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 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.
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

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
Applications communicate 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 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.
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 exchanged between the sending and receiving hosts. This control information is contained in a TCP header.
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.
Application developers choose the appropriate transport layer protocol based on the nature of the application.
7.1.2.1 Introducing TCP

TCP Provides

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

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.
7.1.2.2 Role of TCP

- **Sequence number** (32 bits) - data reassembly purposes.
- **Acknowledgement 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 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.
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.
Data for different applications is directed to the correct application because each application has a unique port number.
• 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.)

<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>

#### Registered TCP Ports:
- 1863 MSN Messenger
- 2000 Cisco SCCP (VoIP)
- 8008 Alternate HTTP
- 8080 Alternate HTTP

#### Registered UDP Ports:
- 1812 RADIUS Authentication Protocol
- 5004 RTP (Voice and Video Transport Protocol)
- 5060 SIP (VoIP)

#### Registered TCP/UDP Common Ports:
- 1433 MS SQL
- 2948 WAP (MMS)

#### Well-known TCP Ports:
- 21 FTP
- 23 Telnet
- 25 SMTP
- 80 HTTP
- 143 IMAP
- 194 Internet Relay Chat (IRC)
- 443 Secure HTTP (HTTPS)

#### Well-known UDP Ports:
- 69 TFTP
- 520 RIP

#### 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.)

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.
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

TCP
- Sequenced Message Segments
- Flow Control
- Guaranteed Delivery
- Ordered Delivery
- Session Establishment

UDP
- Less Overhead
- Connectionless
- Fast Transmission Requirements
- No Ordered Delivery
- No Acknowledgement of Receipt
7.2.1.1 TCP Reliable Delivery

Source to destination connection between processes
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

TCP Connection Establishment

- 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

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.
7.2.1.6 TCP Three-way Handshake Analysis - Step 3

Step 3: The initiating client acknowledges the server-to-client communication session.
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.
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:
- Send SYN
  - SEQ = 600
  - CTL = SYN
- SYN Received
- Established
  - SEQ = 601
  - ACK = 401
- (CTL = which bits in the TCP header are set to 1)

4 Step Process of TCP Termination Session:
- Send FIN
  - Received
- ACK Received
- FIN Received
- Send ACK
  - Received
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

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 6

Having taken different routes to the destination, segments arrive out of order.
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.
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.
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.
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

Different datagrams may take different routes.

UDP: Connectionless and Unreliable

Data is divided into datagrams.

Datagram 1
Datagram 2
Datagram 3
Datagram 4
Datagram 5
Datagram 6

Datagram 1
Datagram 2
Datagram 3
Datagram 4
Datagram 5
Datagram 6

Out of order datagrams are not re-ordered.

Lost datagrams are not re-sent.

Having taken different routes to the destination, datagrams arrive out of order.
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

Client 1 waiting for server DNS response on Port 49152

Client 2 waiting for server RADIUS response on Port 51152

Server response to UDP client uses well known port numbers as the source port.
Using Wireshark to Examine a UDP DNS Capture
7.2.4.1 Applications that use TCP

Applications that use TCP

FTP          SMTP

HTTP          Telnet

TCP

IP
Applications that use UDP

SNMP*  TFTP
DNS*    VoIP
DHCP    IPTV

*These applications can also use TCP.
Using Wireshark to Examine FTP and TFTP Captures
7.2.4.4 Activity - TCP, UDP or Both

Transport Layer Delivery Method

**TCP**
- HTTP
- Telnet
- FTP
- SMTP

**UDP**
- DHCP
- VoIP
- TFTP
- IPTV

**Both**
- SNMP
- DNS
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

TCP and UDP Communications
The transport layer prepares application data for transport over the network and processes network data for use by applications.
Thanks for your attention!!