Chapter 9: Transport Layer

CCNA Routing and Switching
Introduction to Networks v6.0
Chapter 9 - Sections & Objectives

9.1 Transport Layer Protocols
- Explain how transport layer protocols and services support communications across data networks.
- Explain the purpose of the transport layer in managing the transportation of data in end-to-end communication.
- Explain characteristics of the TCP and UDP protocols, including port numbers and their uses.

9.2 TCP and UDP
- Compare the operations of transport layer protocols in supporting end-to-end communication.
- Explain how TCP session establishment and termination 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.
- Determine whether high-reliability TCP transmissions, or non-guaranteed UDP transmissions, are best suited for common applications.
9.1 Transport Layer Protocols
Transportation of Data

Role of the Transport Layer

- Responsible for establishing a temporary communication session between two applications and delivering data between them.
- Link between the application layer and the lower layers that are responsible for network transmission.
Transportation of Data

Transport Layer Responsibilities

- **Tracking the Conversation** - Tracks each individual conversation flowing between a source and a destination application.

- **Segmentation** - Divides the data into segments that are easier to manage and transport. Header used for reassembly is used for tracking.

- **Identifying the Application** - Ensures that even with multiple applications running on a device, all applications receive the correct data via port numbers.
Segmenting the data into smaller chunks enables many different communications to be multiplexed on the same network.
Transportation of Data

Transport Layer Reliability

- TCP/IP provides two transport layer protocols:
  - Transmission Control Protocol (TCP)
    - Considered reliable which ensures that all of the data arrives at the destination.
    - Additional fields needed in header which increases size and delay.
  - User Datagram Protocol (UDP)
    - Does not provide for reliability.
    - Fewer fields and is faster than TCP.
TCP transport is similar to sending tracked packages. If a shipping order is broken up into several packages, a customer can check online to see the order of the delivery.
Transportation of Data
TCP (Cont.)
TCP Three Responsibilities:

- Numbering and tracking data segments
- Acknowledging received data
- Retransmitting any unacknowledged data after a certain period of time
Use UDP for less overhead and to reduce possible delays.

- Best-effort delivery (unreliable)
- No acknowledgment
- Similar to a non-registered letter
Transportation of Data

The Right Transport Layer Protocol for the Right Application

- **TCP** - databases, web browsers, and email clients require that all data that is sent arrives at the destination in its original condition.

- **UDP** - if one or two segments of a live video stream fail to arrive, if disruption in the stream, may not be noticeable to the user.

**UDP**

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

**TCP**

- **Required protocol properties:**
  - Reliable
  - Acknowledges data
  - Resends lost data
  - Delivers data in sequenced order
TCP and UDP Overview

TCP Features

- Establishing a Session
  - Connection-oriented protocol
  - Ensures the application is ready to receive the data
  - Negotiate the amount of traffic that can be forwarded at a given time

- Reliable Delivery
  - Ensuring that each segment that the source sends arrives at the destination

- Same-Order Delivery
  - Numbering & Sequencing the segments guarantees reassembly into the proper order

- Flow Control
  - Regulate the amount of data the source transmits
TCP and UDP Overview

TCP Header

- Source and Destination Port used to identify application
- Sequence number used for data reassembly
- Acknowledgement number indicates data has been received and ready for next byte from source
- Header length – length of TCP segment header
- Control bits – purpose and function of TCP segment
- Window size – number of bytes that can be accepted at one time
- Checksum – Used for error checking of segment header and data

20 Bytes Total
TCP and UDP Overview

UDP Features

Features of UDP

- Data is reconstructed in the order that it is received.
- Any segments lost are not resent.
- No session establishment.
- Does not inform the sender about resource availability.
TCP and UDP Overview

UDP Header

- UDP is a stateless protocol – no tracking
- Reliability handled by application
TCP and UDP Overview

Multiple Separate Communications

- Users expect to simultaneously receive and send email, view websites and make a VoIP phone call
- TCP and UDP manage multiple conversations by using unique identifiers called port numbers
### TCP and UDP Overview

#### Port Numbers

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Originating application port that is dynamically generated by sending device</td>
<td>- Tell the destination what service is being requested</td>
</tr>
<tr>
<td>- Example: Each separate HTTP conversation is tracked based on the source ports.</td>
<td>- Example: Port 80 web services are being requested</td>
</tr>
</tbody>
</table>

#### Port Numbers Table

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Electronic Mail</th>
<th>HTML Page</th>
<th>Internet Chat</th>
</tr>
</thead>
<tbody>
<tr>
<td>POP3</td>
<td>110</td>
<td>80</td>
<td>531</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protocol Type</th>
<th>Application Port</th>
<th>Data</th>
<th>Application Port</th>
<th>Data</th>
<th>Application Port</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>To: <a href="mailto:you@example.com">you@example.com</a> From: <a href="mailto:me@example.com">me@example.com</a> Subject: Email</td>
<td>Application Port</td>
<td>Data</td>
<td>Application Port</td>
<td>Data</td>
<td>Application Port</td>
<td>Data</td>
</tr>
</tbody>
</table>
TCP and UDP Overview

Socket Pairs

- Source and destination port placed in segment
- Segments encapsulated in IP packet
- IP and port number = socket
- Example: 192.168.1.7:80
- Sockets enable multiple processes to be distinguished
- Source port acts as a return address
TCP and UDP Overview

Port Number Groups

<table>
<thead>
<tr>
<th>Port Number Range</th>
<th>Port Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 to 1023</td>
<td>Well-known Ports</td>
</tr>
<tr>
<td>1024 to 49151</td>
<td>Registered Ports</td>
</tr>
<tr>
<td>49152 to 65535</td>
<td>Private and/or Dynamic Ports</td>
</tr>
</tbody>
</table>

- **Well-known Ports** (Numbers 0 to 1023) - These numbers are reserved for services and applications.

- **Registered Ports** (Numbers 1024 to 49151) - These port numbers are assigned by IANA to a requesting entity to use with specific processes or applications.

- **Dynamic or Private Ports** (Numbers 49152 to 65535) - Usually assigned dynamically by the client’s OS and used to identify the client application during communication.
### TCP and UDP Overview

#### Port Number Groups (Cont.)

<table>
<thead>
<tr>
<th>Port Number</th>
<th>Protocol</th>
<th>Application</th>
<th>Acronym</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>TCP</td>
<td>File Transfer Protocol (data)</td>
<td>FTP</td>
</tr>
<tr>
<td>21</td>
<td>TCP</td>
<td>File Transfer Protocol (control)</td>
<td>FTP</td>
</tr>
<tr>
<td>22</td>
<td>TCP</td>
<td>Secure Shell</td>
<td>SSH</td>
</tr>
<tr>
<td>23</td>
<td>TCP</td>
<td>Telnet</td>
<td>–</td>
</tr>
<tr>
<td>25</td>
<td>TCP</td>
<td>Simple Mail Transfer Protocol</td>
<td>SMTP</td>
</tr>
<tr>
<td>53</td>
<td>UDP, TCP</td>
<td>Domain Name Service</td>
<td>DNS</td>
</tr>
<tr>
<td>67</td>
<td>UDP</td>
<td>Dynamic Host Configuration Protocol (server)</td>
<td>DHCP</td>
</tr>
<tr>
<td>68</td>
<td>UDP</td>
<td>Dynamic Host Configuration Protocol (client)</td>
<td>DHCP</td>
</tr>
<tr>
<td>69</td>
<td>UDP</td>
<td>Trivial File Transfer Protocol</td>
<td>TFTP</td>
</tr>
<tr>
<td>80</td>
<td>TCP</td>
<td>Hypertext Transfer Protocol</td>
<td>HTTP</td>
</tr>
<tr>
<td>110</td>
<td>TCP</td>
<td>Post Office Protocol version 3</td>
<td>POP3</td>
</tr>
<tr>
<td>143</td>
<td>TCP</td>
<td>Internet Message Access Protocol</td>
<td>IMAP</td>
</tr>
<tr>
<td>161</td>
<td>UDP</td>
<td>Simple Network Management Protocol</td>
<td>SNMP</td>
</tr>
<tr>
<td>443</td>
<td>TCP</td>
<td>Hypertext Transfer Protocol Secure</td>
<td>HTTPS</td>
</tr>
</tbody>
</table>
TCP and UDP Overview

The netstat Command

- Network utility that can be used to verify connections
- By default, will attempt to resolve IP addresses to domain names and port numbers to well-known applications
- -n option used to display IPs and ports in 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:\>
```
9.2 TCP and UDP
TCP Communication Process

TCP Server Process

Clients Sending TCP Requests

- Client 1
  - HTTP Request: Source Port: 49152 Destination Port: 80
  - SMTP Request: Source Port: 51152 Destination Port: 25

- Client 2
  - HTTP Request: Source Port: 49152 Destination Port: 25
  - SMTP Request: Source Port: 51152 Destination Port: 25

Request Destination Ports

- Client 1
  - HTTP: Port 80
  - SMTP: Port 25

- Client 2

Request Source Ports

- Client 1
  - HTTP Request: Source Port: 49152 Destination Port: 80
  - SMTP Request: Source Port: 51152 Destination Port: 25

- Client 2

Use well known port numbers as the destination port.

Use dynamic port numbers as the source port.
TCP Communication Process

TCP Server Process (Cont.)

Response Destination Ports

HTTP response:
Source Port 80
Destination Port 49152

Response Source Ports

HTTP response:
Source Port 80
Destination Port 49152

SMTP Request:
Source Port 51152
Destination 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: Port 80
SMTP: Port 25

Client 1

HTTP Request:
Source Port: 49152
Destination Port: 80

SMTP Request:
Source Port: 51152
Destination Port: 25

Client 2

SMTP Response:
Source Port 25
Destination Port 51152
TCP Communication Process

TCP Connection Establishment

- **Step 1** – Initiating client requests a session with server.

- **Step 2** – Server acknowledges and requests a session with client.

- **Step 3** – Client acknowledges communication session with server.
To close a connection, the Finish (FIN) control flag must be set in the segment header.

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

To terminate a single conversation supported by TCP, four exchanges are needed to end both sessions.
TCP Communication Process

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.

- The six bits in the Control Bits field of the TCP segment header are also known as flags.
  - RST flag is used to reset a connection when an error or timeout occurs.
TCP Communication Process

Video Demonstration - TCP 3-Way Handshake

<table>
<thead>
<tr>
<th>No</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.16.303490</td>
<td>10.1.1.1</td>
<td>192.168.254.254</td>
<td>TCP</td>
<td>http &gt; kiosk [SYN] Seq=0 W</td>
</tr>
<tr>
<td>2</td>
<td>11.16.304896</td>
<td>192.168.254.254</td>
<td>10.1.1.1</td>
<td>TCP</td>
<td>http &gt; kiosk [SYN, ACK] Seq=0 W</td>
</tr>
<tr>
<td>3</td>
<td>12.16.304925</td>
<td>10.1.1.1</td>
<td>192.168.254.254</td>
<td>TCP</td>
<td>kiosk &gt; http [ACK] Seq=1 Ack</td>
</tr>
<tr>
<td>4</td>
<td>13.16.305153</td>
<td>10.1.1.1</td>
<td>192.168.254.254</td>
<td>HTTP</td>
<td>GET / HTTP/1.1</td>
</tr>
<tr>
<td>5</td>
<td>14.16.307875</td>
<td>192.168.254.254</td>
<td>10.1.1.1</td>
<td>TCP</td>
<td>http &gt; kiosk [ACK] Seq=1 Ack</td>
</tr>
</tbody>
</table>

Frame 10: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
Transmission Control Protocol, Src Port: kiosk (1061), Dst Port: http (80), Seq: 0, Len: 28 bytes
Source port: kiosk (1061)
Destination port: http (80)
[Stream index: 0]
Sequence number: 0 (relative sequence number)
Header length: 28 bytes
Flags: 0x02 (SYN)
Sequence numbers are assigned in the header of each packet.

Represents the first data byte of the TCP segment.

During session setup, an initial sequence number (ISN) is set - represents the starting value of the bytes.

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

Missing segments can then be identified.
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.
Reliability and Flow Control

TCP Flow Control – Congestion Avoidance

- Congestion causes retransmission of lost TCP segments
- Retransmission of segments can make the congestion worse
- To avoid and control congestion, TCP employs several congestion handling mechanisms, timers, and algorithms
- Example: Reduce the number of bytes it sends before receiving an acknowledgment

TCP Congestion Control

Acknowledgement numbers are for the next expected byte and not for a segment. The segment numbers used are simplified for illustration purposes.
UDP Communication

UDP Low Overhead versus Reliability

- UDP not connection-oriented
- No retransmission, sequencing, and flow control
- Functions not provided by the transport layer implemented elsewhere

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

UDP Datagram Reassembly

- UDP reassembles data in order received and forwards to application.
- Application must identify the proper sequence.

UDP: Connectionless and Unreliable
UDP Communication

UDP Server Processes and Requests

**Note:** The Remote Authentication Dial-in User Service (RADIUS) server shown in the figure provides authentication, authorization, and accounting services to manage user access.

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.
UDP Communication

UDP Client Processes

Clients Sending UDP Requests

Client 1 DNS Request: 
Source Port 49152 
Destination Port 53

Client 2 RADIUS User Authentication Request: 
Source Port 51152 
Destination Port 53

Server

DNS: Port 53 
RADIUS: Port 1812

Client 1

Client 2

Request Destination Ports

Use random port numbers as the source port.

Request Source Ports
UDP Communication

UDP Client Processes (Cont.)

Clients Sending UDP Requests
TCP or UDP

Applications that use TCP

TCP frees applications from having to manage reliability
TCP or UDP

Applications that use UDP

Three types of applications best suited for UDP:

- Live video and multimedia
- Simple request and reply
- Handle reliability themselves
9.3 Chapter Summary
Chapter 9: Transport Layer

- Explain how transport layer protocols and services support communications across data networks.
- Compare the operations of transport layer protocols in supporting end-to-end communication.