP) application. UDP divides the data into datagrams and sends them using best-effort delivery.
Internet
ISP 1
ISP 2
TFTP
Server Farm

Transportation of Data

The Right Transport Layer Protocol for the Right Application

UDP is also used by **request-and-reply** applications where the data is minimal, and retransmission can be done quickly.

If it is important that all the data arrives and that it can be processed in its proper sequence, TCP is used as the transport protocol.



UDP

VoIP
(IP telephony)

DNS
(Domain Name Resolution)

Required protocol properties:

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

TCP

SMTP/IMAP
(Email)

HTTP/HTTPS
(World Wide Web

Required protocol properties:

- Reliable
- Acknowledges data
- Resends lost data
- Delivers data in sequenced order

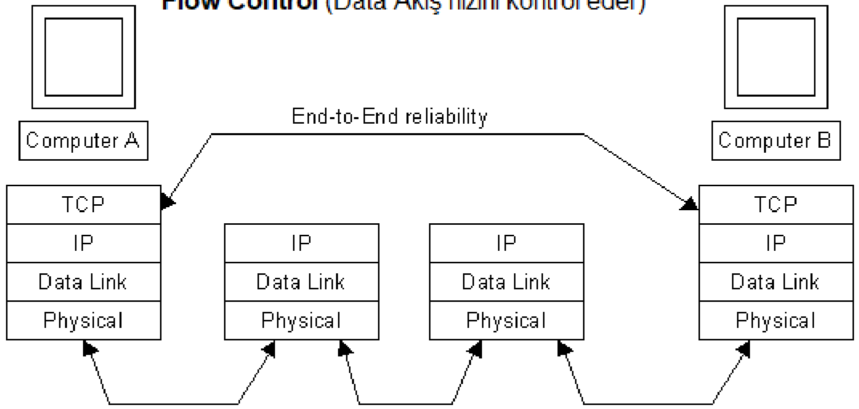
14.2 TCP’ye Genel Bakış

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.

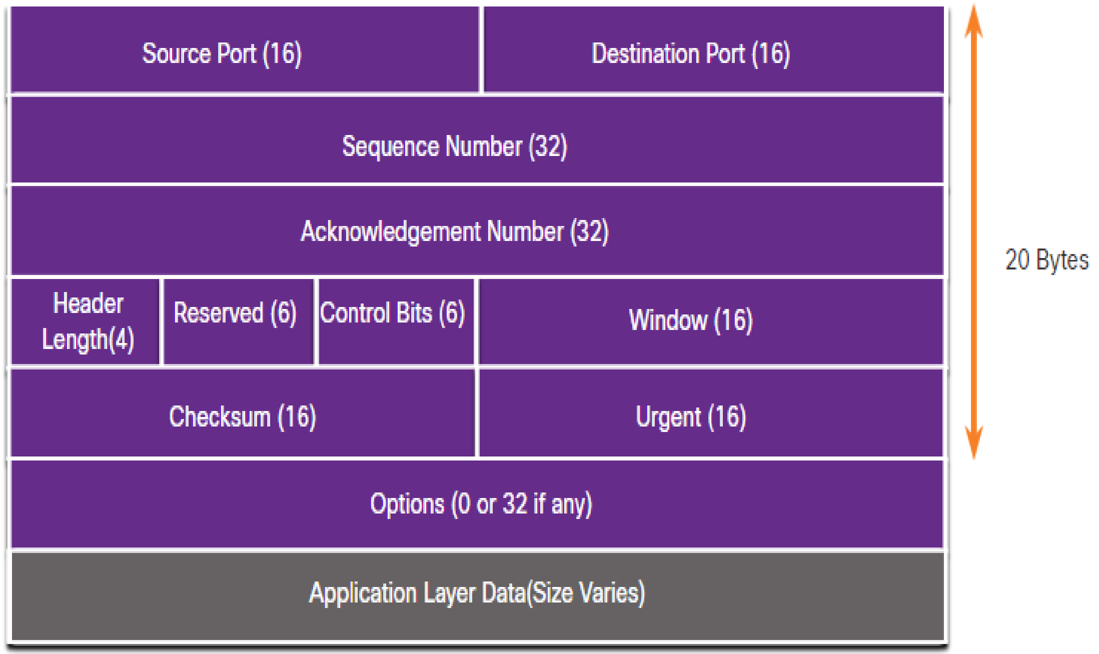
Connection Oriented (data iletmeden önce bağlantı kurulur)
Segmentation (veriyi segmentlere böler ve yeniden birleştirir)
End-to-end reliability (güvenli iletimi sağlar) seq.numbers, ack.numbers
eksik ve bozuk datayı yeniden iletir
Flow Control (Data Akış hızını kontrol eder)



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.



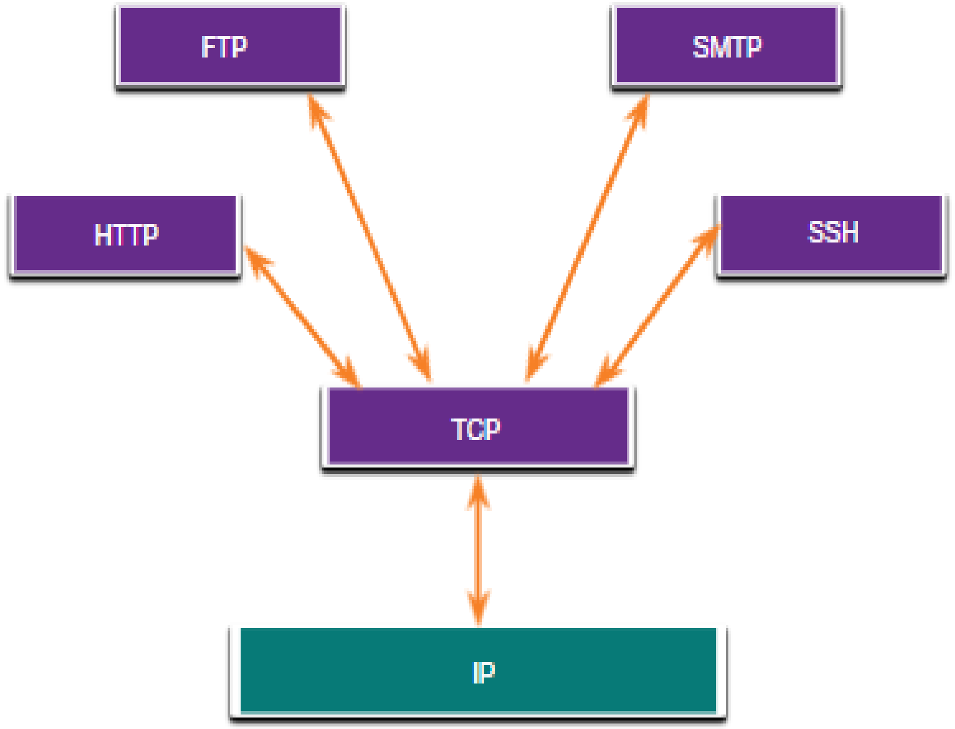
TCP Overview
TCP Header Fields

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.

TCP Overview

Applications that use TCP

TCP handles all tasks associated with dividing the data stream into segments, providing reliability, controlling data flow, and reordering segments.



14.3 UDP’ye Genel Bakış

UDP Features

UDP features include the following:

- Data is reconstructed in the order that it is received.
- Any segments that are lost are not resent.
- There is no session establishment.
- The sending is not informed about resource availability.

User Datagram Protocol (UDP)

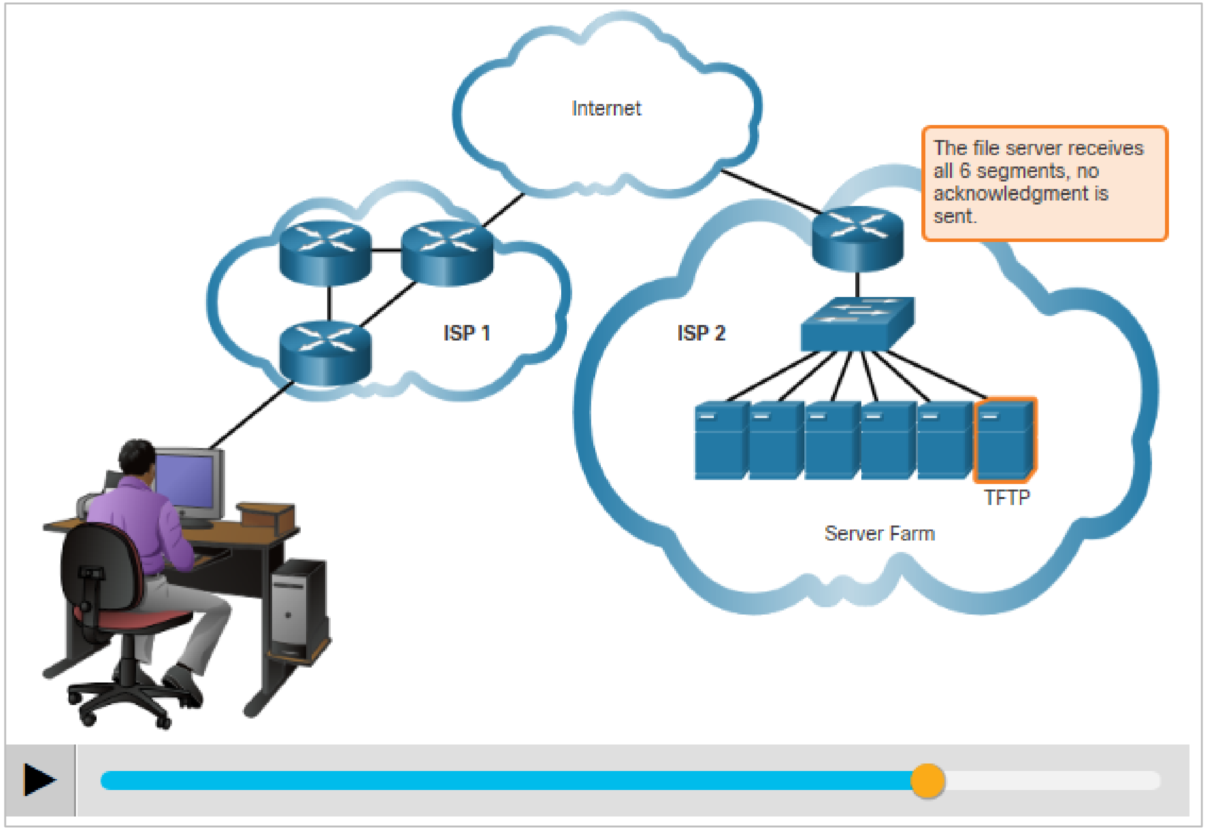
- UDP is a simpler transport layer protocol than TCP.
- It does not provide reliability and flow control, which means it requires fewer header fields.
- The sender and the receiver UDP processes do not have to manage reliability and flow control, this means UDP datagrams can be processed faster than 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.*

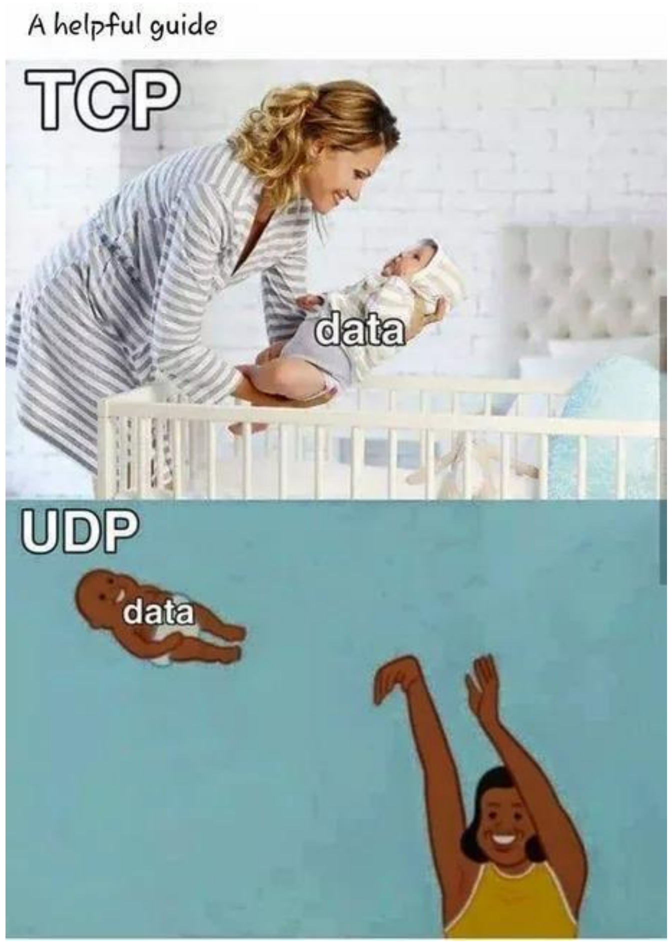
The Transport Layer

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.

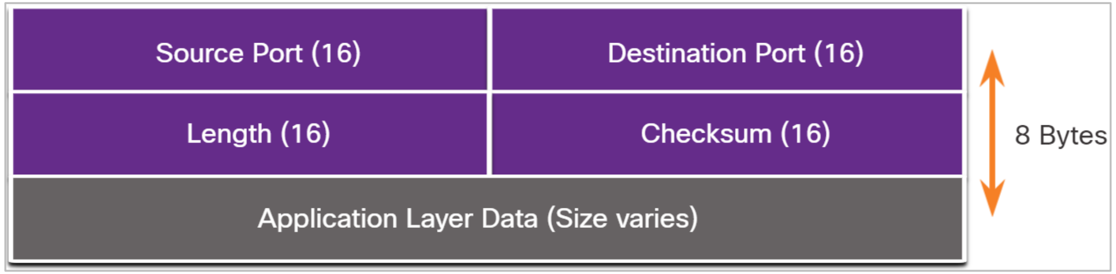

Internet
ISP 1
ISP 2
The file server receives all 6 segments, no acknowledgment is sent.
TFTP
Server Farm

TCP vs UDP



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.



UDP Header Fields

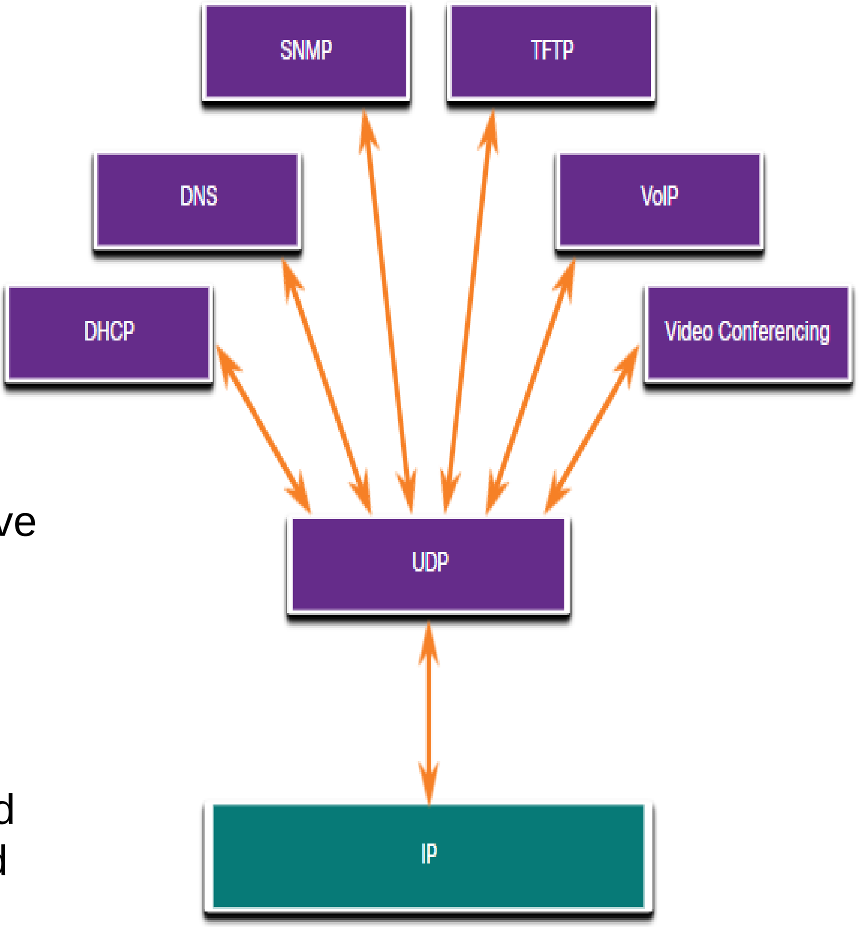
The table identifies and describes the four fields in a UDP header.

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.
Length	A 16-bit field that indicates the length of the UDP datagram header and data.
Checksum	A 16-bit field used for error checking of the datagram header and data.

UDP Overview

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.
- 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.



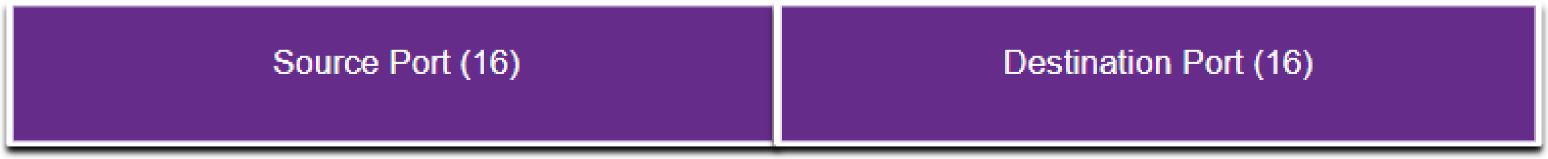
Sonuç olarak; Real time protokoller (ses ve video gibi) ve tek soru tek cevap şeklinde çalışan protokoller DNS gibi UDP protokolünü kullanır.

14.4 Port Numaraları

Port Numbers
Multiple Separate Communications

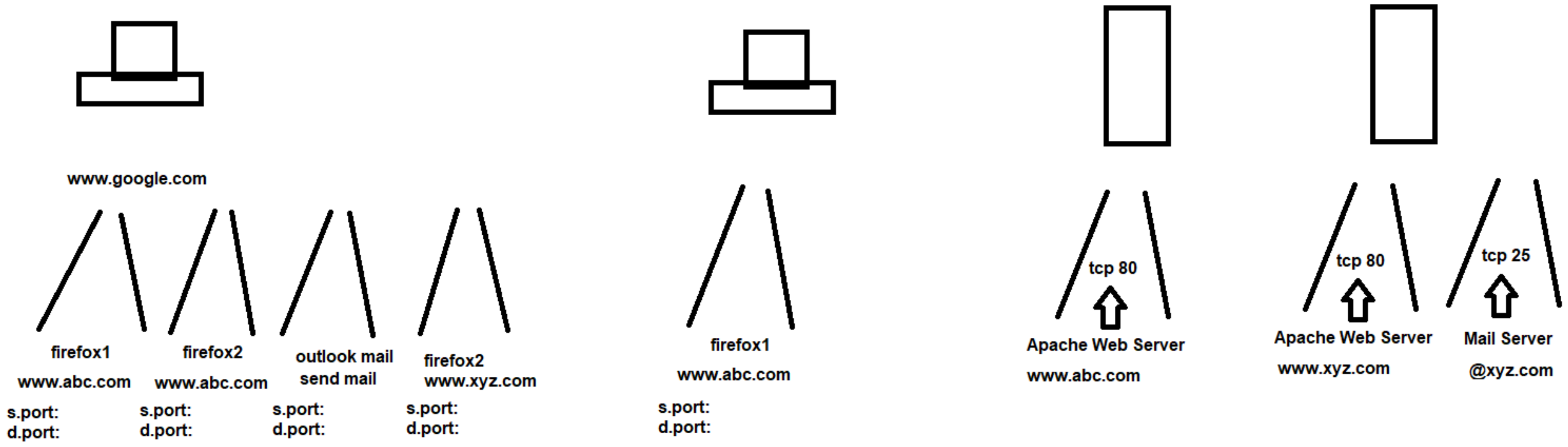
TCP and UDP transport layer protocols use port numbers to manage multiple, simultaneous conversations.

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.



Port Numbers

Multiple Separate Communications

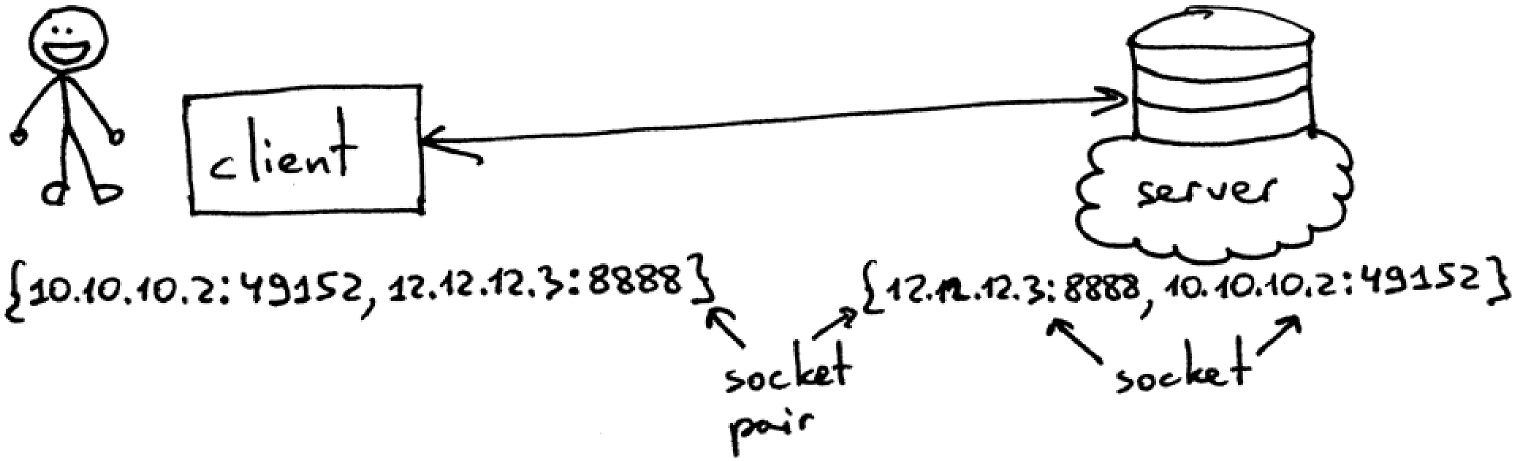


Port Numbers
Port Number Groups

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.

Socket Pairs

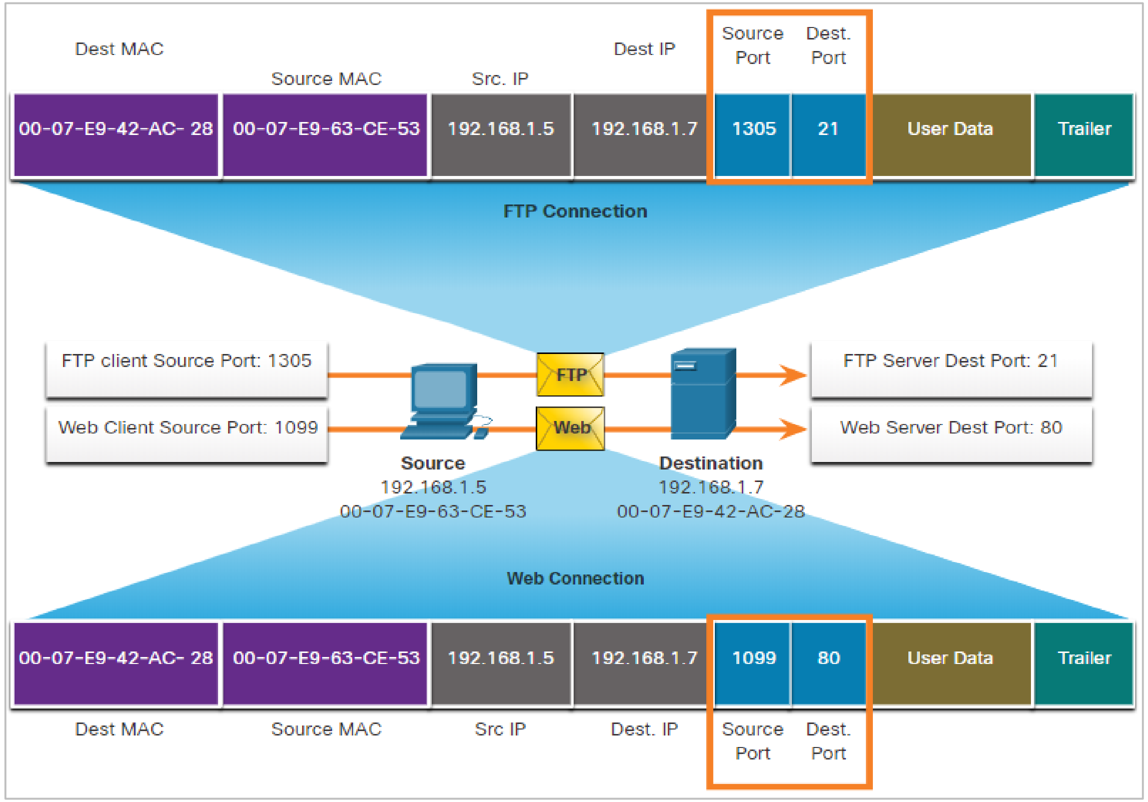
- The source and destination ports are placed within the segment. The segments are then encapsulated within an IP packet.
- The IP packet contains the IP address of the source and destination. The combination of the source IP address and source port number, or the destination IP address and destination 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.



The Transport Layer

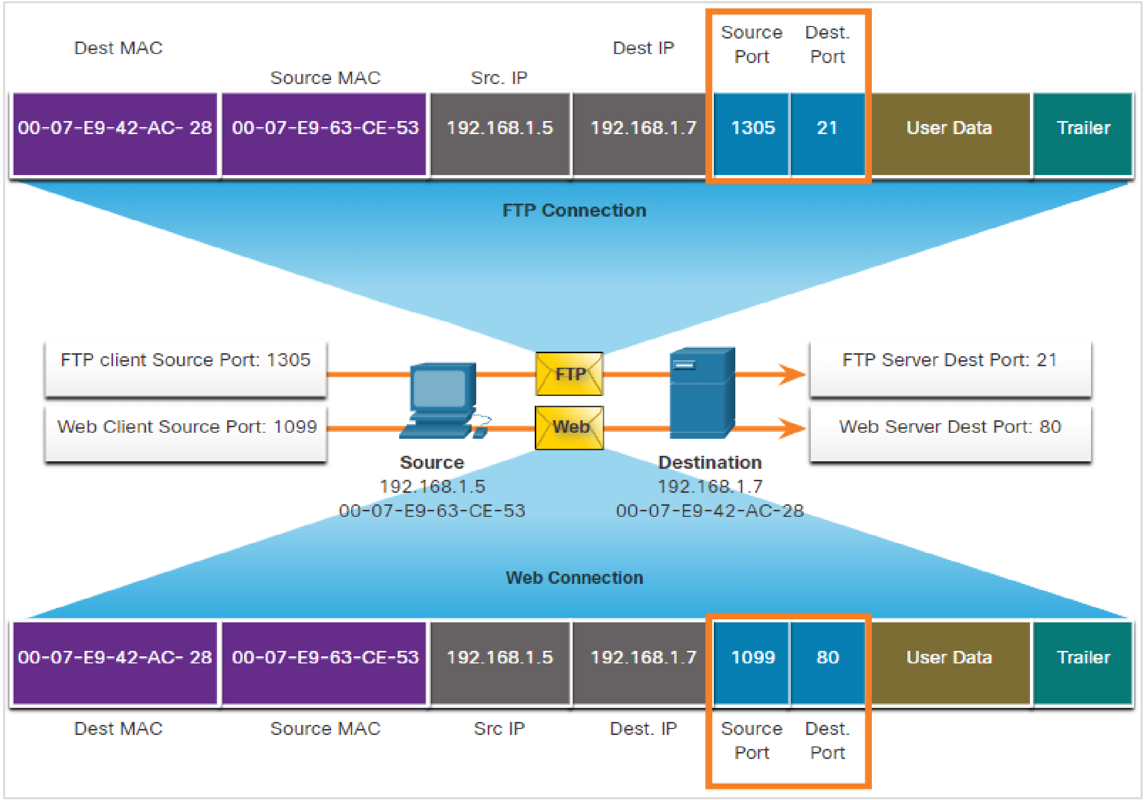
Socket Pairs (Contd.)

- 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



Port Number Groups (Cont.)

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)

Port Numbers
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
```

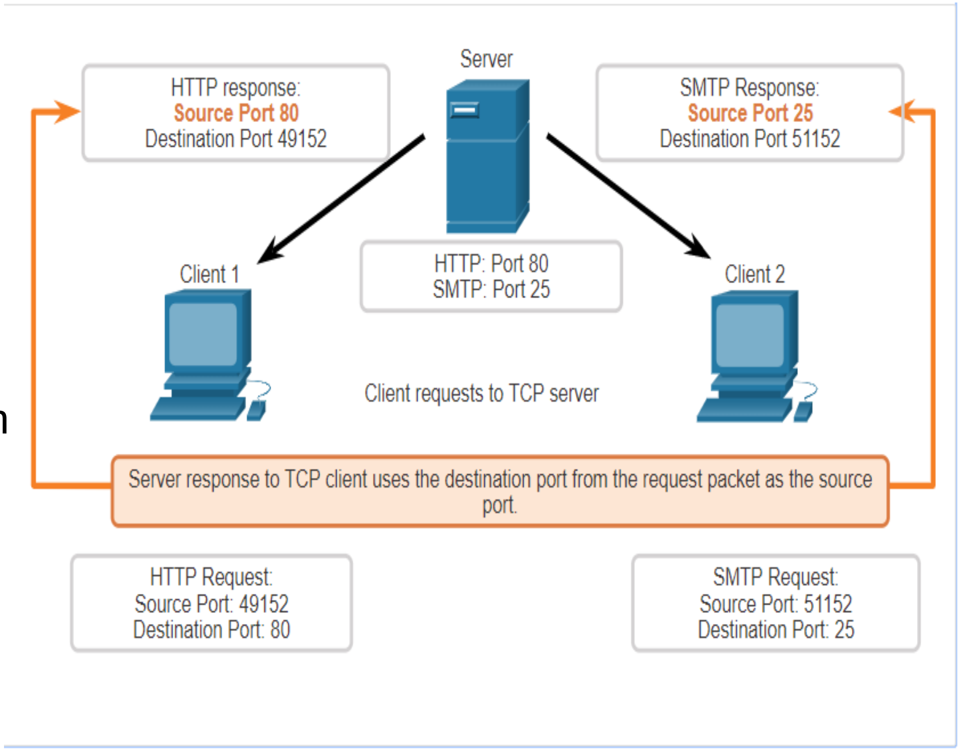
14.5 TCP İletişim Süreci

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.


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

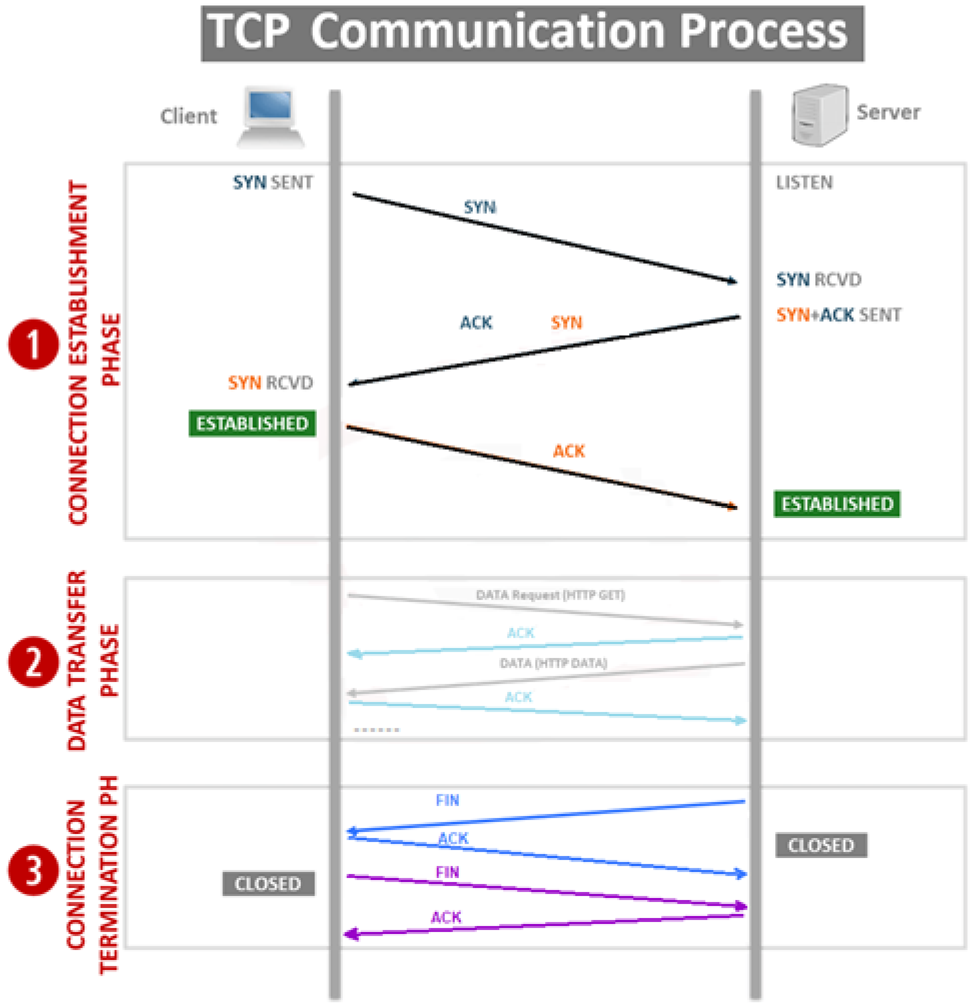
Transport Layer Session Establishment

TCP Connection

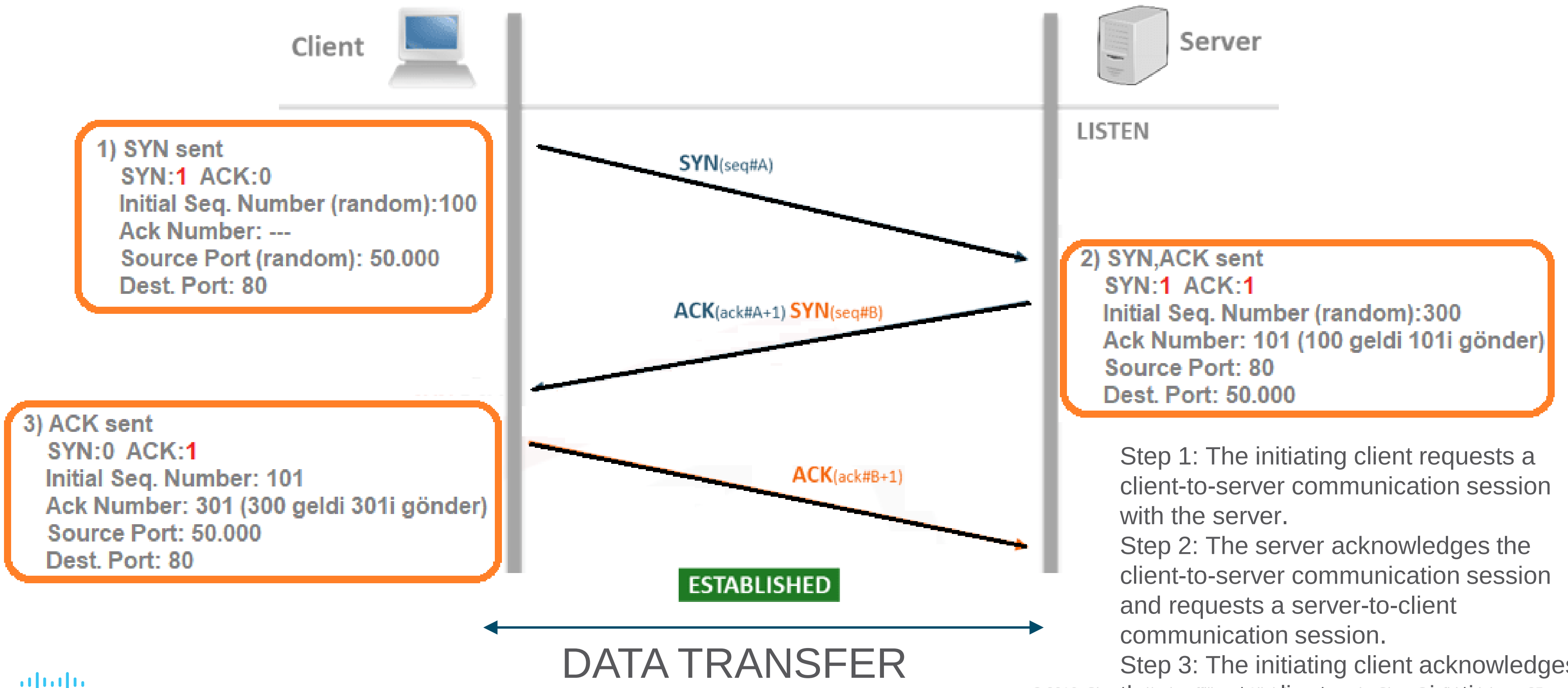
Connection Establishment Phase:
(three-way handshake)

Data Transfer Phase:

Connection Termination Phase:



TCP Connection Establishment (Three Way Handshake)



1) SYN sent
SYN:1 ACK:0
Initial Seq. Number (random):100
Ack Number: ---
Source Port (random): 50.000
Dest. Port: 80

2) SYN,ACK sent
SYN:1 ACK:1
Initial Seq. Number (random):300
Ack Number: 101 (100 geldi 101i gönder)
Source Port: 80
Dest. Port: 50.000

3) ACK sent
SYN:0 ACK:1
Initial Seq. Number: 101
Ack Number: 301 (300 geldi 301i gönder)
Source Port: 50.000
Dest. Port: 80

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

Transport Layer Session Establishment

Session Termination (Contd.)

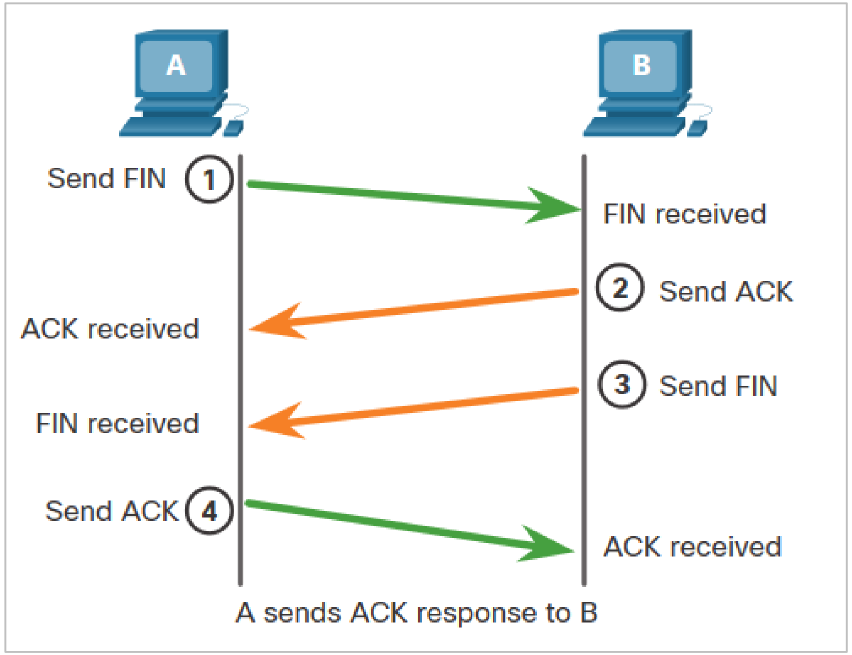
The session termination steps are:

Step 1. FIN: When the client has no more data to send in the stream, it sends a segment with the FIN flag set.

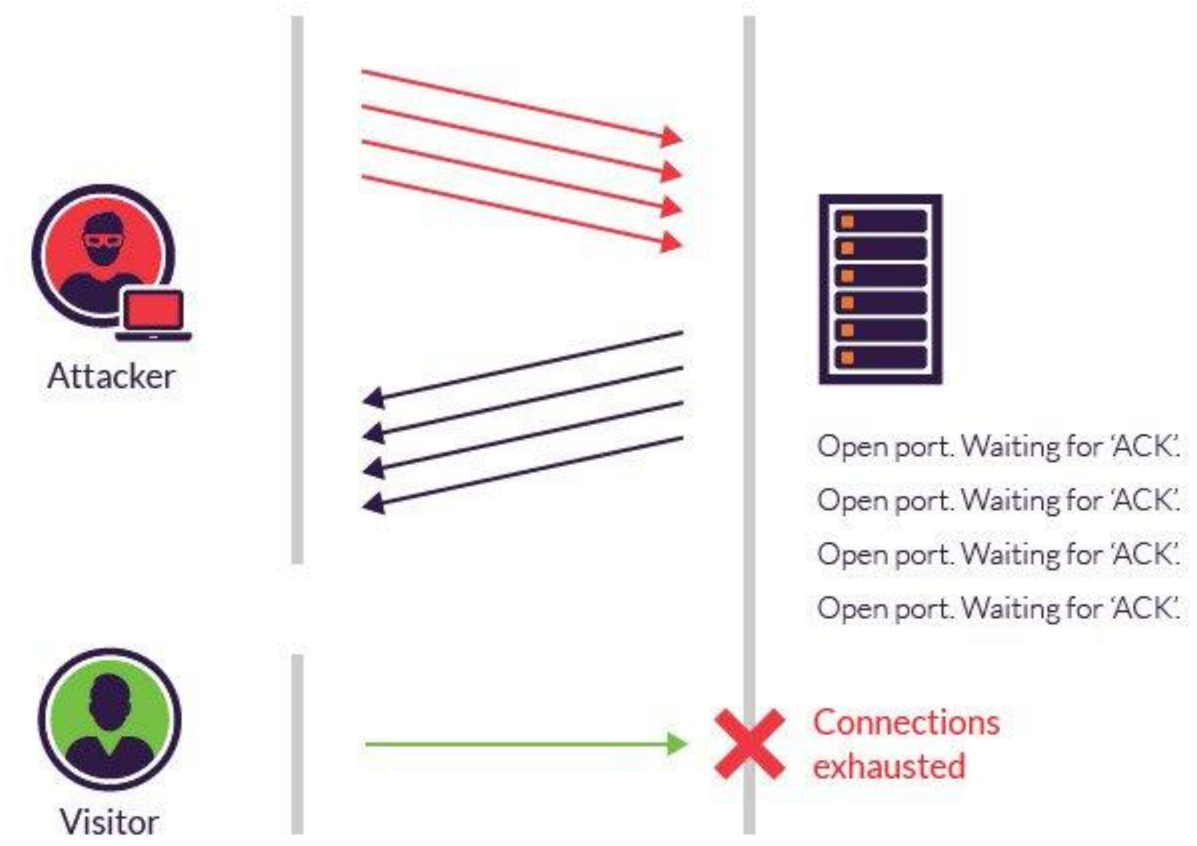
Step 2. ACK: The server sends an ACK to acknowledge the receipt of the FIN to terminate the session from client to server.

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.



SYN Attack


Attacker
Open port. Waiting for 'ACK'.
Open port. Waiting for 'ACK'.
Open port. Waiting for 'ACK'.
Open port. Waiting for 'ACK'.
Visitor
Connections exhausted

TCP Three-way Handshake Analysis

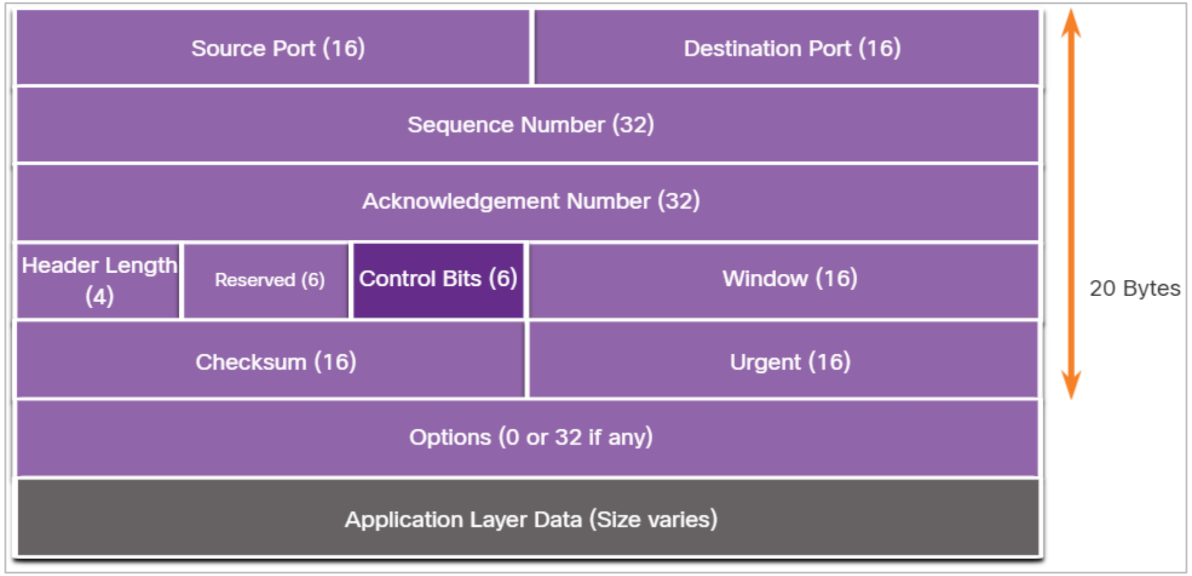
- The functions of the three-way handshake are:
  - It establishes that the destination device is present on the network.
  - It verifies that the destination device has an active service and is accepting requests on the destination port number that the initiating client intends to use.
  - It informs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

Transport Layer Session Establishment

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



14.6 Reliability and Flow Control (Güvenilirlik ve Akış Kontrolü)

Transport Layer Reliability

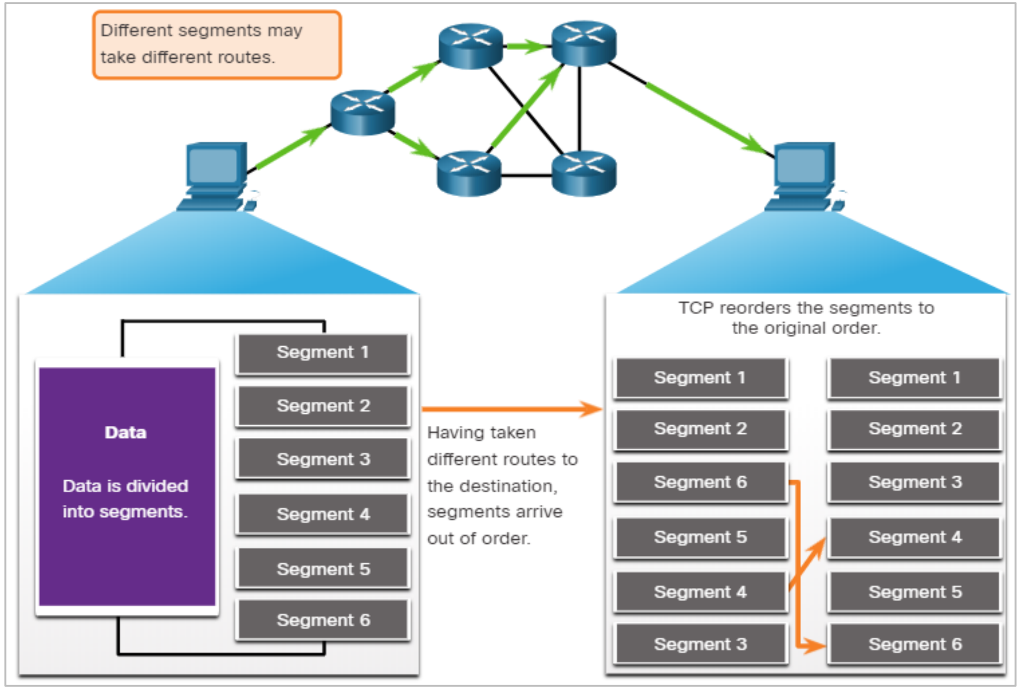
TCP Reliability - Guaranteed and Ordered Delivery

- 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.

Transport Layer Reliability

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.


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

Transport Layer Reliability

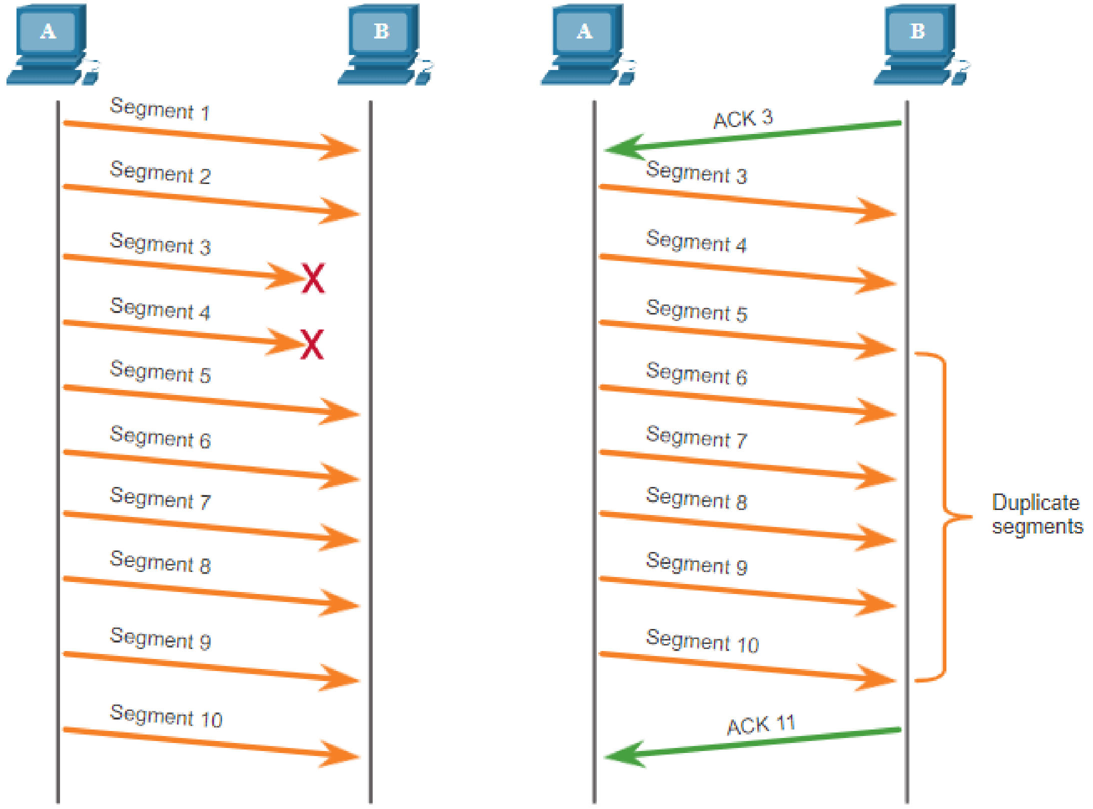
TCP Reliability - Data Loss and Retransmission

- 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.

Reliability and Flow Control

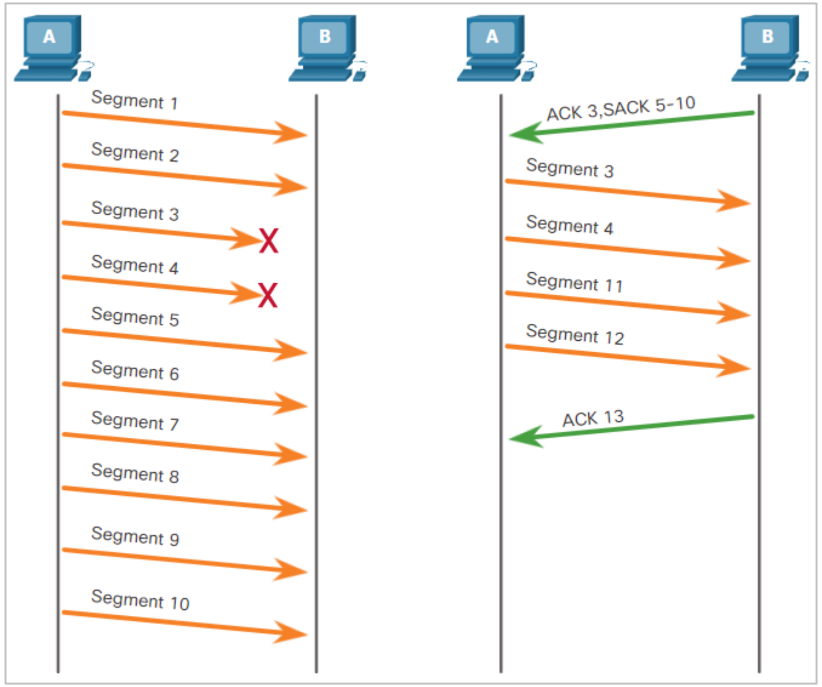
TCP Reliability – Data Loss and Retransmission

- In the figure, Host A sends segments 1 through 10 to host B. If all the segments arrive except segments 3 and 4, host B would reply with acknowledgment specifying that the next segment expected is segment 3.
- Host A has no idea if any other segments arrived or not. It would resend segments 3 through 10.
- If all the resent segments arrived successfully, segments 5 through 10 would be duplicates. This can lead to delays, congestion, and inefficiencies.



TCP Reliability - Data Loss and Retransmission (Contd.)

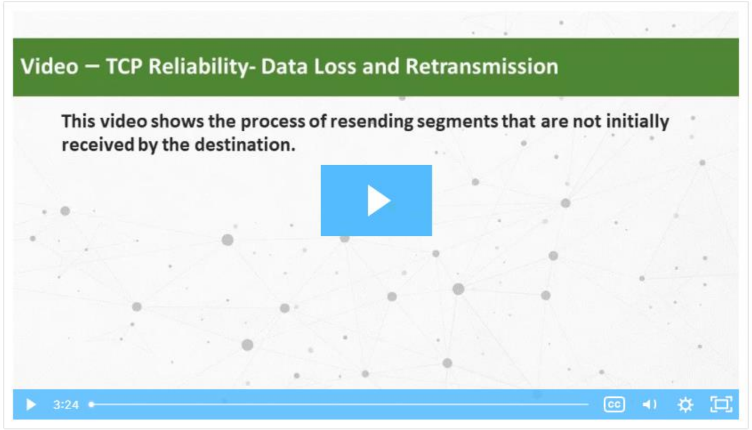
- Host operating systems employ an optional TCP feature called **selective acknowledgment (SACK),** negotiated during the three-way handshake.
- If both hosts support SACK, the receiver can acknowledge which segments (bytes) were received including any discontinuous segments.
- The sending host would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



Transport Layer Reliability

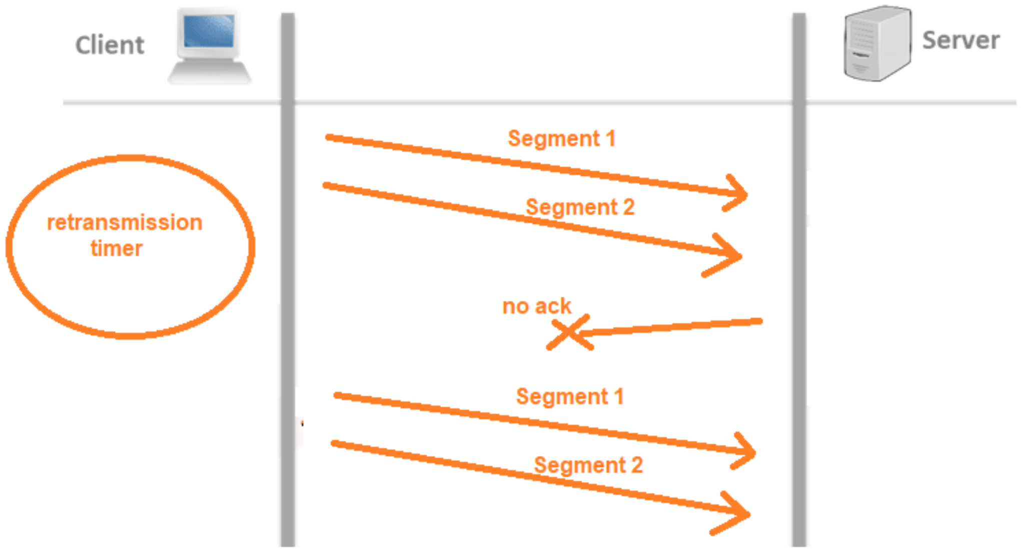
Video - TCP Reliability - Data Loss and Retransmission

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



Video – TCP Reliability- Data Loss and Retransmission

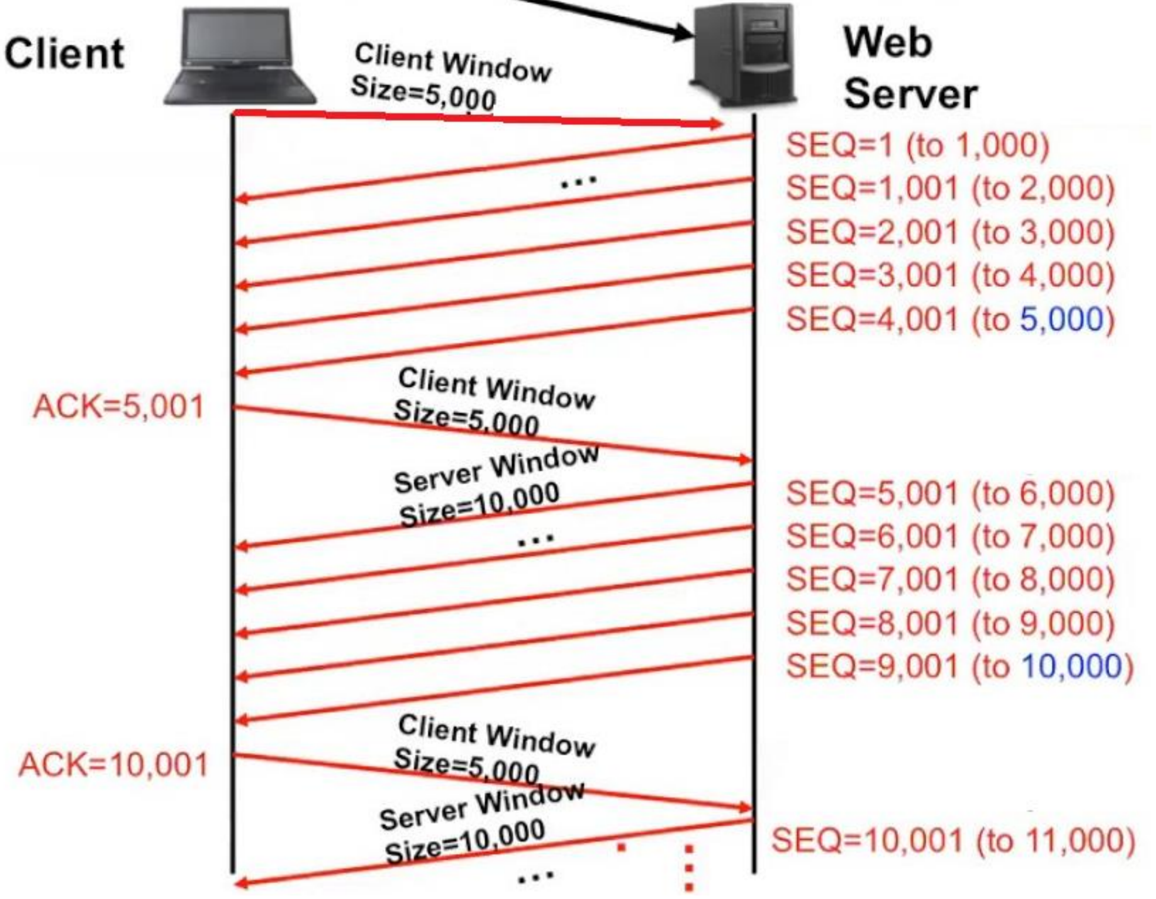
This video shows the process of resending segments that are not initially received by the destination.



Transport Layer Reliability

TCP Flow Control - Window Size and Acknowledgments

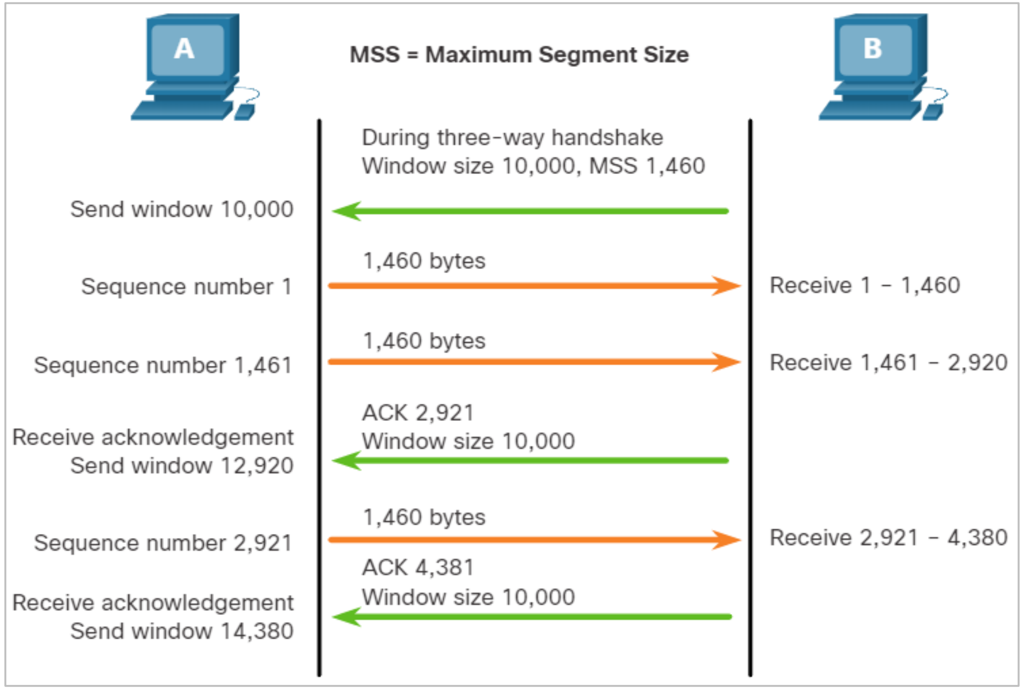
- TCP also provides mechanisms for flow control. Flow control is the amount of data that the destination can receive and process reliably.
- Flow control helps maintain the reliability of TCP transmission by adjusting the rate of data flow between source and destination for a given session.
- To accomplish 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.



Transport Layer Reliability

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.



Transport Layer Reliability

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

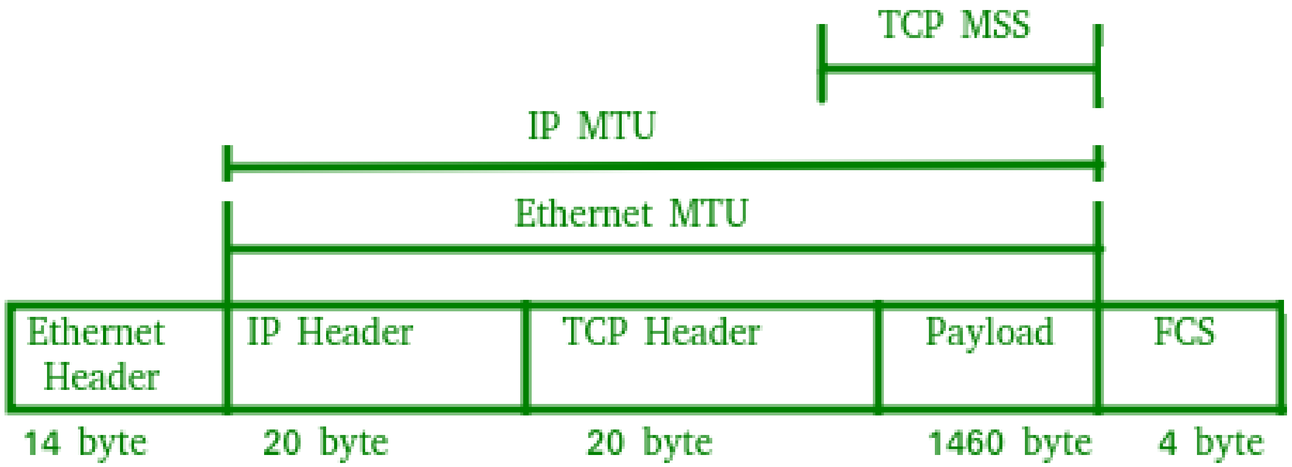
- The destination will not wait for all the bytes for its window size to be received before replying with an acknowledgment.
- As the bytes are received and processed, the destination will send acknowledgments to inform the source that it can continue to send additional bytes.
- A destination sending 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.*

Transport Layer Reliability

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

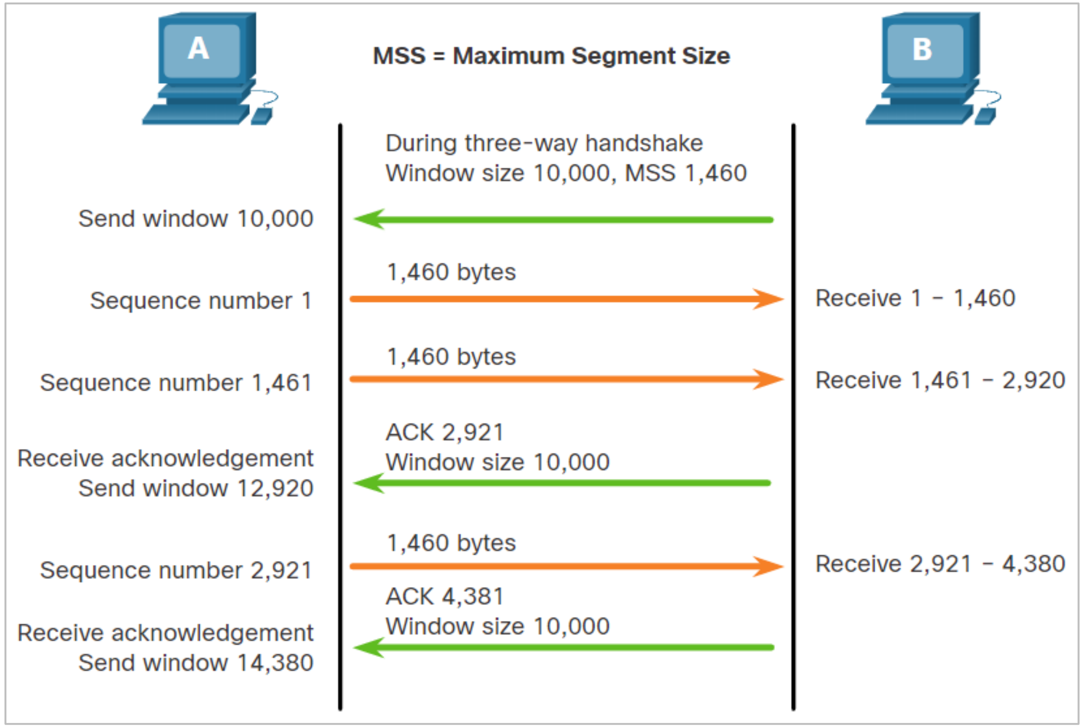
- Maximum Segment Size (MSS) is the maximum amount of data that the destination device can receive.
- A common MSS is 1,460 bytes when using IPv4. A host determines the value of its MSS field by subtracting the IP and TCP headers from the Ethernet maximum transmission unit (MTU).
- On an Ethernet interface, 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



Transport Layer Reliability

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.



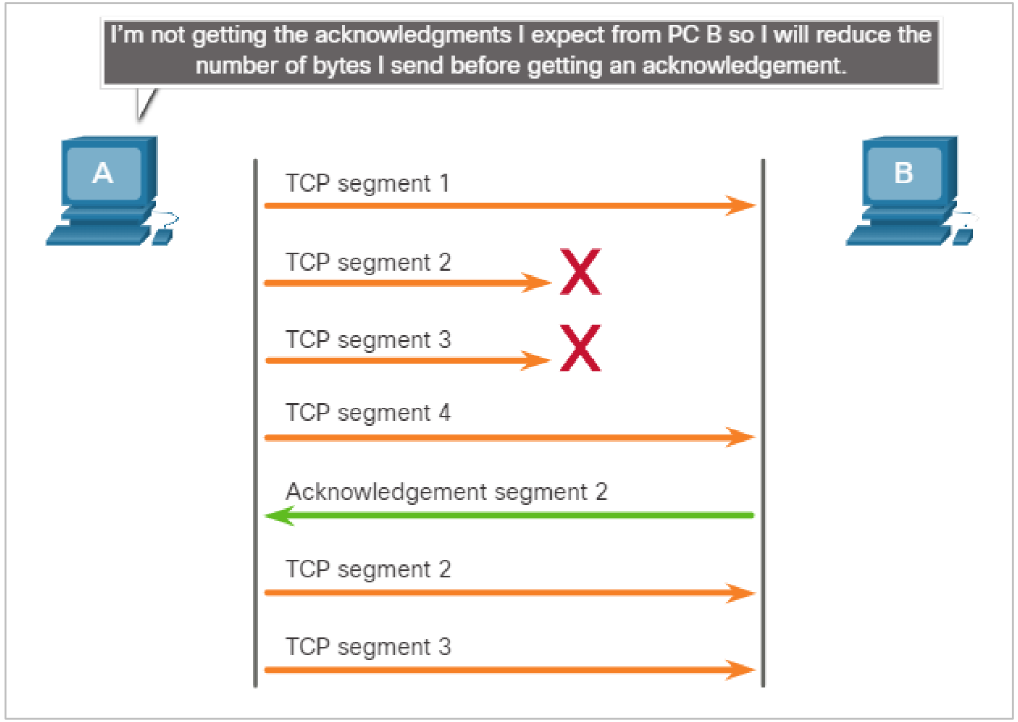
TCP Flow Control - Congestion Avoidance

- 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.

Transport Layer Reliability

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.

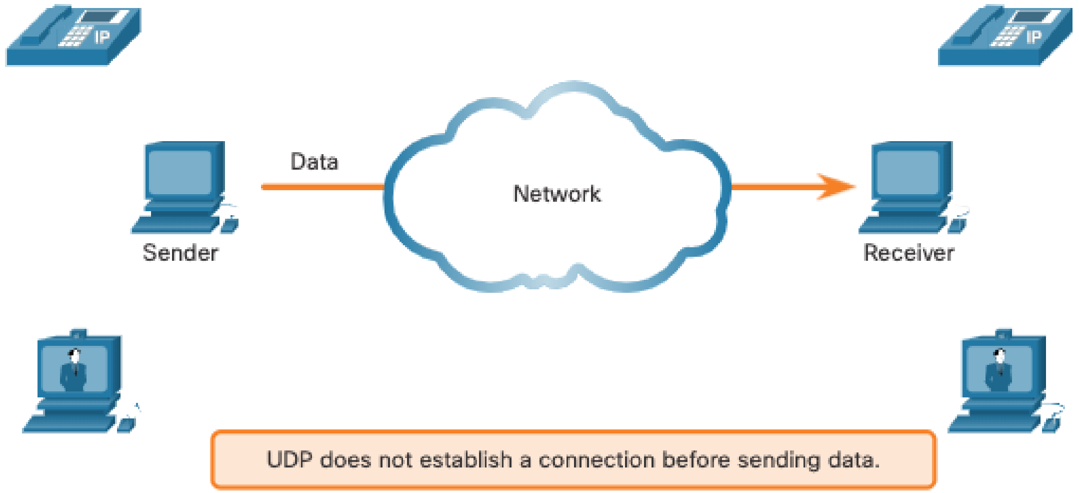

I'm not getting the acknowledgments I expect from PC B so I will reduce the number of bytes I send before getting an acknowledgement.
A
B
TCP segment 1
TCP segment 2
X
TCP segment 3
X
TCP segment 4
Acknowledgement segment 2
TCP segment 2
TCP segment 3

14.7 UDP İletişimi

UDP Communication

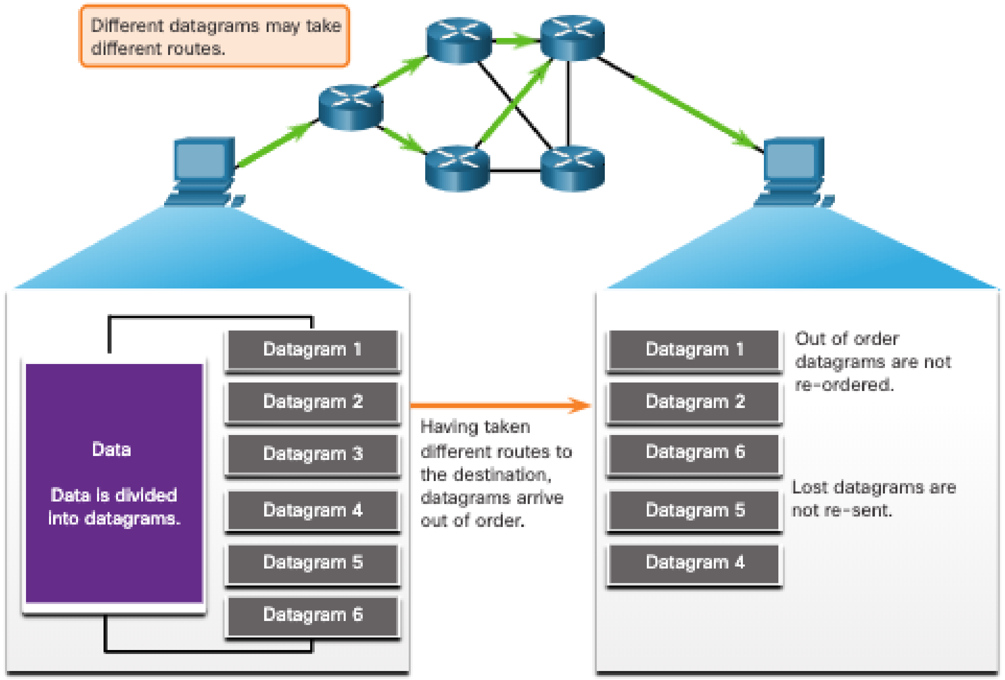
UDP Low Overhead versus Reliability

UDP does not establish a connection. UDP provides low overhead data transport because it has a small datagram header and no network management traffic.



UDP Communication
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.

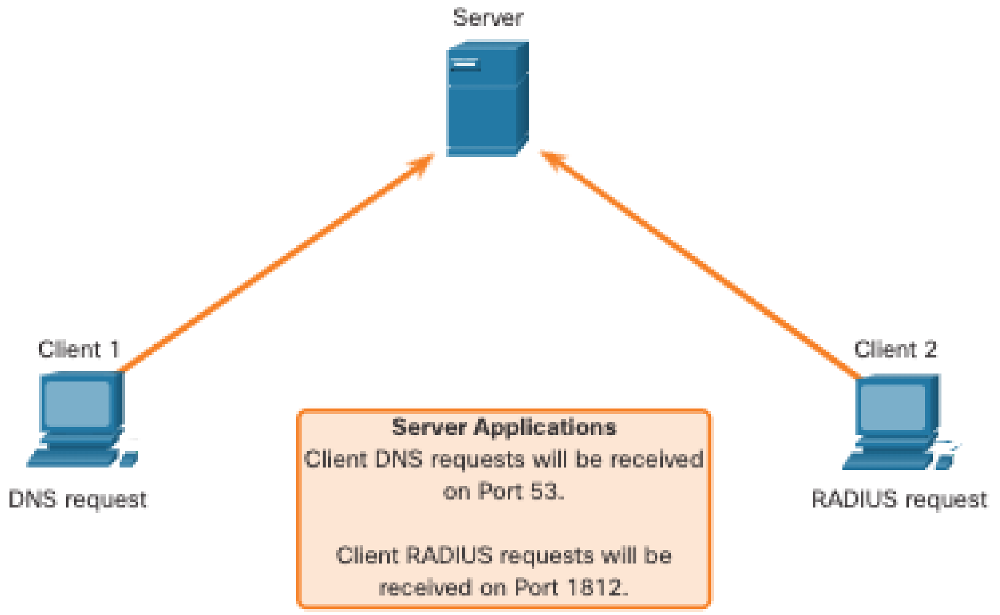

Different datagrams may take different routes.
Data
Data is divided into datagrams.
Datagram 1
Datagram 2
Datagram 3
Datagram 4
Datagram 5
Datagram 6
Having taken different routes to the destination, datagrams arrive out of order.
Datagram 1
Datagram 2
Datagram 6
Datagram 5
Datagram 4
Out of order datagrams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

UDP Communication

UDP Server Processes and Requests

UDP-based server applications are assigned well-known or registered port numbers.

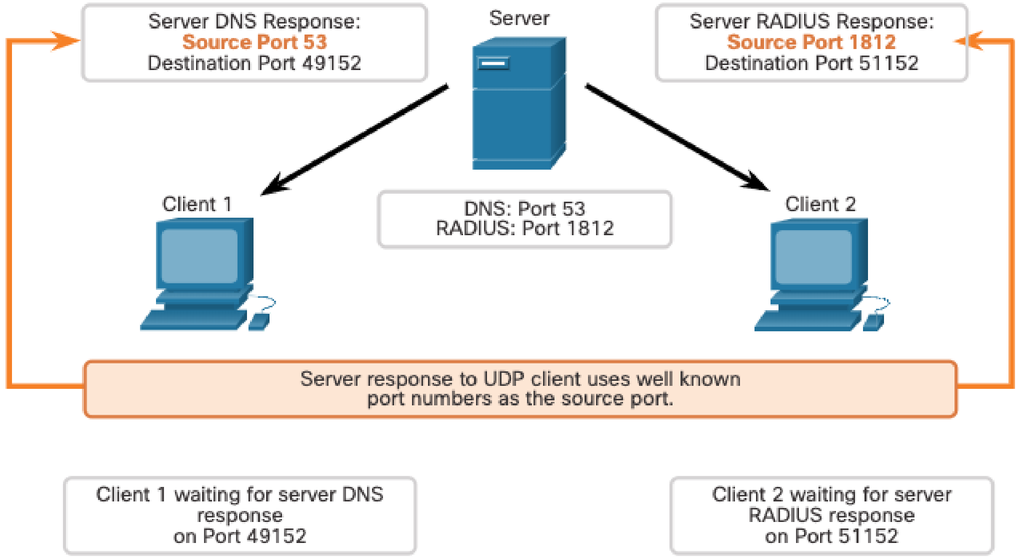
UDP receives a datagram destined for one of these ports, it forwards the application data to the appropriate application based on its port number.


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

UDP Communication

UDP Client Processes

- 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.


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.
- 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.

What did I learn in this module (Cont.)?

- 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.