UNIT - 4

Transport Layer

Outline

3.1 INTRODUCTION

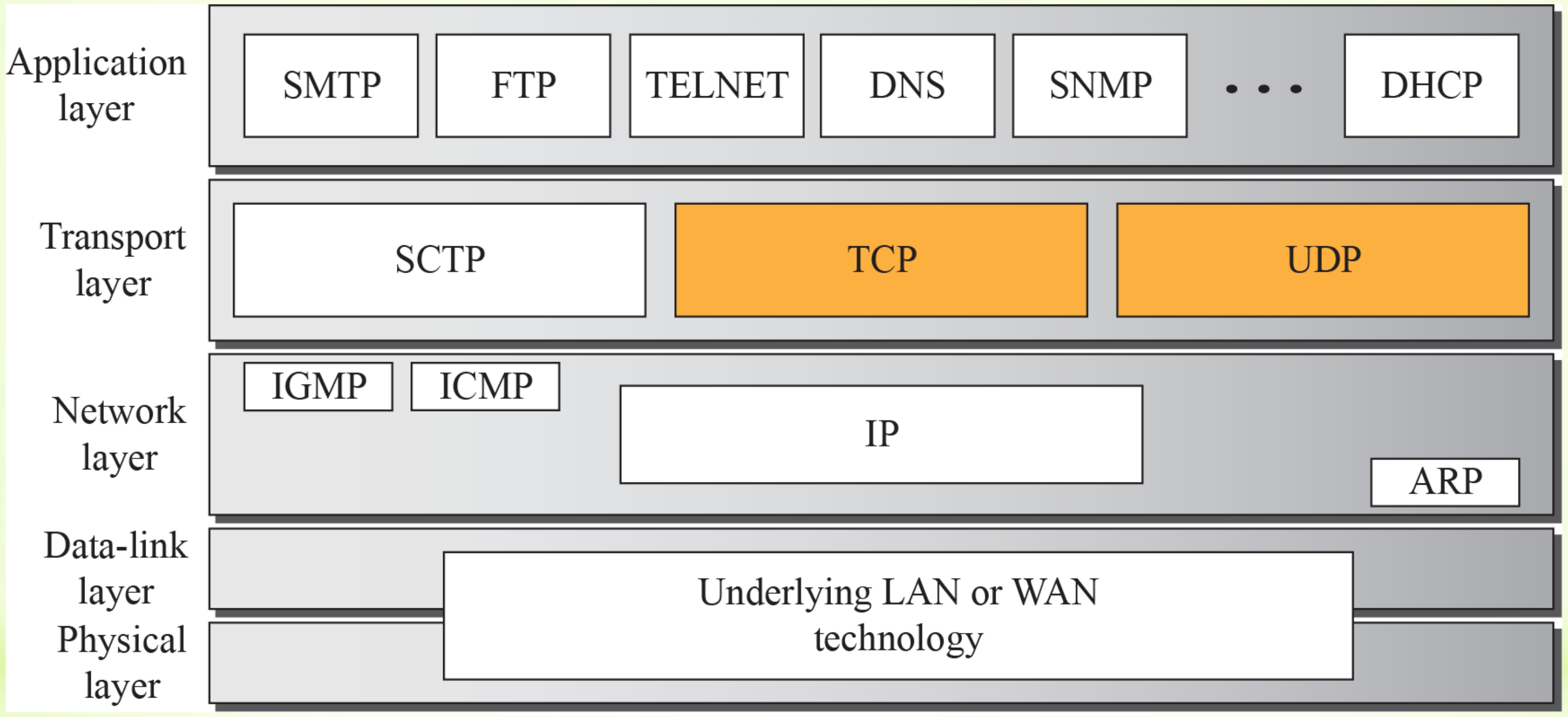
3.2 TRANSPORT-LAYER PROTOCOLS

- Simple Protocol
- Stop-and-Wait Protocol
- Go-Back-N Protocol (GBN)
- Selective-Repeat Protocol
- Bidirectional Protocols: Piggybacking

3.3 INTERNET TRANSPORT-LAYER PROTOCOLS

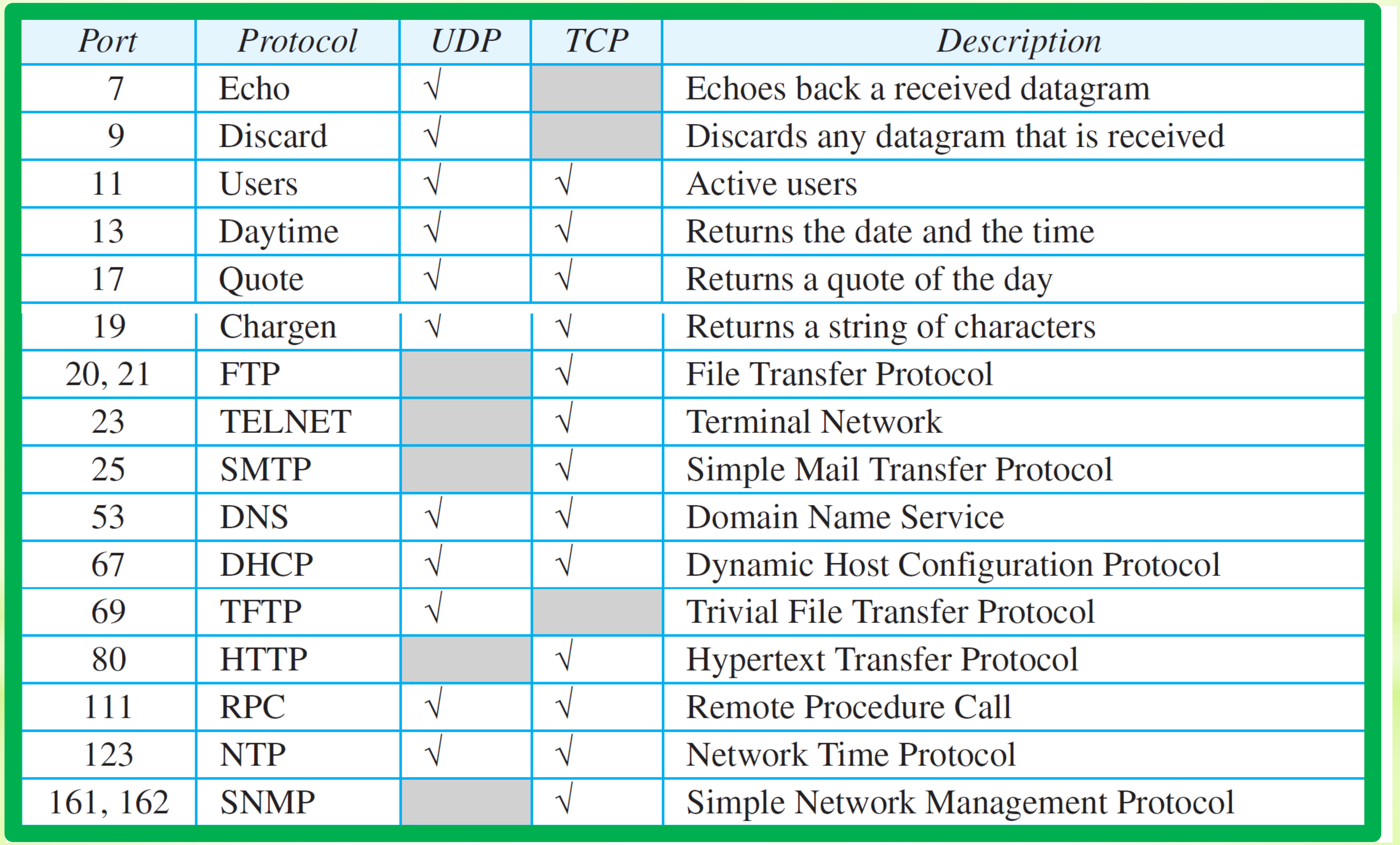
- USER DATAGRAM PROTOCOL (UDP)
- TRANSMISSION CONTROL PROTOCOL (TCP)

Internet Transport-Layer Protocols



Position of transport-layer protocols in the TCP/IP protocol suite

Some well-known ports used with UDP and TCP



Port	*Protocol*	*UDP*	*TCP*	*Description*
7	Echo	√		Echoes back a received datagram
9	Discard	√		Discards any datagram that is received
11	Users	√	√	Active users
13	Daytime	√	√	Returns the date and the time
17	Quote	√	√	Returns a quote of the day
19	Chargen	√	√	Returns a string of characters
20, 21	FTP		√	File Transfer Protocol
23	TELNET		√	Terminal Network
25	SMTP		√	Simple Mail Transfer Protocol
53	DNS	√	√	Domain Name Service
67	DHCP	√	√	Dynamic Host Configuration Protocol
69	TFTP	√		Trivial File Transfer Protocol
80	HTTP		√	Hypertext Transfer Protocol
111	RPC	√	√	Remote Procedure Call
123	NTP	√	√	Network Time Protocol
161, 162	SNMP		√	Simple Network Management Protocol

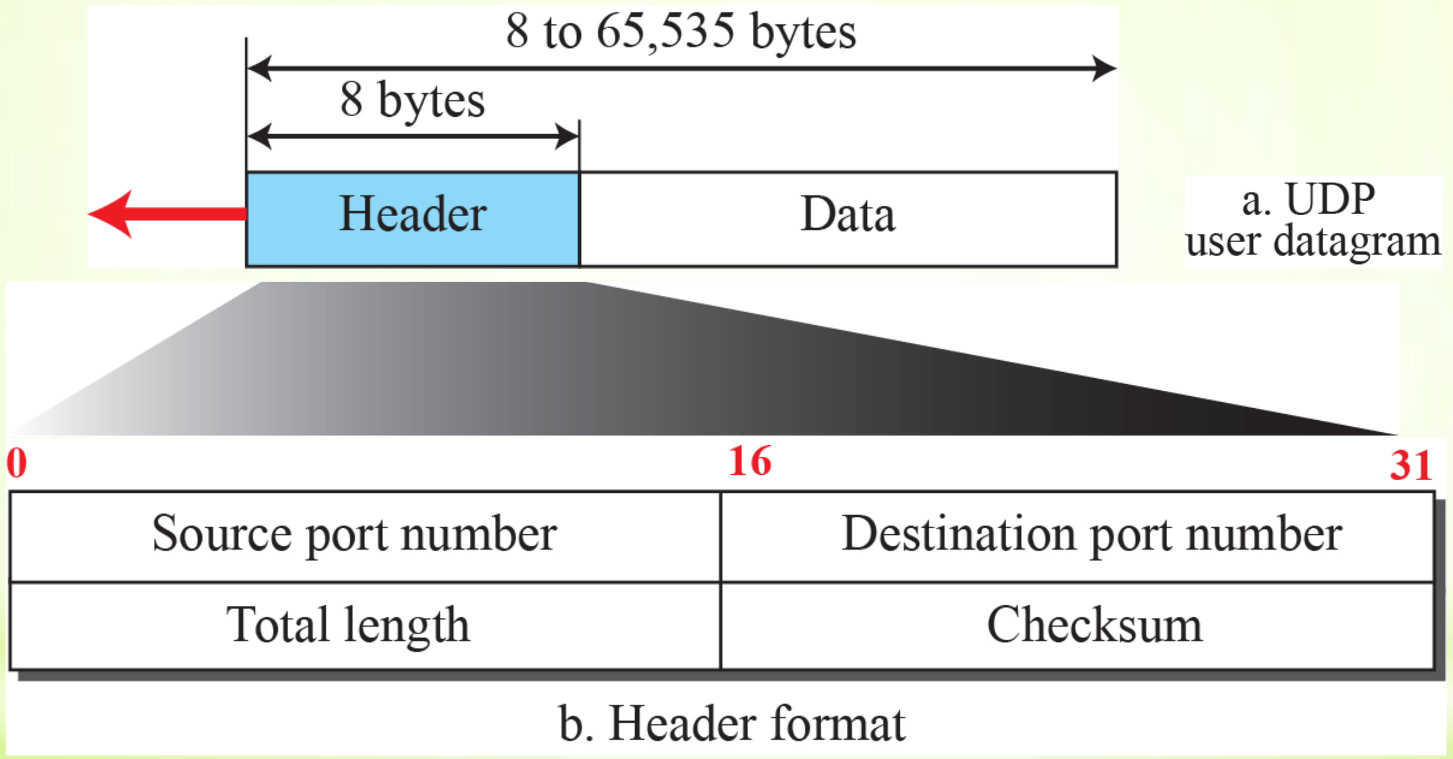
USER DATAGRAM PROTOCOL (UDP)

- ***The User Datagram Protocol (UDP) is a connectionless, unreliable transport protocol.***
- ***It does not add anything to the services of IP except for providing process-to-process instead of host-to-host communication.***
- ***UDP is a very simple protocol using a minimum of overhead.***

User Datagram

UDP packets, called user datagrams, have a fixed size header of 8 bytes made of four fields, each of 2 bytes (16 bits). the format of a user datagram. The first two fields define the source and destination port numbers. The third field defines the total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65,535 bytes.

User datagram packet format



Example

The following is the contents of a UDP header in hexadecimal format.

```
CB84000D001C001C
```

a. What is the source port number?
b. What is the destination port number?
c. What is the total length of the user datagram?
d. What is the length of the data?
e. Is the packet directed from a client to a server or vice versa?
f. What is the client process?

Solution

a. The source port number is the first four hexadecimal digits $(CB84)_{16}$ or 52100

b. The destination port number is the second four hexadecimal digits $(000D)_{16}$ or 13.

c. The third four hexadecimal digits $(001C)_{16}$ define the length of the whole UDP packet as 28 bytes.

d. The length of the data is the length of the whole packet minus the length of the header, or $28 - 8 = 20$ bytes.

e. Since the destination port number is 13 (well-known port), the packet is from the client to the server.

f. The client process is the Daytime (see Table 3.1).

UDP Services

Earlier we discussed the general services provided by a transport-layer protocol. In this section, we discuss what portions of those general services are provided by UDP.

- Process-to-Process Communication

UDP provides process-to-process communication using socket addresses, a combination of IP addresses and port numbers.

UDP Services

- Connectionless Services
  - There is no connection establishment and no connection termination
  - This means that each user datagram sent by UDP is an independent datagram. There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program.
  - The user datagrams are not numbered.
  - This means that each user datagram can travel on a different path.
  - Only those processes sending short messages, messages less than 65,507 bytes (65,535 minus 8 bytes for the UDP header and minus 20 bytes for the IP header), can use UDP.

UDP Services

- Flow Control

  UDP is a very simple protocol. There is no flow control, and hence no window mechanism.

- Error Control

  - There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated.
  - When the receiver detects an error through the checksum, the user datagram is silently discarded.

- Congestion Control

  - Since UDP is a connectionless protocol, it does not provide congestion control.
  - UDP assumes that the packets sent are small and sporadic and cannot create congestion in the network.

❑ Checksum

UDP checksum calculation includes three sections: a pseudoheader, the UDP header, and the data coming from the application layer.

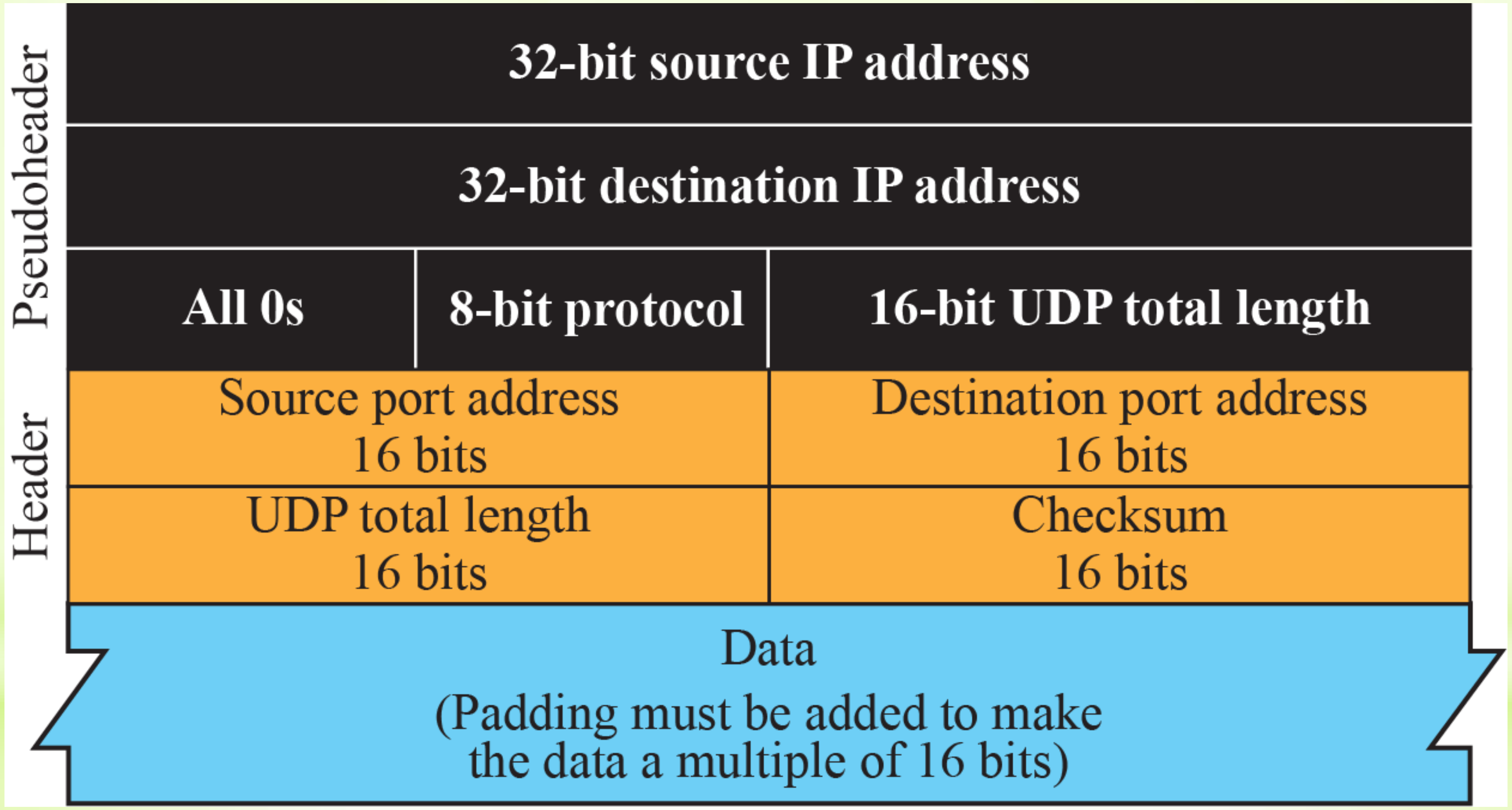
The pseudoheader is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with 0s.

If the checksum does not include the pseudoheader, a user datagram may arrive safe and sound. However, if the IP header is corrupted, it may be delivered to the wrong host.

The protocol field is added to ensure that the packet belongs to UDP, and not to TCP.
The value of the protocol field for UDP is 17.

If this value is changed during transmission, the checksum calculation at the receiver will detect it and UDP drops the packet. It is not delivered to the wrong protocol.

Pseudoheader for checksum calculation



- Encapsulation and Decapsulation

  To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages

- Queuing

  At the client site, when a process starts, it requests a port number from the operating system. Some implementations create both an incoming and an outgoing queue associated with each process.

- Multiplexing and Demultiplexing

  In a host running a TCP/IP protocol suite, there is only one UDP but possibly several processes that may want to use the services of UDP. To handle this situation, UDP multiplexes and demultiplexes

- Comparison : UDP and Simple Protocol

  The only difference is that UDP provides an optional checksum to detect corrupted packets at the receiver site.

Example 3.12

What value is sent for the checksum in one of the following hypothetical situations?

a. The sender decides not to include the checksum.

b. The sender decides to include the checksum, but the value of the sum is all 1s.

c. The sender decides to include the checksum, but the value of the sum is all 0s.

Example 3.12 (continued)

Solution

a. The value sent for the checksum field is all 0s to show that the checksum is not calculated.

b. When the sender complements the sum, the result is all 0s; the sender complements the result again before sending. The value sent for the checksum is all 1s. The second complement operation is needed to avoid confusion with the case in part a.

c. This situation never happens because it implies that the value of every term included in the calculation of the sum is all 0s, which is impossible; some fields in the pseudoheader have nonzero values.

UDP Applications

Although UDP meets almost none of the criteria we mentioned earlier for a reliable transport-layer protocol, UDP is preferable for some applications. The reason is that some services may have some side effects that are either unacceptable or not preferable. An application designer sometimes needs to compromise to get the optimum.

- UDP Features
  - Connectionless Service
  - Lack of Error Control
  - Lack of Congestion Control

A client-server application such as DNS uses the services of UDP because a client needs to send a short request to a server and to receive a quick response from it. The request and response can each fit in one user datagram. Since only one message is exchanged in each direction, the connectionless feature is not an issue; the client or server does not worry that messages are delivered out of order.

A client-server application such as SMTP, which is used in electronic mail, cannot use the services of UDP because a user can send a long e-mail message, which may include multimedia (images, audio, or video). If the application uses UDP and the message does not fit in one single user datagram, the message must be split by the application into different user datagrams. Here the connectionless service may create problems. The user datagrams may arrive and be delivered to the receiver application out of order. The receiver application may not be able to reorder the pieces. This means the connectionless service has a disadvantage for an application program that sends long messages.

Assume we are downloading a very large text file from the Internet. We definitely need to use a transport layer that provides reliable service. We don’t want part of the file to be missing or corrupted when we open the file. The delay created between the deliveries of the parts is not an overriding concern for us; we wait until the whole file is composed before looking at it. In this case, UDP is not a suitable transport layer.

Assume we are using a real-time interactive application, such as Skype. Audio and video are divided into frames and sent one after another. If the transport layer is supposed to resend a corrupted or lost frame, the synchronizing of the whole transmission may be lost. The viewer suddenly sees a blank screen and needs to wait until the second transmission arrives. This is not tolerable. However, if each small part of the screen is sent using one single user datagram, the receiving UDP can easily ignore the corrupted or lost packet and deliver the rest to the application program. That part of the screen is blank for a very short period of time, which most viewers do not even notice.

Typical Applications of UDP

- UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control. It is not usually used for a process such as FTP that needs to send bulk data
- UDP is suitable for a process with internal flow- and error-control mechanisms. For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control. It can easily use UDP.
- UDP is a suitable transport protocol for multicasting. Multicasting capability is embedded in the UDP software but not in the TCP software.
- UDP is used for management processes such as SNMP
- UDP is used for some route updating protocols such as Routing Information Protocol (RIP)
- UDP is normally used for interactive real-time applications that cannot tolerate uneven delay between sections of a received message