OSI Transport Layer

Network Fundamentals – Chapter 4

Cisco | Networking Academy®
Mind Wide Open™

Objectives

Explain the role of Transport Layer protocols and services in supporting communications across data networks

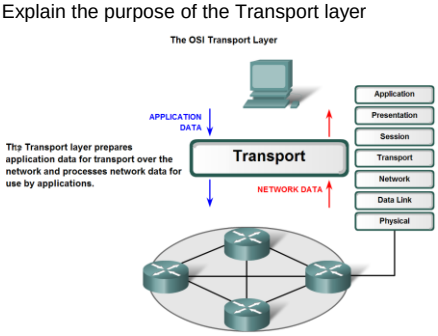
Analyze the application and operation of TCP mechanisms that support reliability

Analyze the application and operation of TCP mechanisms that support reassembly and manage data loss.

Analyze the operation of UDP to support communicate between two processes on end devices

Transport Layer Role and Services

Explain the purpose of the Transport layer



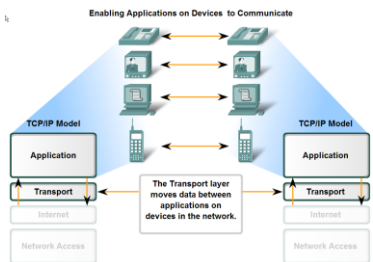
Transport Layer Functions

The Transport layer provides for the segmentation of data and the control necessary to reassemble these pieces into the various communication streams. Its primary responsibilities to accomplish this are:

1. Tracking the individual communication between applications on the source and destination hosts
2. Segmenting data and managing each piece
3. Reassembling the segments into streams of application data
4. Identifying the different applications

Transport Layer Role and Services

Major functions of the transport layer and the role it plays in data networks



Tracking Individual Conversations

Any host may have multiple applications that are communicating across the network.

Each of these applications will be communicating with one or more applications on remote hosts.

It is the responsibility of the Transport layer to maintain the multiple communication streams between these applications.

Segmenting Data

As each application creates a stream data to be sent to a remote application, this data must be prepared to be sent across the media in manageable pieces.

The Transport layer protocols describe services that segment this data from the Application layer.

This includes the encapsulation required on each piece of data.

Each piece of application data requires headers to be added at the Transport layer to indicate to which communication it is associated.

Reassembling Segments

At the receiving host, each piece of data may be directed to the appropriate application.

Additionally, these individual pieces of data must also be reconstructed into a complete data stream that is useful to the Application layer.

The protocols at the Transport layer describe the how the Transport layer header information is used to reassemble the data pieces into streams to be passed to the Application layer.

Identifying Applications

In order to pass data streams to the proper applications, the Transport layer must identify the target application.

To accomplish this, the Transport layer assigns an application an identifier. The TCP/IP protocols call this identifier a port number.

Each software process that needs to access the network is assigned a port number unique in that host.

This port number is used in the Transport layer header to indicate to which application that piece of data is associated.

Applications do not need to know the operational details of the network in use.

The applications generate data that is sent from one application to another, without regard to the destination host type, the type of media over which the data must travel, the path taken by the data, the congestion on a link, or the size of the network.

Data Requirements Vary

Because different applications have different requirements, there are multiple Transport layer protocols. For some applications, segments must arrive in a very specific sequence in order to be processed successfully.

The different Transport layer protocols have different rules allowing devices to handle these diverse data requirements.

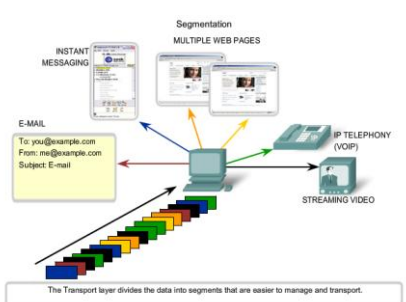
Some protocols provide just the basic functions for efficiently delivering the data pieces between the appropriate applications.

These types of protocols are useful for applications whose data is sensitive to delays.

Other Transport layer protocols describe processes that provide additional features, such as ensuring reliable delivery between the applications.

One problem – Overhead!

Segmentation



Segmentation

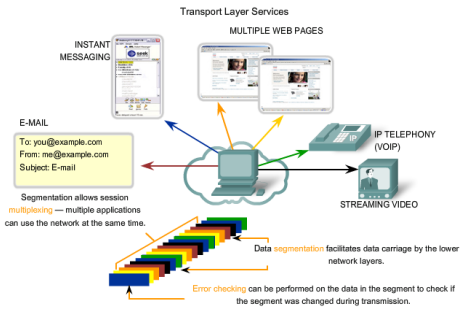
Segmentation of the data, in accordance with Transport layer protocols, provides the means to both send and receive data when running multiple applications concurrently on a computer.

Without segmentation, only one application, the streaming video for example, would be able to receive data.

You could not receive e-mails, chat on instant messenger, or view web pages while also viewing the video.

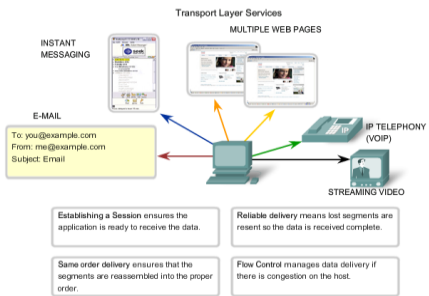
At the Transport layer, each particular set of pieces flowing between a source application and a destination application is known as a conversation.

Transport Layer Services



 13

Services in Action



 14

Establishing a Session and Reliable Delivery

1. The Transport layer can provide this connection orientation by creating a sessions between the applications. These connections prepare the applications to communicate with each other before any data is transmitted.

   Within these sessions, the data for a communication between the two applications can be closely managed.

2. For many reasons, it is possible for a piece of data to become corrupted, or lost completely, as it is transmitted over the network. The Transport layer can ensure that all pieces reach their destination by having the source device to retransmit any data that is lost.

 15

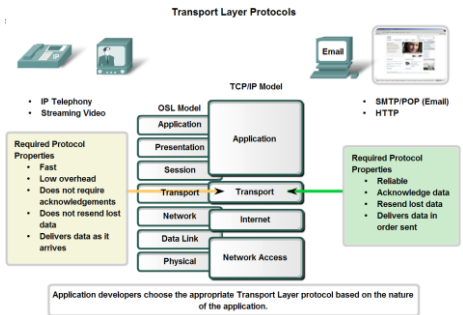
Same order Delivery and Flow Control

1. Because networks may provide multiple routes that can have different transmission times, data can arrive in the wrong order. By numbering and sequencing the segments, the Transport layer can ensure that these segments are reassembled into the proper order.
2. Network hosts have limited resources, such as memory or bandwidth. When Transport layer is aware that these resources are overtaxed, some protocols can request that the sending application reduce the rate of data flow. This is done at the Transport layer by regulating the amount of data the source transmits as a group. Flow control can prevent the loss of segments on the network and avoid the need for retransmission.

 16

Transport Layer Role and Services

Supporting Reliable Communication



Transport Layer Operations

At the Transport layer the three basic operations of reliability are:

1. tracking transmitted data
2. acknowledging received data
3. retransmitting any unacknowledged data

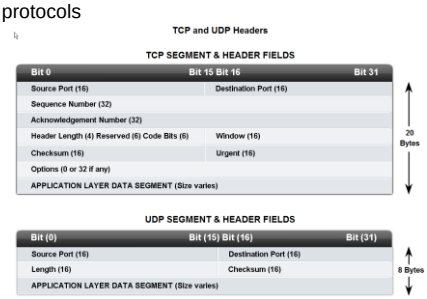
These reliability processes place additional overhead on the network resources due to the acknowledgement, tracking, and retransmission.

This creates a trade-off between the value of reliability and the burden it places on the network.

Application developers must choose which transport protocol type is appropriate based on the requirements of their applications

Transport Layer Role and Services

Identify the basic characteristics of the UDP and TCP protocols



UDP

UDP is a simple, connectionless protocol, described in RFC 768. It has the advantage of providing for low overhead data delivery.

The pieces of communication in UDP are called datagrams.

These datagrams are sent as "best effort" by this Transport layer protocol.

Applications that use UDP include:

1. Domain Name System (DNS)
2. Video Streaming
3. Voice over IP (VoIP)

TCP

TCP is a connection-oriented protocol, described in RFC 793. TCP incurs additional overhead to gain functions.

Additional functions specified by TCP are the same order delivery, reliable delivery, and flow control.

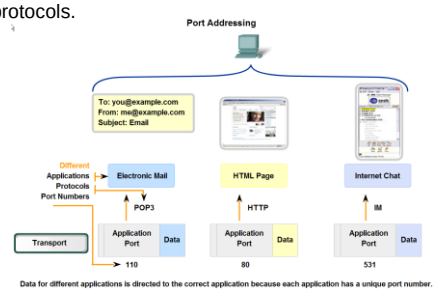
Each TCP segment has 20 bytes of overhead in the header encapsulating the Application layer data, whereas each UDP segment only has 8 bytes of overhead.

Applications that use TCP are:

1. Web Browsers
2. E-mail and File Transfers

Transport Layer Role and Services

Identify how a port number is represented and describe the role port numbers play in the TCP and UDP protocols.



Transport Port Numbers

In the header of each segment or datagram, there is a source and destination port.

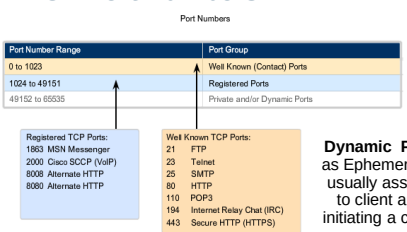
The source port number is the number for this communication associated with the originating application on the local host.

The destination port number is the number for this communication associated with the destination application on the remote host.

The combination of the Transport layer port number and the Network layer IP address assigned to the host uniquely identifies a particular process running on a specific host device. This combination is called a socket.

A socket pair, consisting of the source and destination IP addresses and port numbers, is also unique and identifies the conversation between the two hosts.

TCP Port Numbers



Dynamic Ports - Also known as Ephemeral Ports, these are usually assigned dynamically to client applications when initiating a connection. It is not very common for a client to connect to a service using a Dynamic or Private Port

UDP Port Numbers

Port Numbers

Port Number Range	Port Group
0 to 1023	Well Known (Contact) Ports
1024 to 49151	Registered Ports
49152 to 65535	Private and/or Dynamic Ports

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

UDP and TCP Port Numbers

Port Numbers

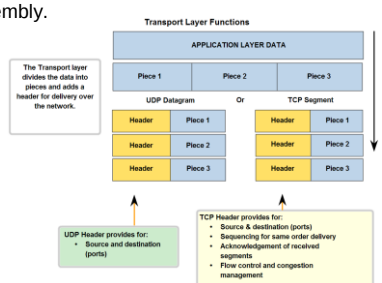
Port Number Range	Port Group
0 to 1023	Well Known (Contact) Ports
1024 to 49151	Registered Ports
49152 to 65535	Private and/or Dynamic Ports

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

Transport Layer Role and Services

Describe the role of segments in the transport layer and the two principle ways segments can be marked for reassembly.

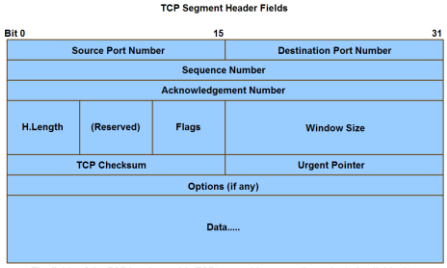


TCP and UDP Segmentation

1. In TCP, each segment header contains a sequence number. This sequence number allows the Transport layer functions on the destination host to reassemble segments in the order in which they were transmitted. This ensures that the destination application has the data in the exact form the sender intended.
2. Although services using UDP also track the conversations between applications, they are not concerned with the order in which the information was transmitted, or in 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.

Application and Operation of TCP Mechanisms

Trace the steps that show how the TCP reliability mechanism works as part of a session



The fields of the TCP header enable TCP to provide connection-oriented, reliable data communications.

Reliability

The key distinction between TCP and UDP is reliability. The reliability of TCP communication is performed using connection-oriented sessions.

Before a host using TCP sends data to another host, the Transport layer initiates a process to create a connection with the destination.

This connection enables the tracking of a session, or communication stream between the hosts.

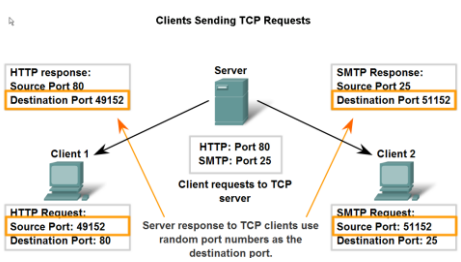
This process ensures that each host is aware of and prepared for the communication. A complete TCP conversation requires the establishment of a session between the hosts in both directions.

After a session has been established, the destination sends acknowledgements to the source for the segments that it receives.

If the source does not receive an acknowledgement within a predetermined amount of time, it retransmits that data to the destination.

Application and Operation of TCP Mechanisms

Describe the role of port numbers in establishing TCP sessions and directing segments to server process



Server Services

Each application process running on the server is configured to use a port number, either by default or manually by a system administrator.

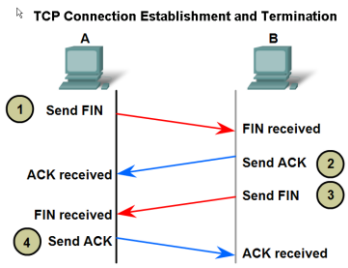
An individual server cannot have two services assigned to the same port number within the same Transport layer services.

A host running a web server application and a file transfer application cannot have both configured to use the same port (for example, TCP port 8080).

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

Application and Operation of TCP Mechanisms

Trace the steps in the handshake in the establishment of TCP sessions



Three-way handshake

To establish the connection, the hosts perform a three-way handshake. Control bits in the TCP header indicate the progress and status of the connection. The three-way handshake:

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

Steps in Action

1. The initiating client sends a segment containing an initial sequence value, which serves as a request to the server to begin a communications session.
2. The server responds with a segment containing an acknowledgement value equal to the received sequence value plus 1, plus its own synchronizing sequence value. The value is one greater than the sequence number because the ACK is always the next expected Byte or Octet. This acknowledgement value enables the client to tie the response back to the original segment that it sent to the server.
3. Initiating client responds with an acknowledgement value equal to the sequence value it received plus one. This completes the process of establishing the connection.

Diagram

Control Information

Within the TCP segment header, there are six 1-bit fields that contain control information used to manage the TCP processes. Those fields are:

URG - Urgent pointer field significant

ACK - Acknowledgement field significant

PSH - Push function

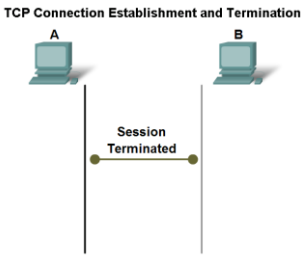
RST - Reset the connection

SYN - Synchronize sequence numbers

FIN - No more data from sender

Application and Operation of TCP Mechanisms

Trace the steps in the handshake in the termination of TCP sessions



Closing a Connection

To close a connection, the FIN (Finish) control flag in the segment header must be set.

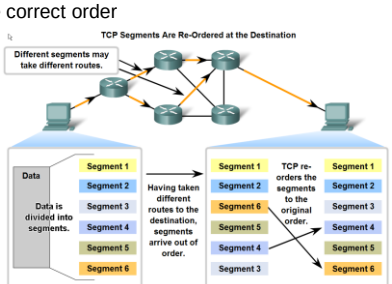
To end each one-way TCP session, a two-way handshake is used, consisting of a FIN segment and an ACK segment.

Therefore, to terminate a single conversation supported by TCP, four exchanges are needed to end both sessions

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

Managing TCP Sessions

Describe how TCP sequence numbers are used to reconstruct the data stream with segments placed in the correct order



Managing Sequence Numbers

Segment sequence numbers enable reliability by indicating how to reassemble and reorder received segments, as shown in the figure.

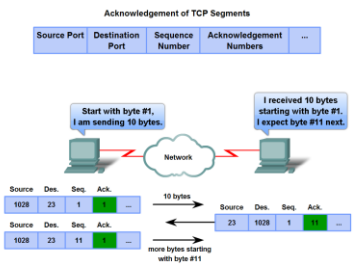
The receiving TCP process places the data from a segment into a receiving buffer. Segments are placed in the proper sequence number order and passed to the Application layer when reassembled.

Any segments that arrive with non-contiguous sequence numbers are held for later processing.

Then, when the segments with the missing bytes arrive, these segments are processed.

Managing TCP Sessions

Trace the steps used by the TCP protocol in which sequence numbers and acknowledgement numbers are used to manage exchanges in a conversation



Managing Acknowledgements

The sequence number is the relative number of bytes that have been transmitted in this session plus 1 (which is the number of the first data byte in the current segment).

TCP uses the acknowledgement number in segments sent back to the source to indicate the next byte in this session that the receiver expects to receive.

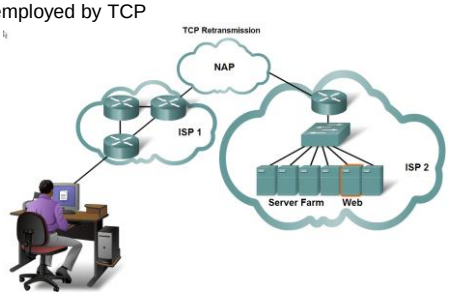
This is called expectational acknowledgement.

The source is informed that the destination has received all bytes in this data stream up to, but not including, the byte indicated by the acknowledgement number.

The sending host is expected to send a segment that uses a sequence number that is equal to the acknowledgement number.

Managing TCP Sessions

Describe the retransmission.remedy for lost data employed by TCP



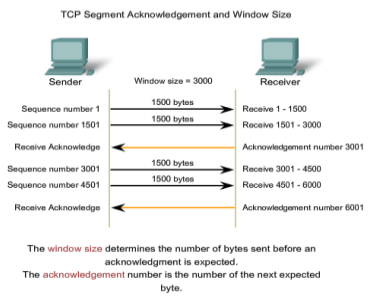
Handling Retransmissions

A destination host service using TCP usually only acknowledges data for contiguous sequence bytes. If one or more segments are missing, only the data in the segments that complete the stream are acknowledged.

For example, if segments with sequence numbers 1500 to 3000 and 3400 to 3500 were received, the acknowledgement number would be 3001. This is because there are segments with the sequence numbers 3001 to 3399 that have not been received.

When TCP at the source host has not received an acknowledgement after a predetermined amount of time, it will go back to the last acknowledgement number that it received and retransmit data from that point forward.

Flow Control with Window Size



Flow Control

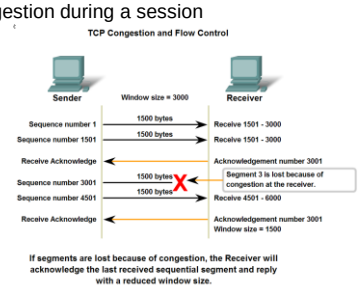
TCP also provides mechanisms for flow control. Flow control assists the reliability of TCP transmission by adjusting the effective rate of data flow between the two services in the session.

When the source is informed that the specified amount of data in the segments is received, it can continue sending more data for this session.

This Window Size field in the TCP header specifies the amount of data that can be transmitted before an acknowledgement must be received. The initial window size is determined during the session startup via the three-way handshake.

Managing TCP Sessions

Describe the mechanisms in TCP that manage the interrelationship between window size, data loss and congestion during a session



Reducing Window Size

If a receiving host has congestion, it may respond to the sending host with a segment with a reduced window size. In this graphic, there was a loss of one of the segments.

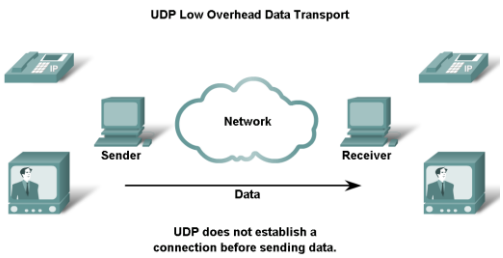
The receiver changed the window field in the TCP header of the returning segments in this conversation from 3000 down to 1500. This caused the sender to reduce the window size to 1500.

After periods of transmission with no data losses or constrained resources, the receiver will begin to increase the window field.

This reduces the overhead on the network because fewer acknowledgments need to be sent. Window size will continue to increase until there is data loss, which will cause the window size to be decreased.

UDP Protocol

Describe the characteristics of the UDP protocol and the types of communication for which it is best suited



UDP Protocol

UDP is a simple protocol that provides the basic Transport layer functions.

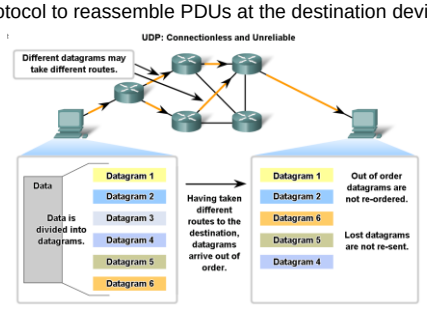
It much lower overhead than TCP, since it is not connection-oriented and does not provide the sophisticated retransmission, sequencing, and flow control mechanisms.

Although the total amount of UDP traffic found on a typical network is often relatively low, key Application layer protocols that use UDP include:

1. Domain Name System (DNS)
2. Simple Network Management Protocol (SNMP)
3. Dynamic Host Configuration Protocol (DHCP)
4. Routing Information Protocol (RIP)
5. Trivial File Transfer Protocol (TFTP)
6. Online games

UDP Protocol

Describe in detail the process specified by the UDP protocol to reassemble PDUs at the destination device



UDP and Datagrams

Because UDP is connectionless, sessions are not established before communication takes place as they are with TCP.

UDP is said to be transaction-based. In other words, when an application has data to send, it simply sends the data.

Many applications that use UDP send small amounts of data that can fit in one segment. However, some applications will send larger amounts of data that must be split into multiple segments.

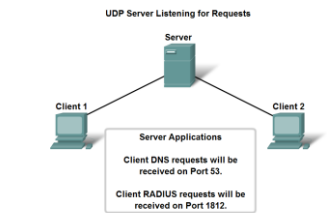
The UDP PDU is referred to as a datagram, although the terms segment and datagram are sometimes used interchangeably to describe a Transport layer PDU.

When multiple datagrams are sent to a destination, they may take different paths and arrive in the wrong order.

UDP does not keep track of sequence numbers the way TCP does. UDP has no way to reorder the datagrams into their transmission order.

UDP Protocol

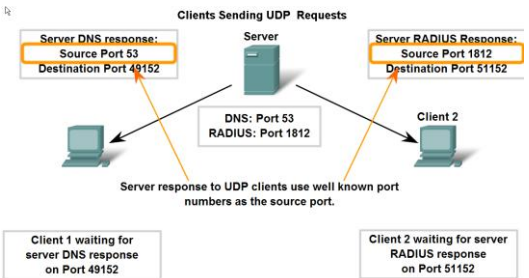
Describe how servers use port numbers to identify a specified application layer process and direct segments to the proper service or application



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

UDP Protocol

Trace the steps as the UDP protocol and port numbers are utilized in client-server communication.



UDP in Action

The UDP client process randomly selects a port number from the dynamic range of port numbers and uses this as the source port for the conversation.

The destination port will usually be the Well Known or Registered port number assigned to the server process.

Randomized source port numbers also help with security.

If there is a predictable pattern for destination port selection, an intruder can more easily simulate access to a client by attempting to connect to the port number most likely to be open.

Because there is no session to be created with UDP, as soon as the data is ready to be sent and the ports identified, UDP can form the datagram and pass it to the Network layer to be addressed and sent on the network.

Summary

In this chapter, you learned to:

- Explain the need for the Transport layer
- Identify the role of the Transport layer as it provides the end-to-end transfer of data between applications
- Describe the role of two TCP/IP Transport layer protocols, TCP and UDP
- Explain the key functions of the Transport layer including reliability, port addressing, and segmentation
- Explain how TCP and UDP each handle these key functions
- Identify when it is appropriate to use TCP or UDP and provide examples of applications that use each protocol