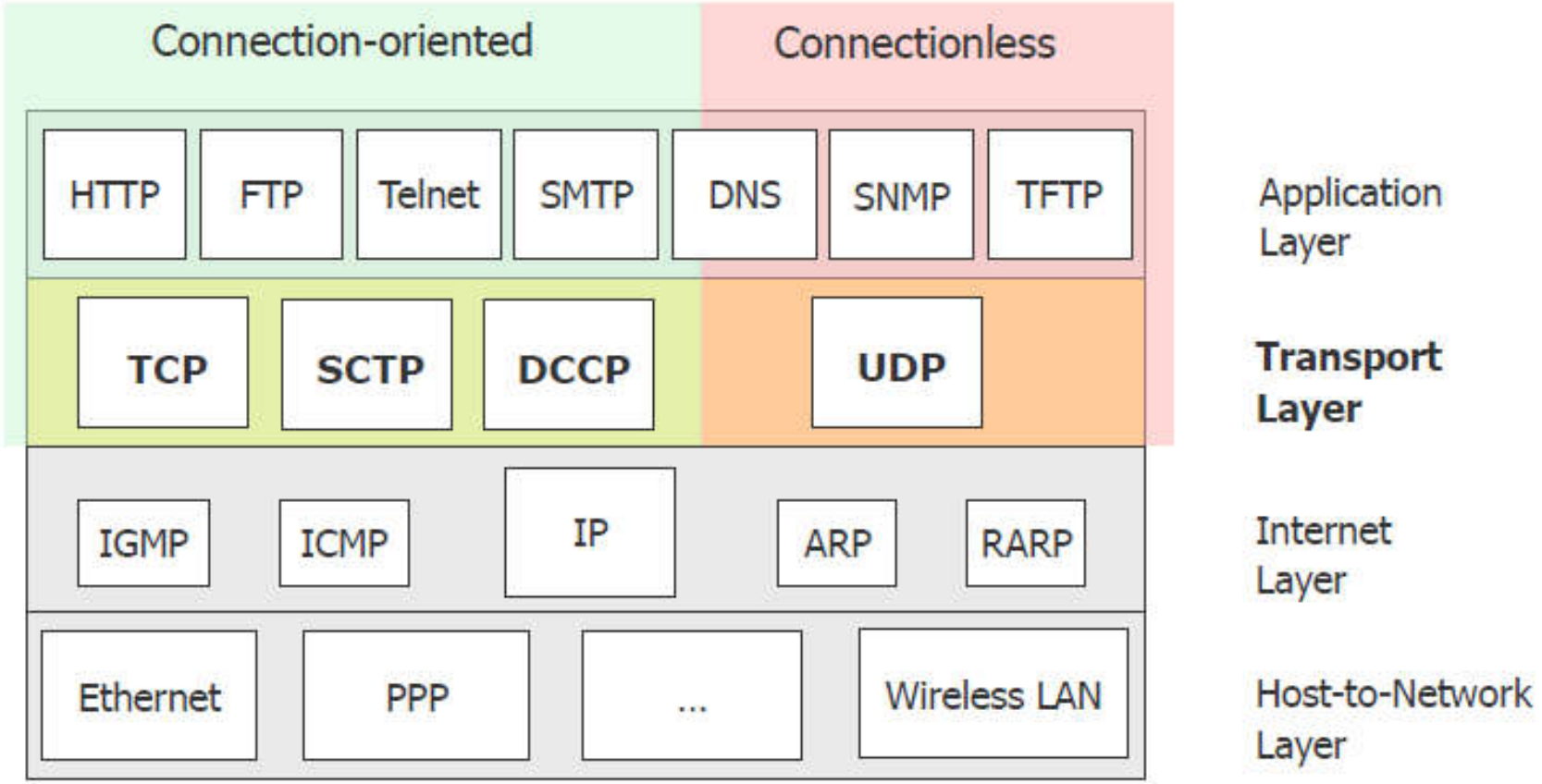
Transport Protocols in the TCP/IP Reference Model

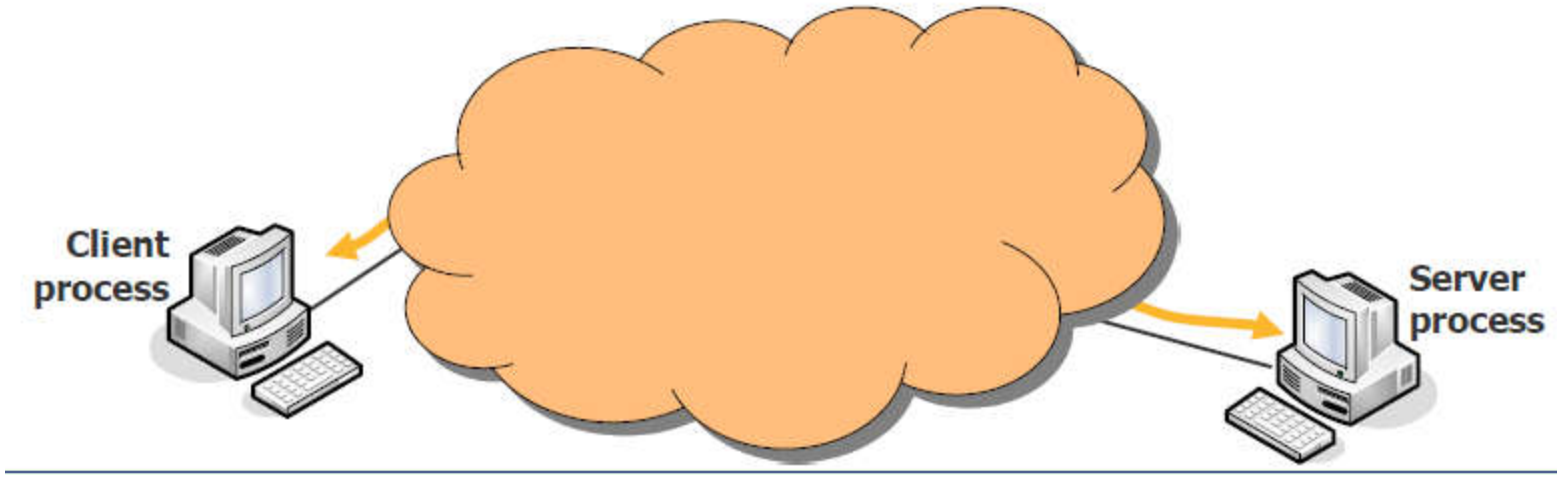
Transport Protocols in the TCP/IP Reference Model



TCP (Transmission Control Protocol): Reliable, connection-oriented
SCTP (Stream Control Transmission Protocol): Reliable, connection-oriented
DCCP (Datagram Congestion Control Protocol): Unrealiable, connection-oriented
UDP (User Datagram Protocol): Unreliable, connectionless

The Classical Transport Layer: TCP and UDP

- Transport protocols are used by the application layer as communication services.
  - They allow the communication between application processes
- TCP is a connection-oriented protocol
- UDP is a connectionless



User Datagram Protocol (UDP)

Introduction

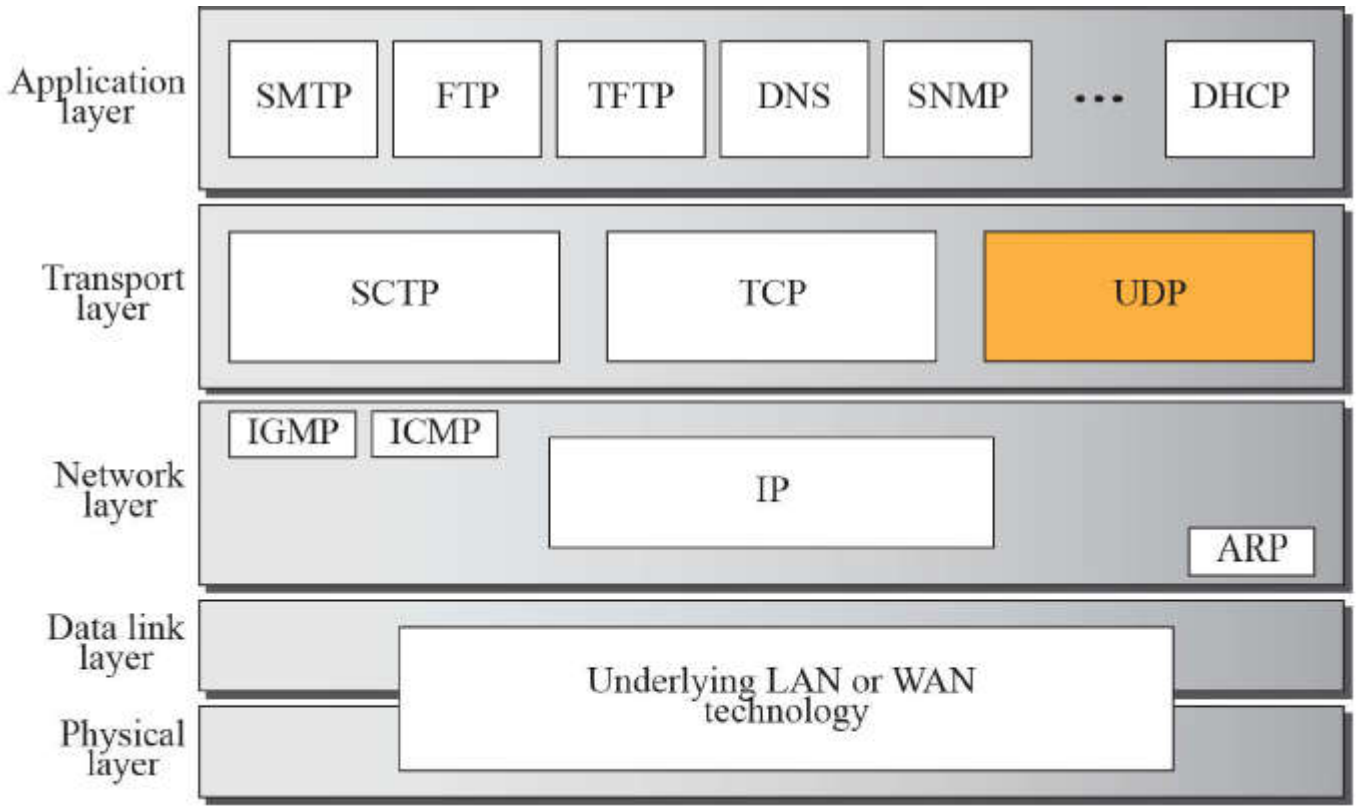
- UDP is *located between the application layer and the IP layer*, and *serves as the intermediary between the application programs and the network operations*.

The User Datagram Protocol (UDP)

- Principle: "**Keep it simple**!"
- 8 byte header
- Like IP: connectionless and unreliable
- Small reliability, but fast exchange of information
- No acknowledgement between communication peers with UDP
- Incorrect packets are simply discarded
- Duplication, sequence order permutation, and packet loss are possible
- The checksum offers the only possibility of testing the packets on transfer errors
- Possible: ACKs and retransmissions are controlled by the application
- Use in multicast (not possible with TCP)
- Why at all UDP?
- Only the addition of a **port to a network address marks communication unique**

(IP Address1, Port1, IP Address2, Port2)

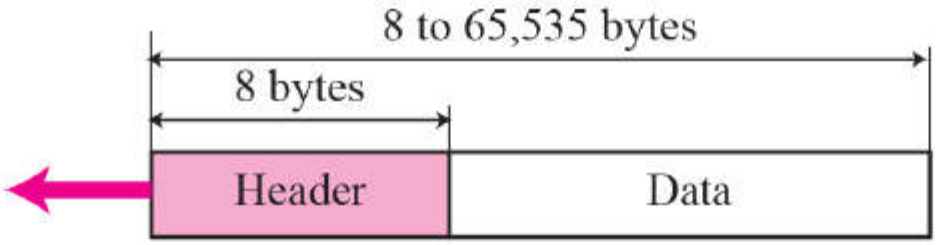
Position of UDP in the TCP/IP Protocol Suite



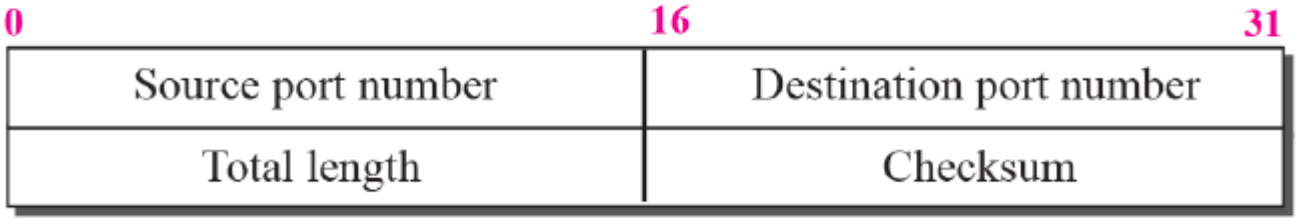
What is meant by a User Datagram?

- UDP packets, called user datagrams, have a fixed size header of 8 bytes.

User Datagram Format



a. UDP user datagram



b. Header format

Example 1 (1/2)

The following is a dump 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?

Example 1 (2/2)

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.

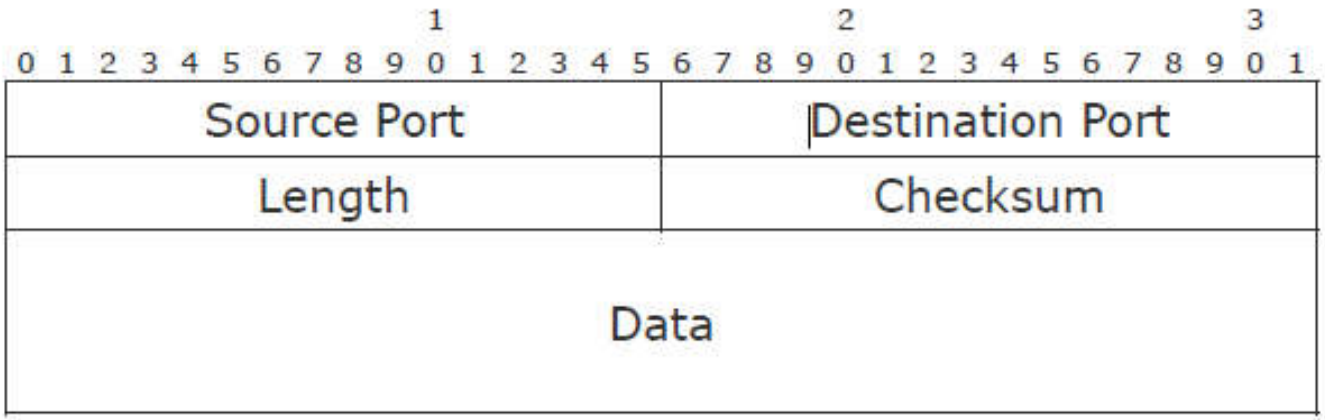
c. The third four hexadecimal digits $(001C)_{16}$ define the length of the whole UDP packet as 28 bytes.

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

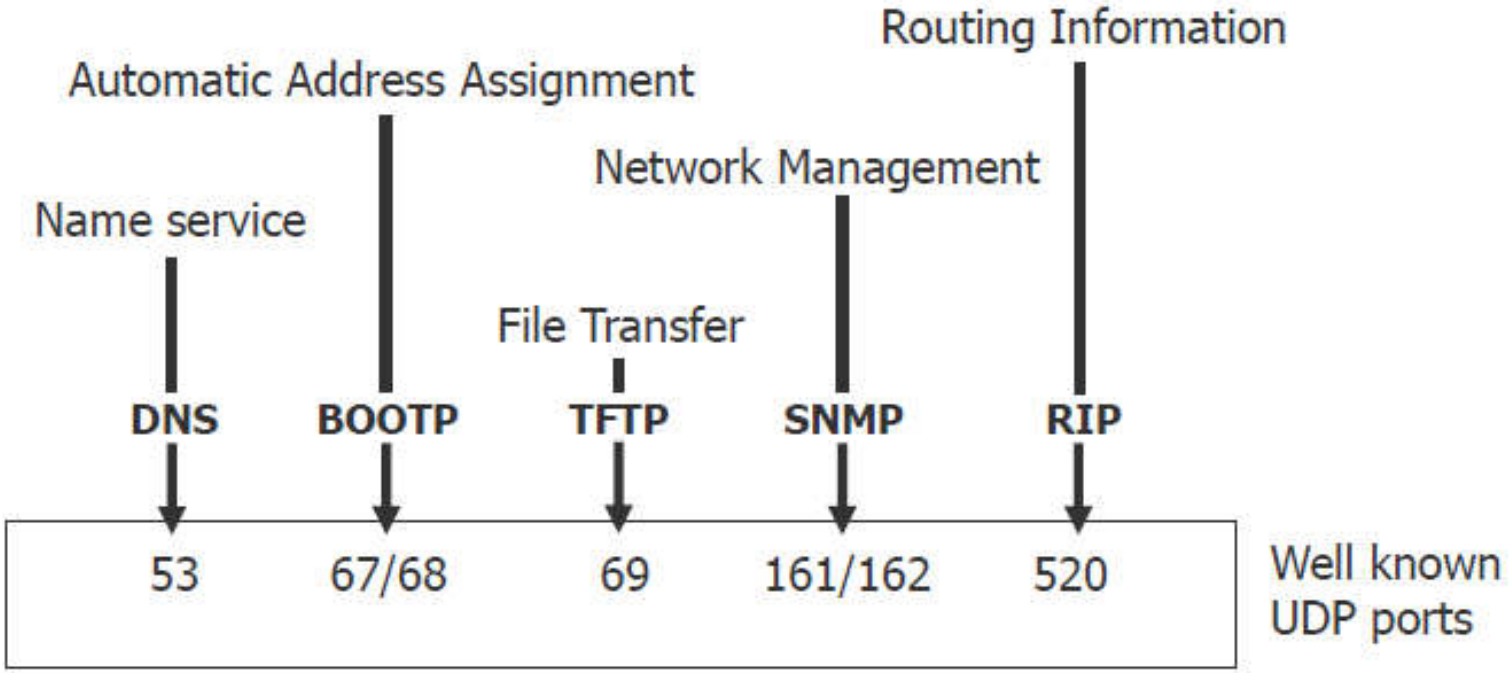
e. Since the destination port number is 13 (well-known port), the packet is from the client to the server.

UDP Header

- **Source Port, Destination Port:** Addressing of the applications by port numbers
- **Length:** The total length of the datagram (header + data) in 32-bit words
- **Checksum** (optional)**:** IP does not have a checksum for the data part, therefore it can be a meaningful addition here
  - The same procedure as in TCP
- **Data:** The payload, it is filled up if necessary to an even byte number, since message length counts in 32-bit words



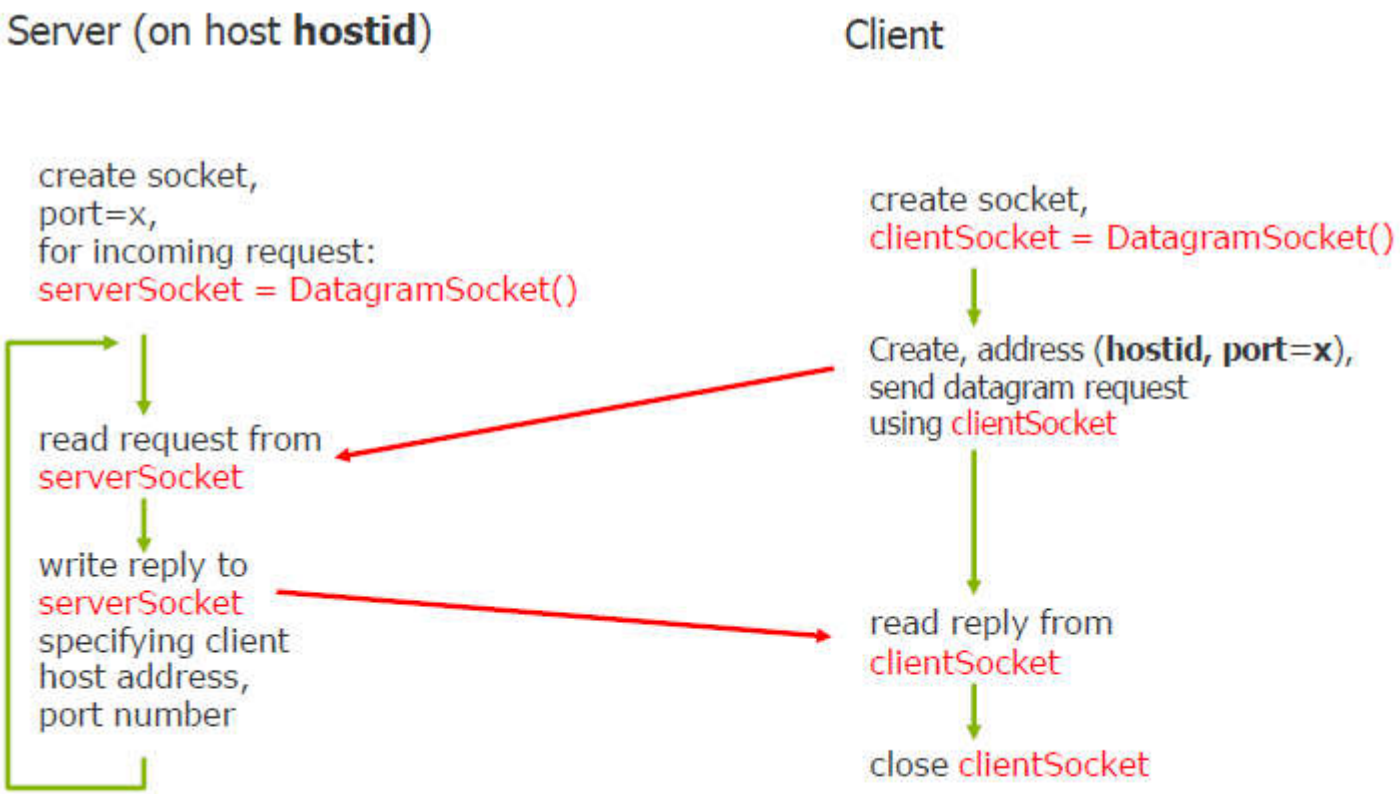
UDP-based Applications



- Port number is 16-bit address, currently three different ranges
  - Ports in the range 0-1023 are **Well-Known** (a.k.a. "system")
  - Ports in the range 1024-49151 are **Registered** (a.k.a. "user")
  - Ports in the range 49152-65535 are **Dynamic/Private**
  - See for more information: http://www.iana.org/assignments/port-numbers

Socket Programming with UDP

Socket Programming with UDP



Example: Java Client (UDP)

```
import java.io.*;
import java.net.*;

class UDPClient {
    public static void main(String arg []) throws Exception
    {
        BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getByName( "hostname" );
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes ();
        DatagramPacket send_pack = new DatagramPacket(sendData, sendData.length,
                                                      IPAddress, 9876);

        clientSocket.send(send_pack);
        DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
        clientSocket.receive(receivePacket);
        String modifiedSentence = new String(receivePacket.getData());
        System.out.println( "FROM SERVER:" + modifiedSentence);
        clientSocket.close();
    }
}
```

Example: Java Server (TCP)

```
import java.io.*;
import java.net.*;

class TCPServer {
    public static void main(string arg []) throws Exception
    {
        String clientSentence;
        String capitalizedSentence;
        ServerSocket welcomeSocket = new ServerSocket(6789);

        while (true) {
            Socket connectionSocket = welcomeSocket.accept();
            BufferedReader inFromClient =
                new BufferedReader(new
                    InputStreamReader(connectionSocket.getInputStream()));

            DataOutputStream outToClient = new
                DataOutputStream(connectionSocket.getOutputStream());

            clientSentence = inFromClient.readLine();
            capitalizedSentence = clientSentence.toUpperCase() + "₩ n";
            outToClient.writeBytes(capitalizedSentence);
        }
        welcomeSocket.close();
    }
}
```

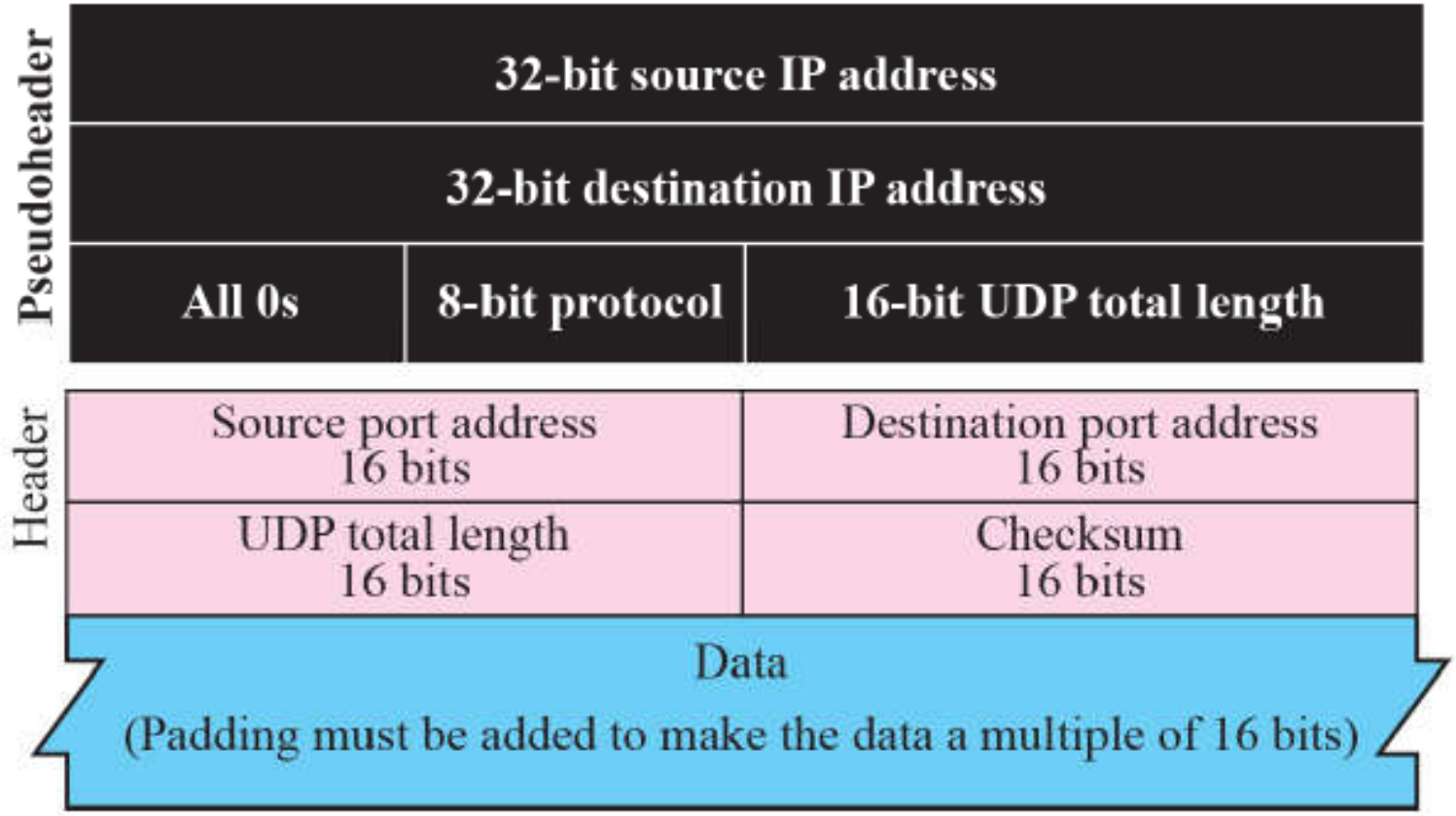
UDP Services

- Process-to-Process Communication
- Connectionless Service
- Flow Control
- Error Control
- Congestion Control
- Encapsulation and Decapsulation
- Queuing
- Multiplexing and Demultiplexing
- Comparison between UDP and Generic Simple Protocol

Well-known Ports used with UDP

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
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
53	Domain	Domain Name Service (DNS)
67	Bootps	Server port to download bootstrap information
68	Bootpc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

Pseudoheader for Checksum Calculation



Example 2: Checksum Calculation for a Simple UDP User Datagram (1/2)

- The example shows the checksum calculation for a very small user datagram with only **7 bytes of data**.
- Because the number of bytes of data is odd, padding is added for checksum calculation.
- The pseudoheader as well as the padding will be dropped when the user datagram is delivered to IP.

Example 2: Checksum Calculation for a Simple UDP User Datagram (2/2)

10011001	00010010	153.18
00001000	01101001	8.105
10101011	00000010	171.2
00001110	00001010	14.10
00000000	00010001	0 and 17
00000000	00001111	15
00000100	00111111	1087
00000000	00001101	13
00000000	00001111	15
00000000	00000000	0 (checksum)
01010100	01000101	T and E
01010011	01010100	S and T
01001001	01001110	I and N
01000111	00000000	G and 0 (padding)
10010110	11101011	Sum
01101001	**00010100**	Checksum

Example 3 (1/2)

- 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 (2/2)

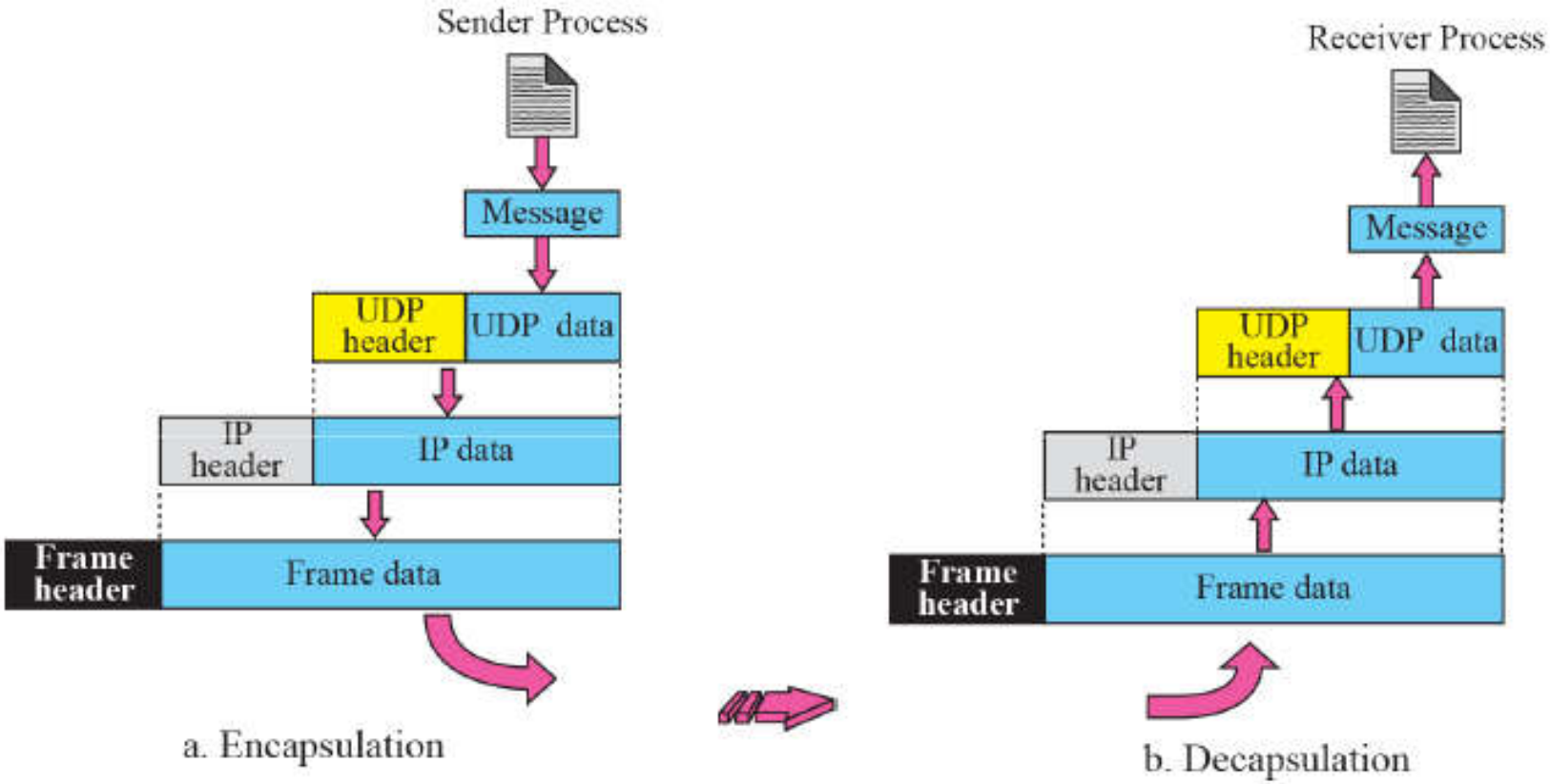
Solution

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

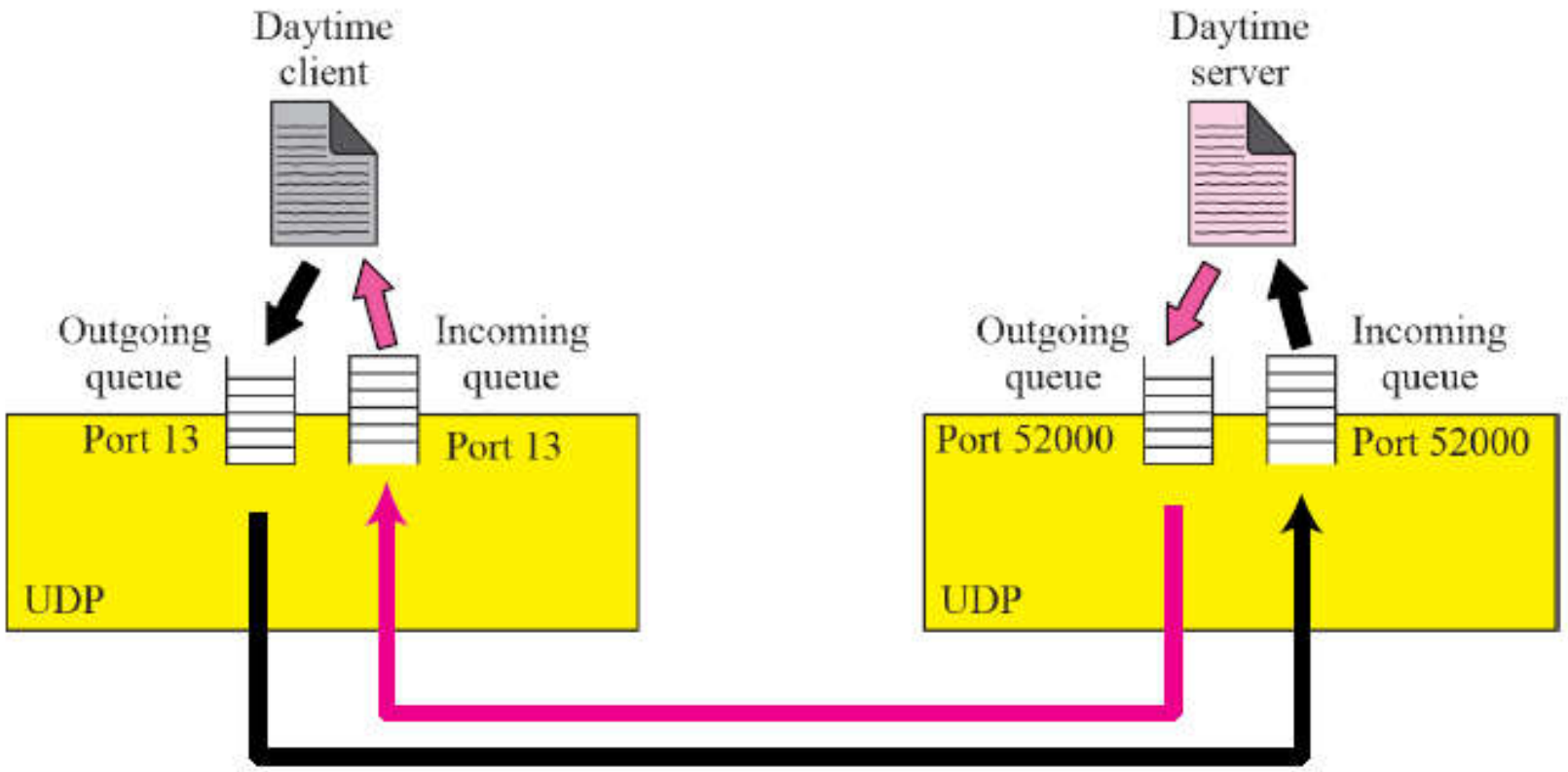
b. When the sender complements the sum, the result is all 0s; the sender complements the result again before sending. The value sent for the checksum is all 1s. The second complement operation is needed to avoid confusion with the case in part a.

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

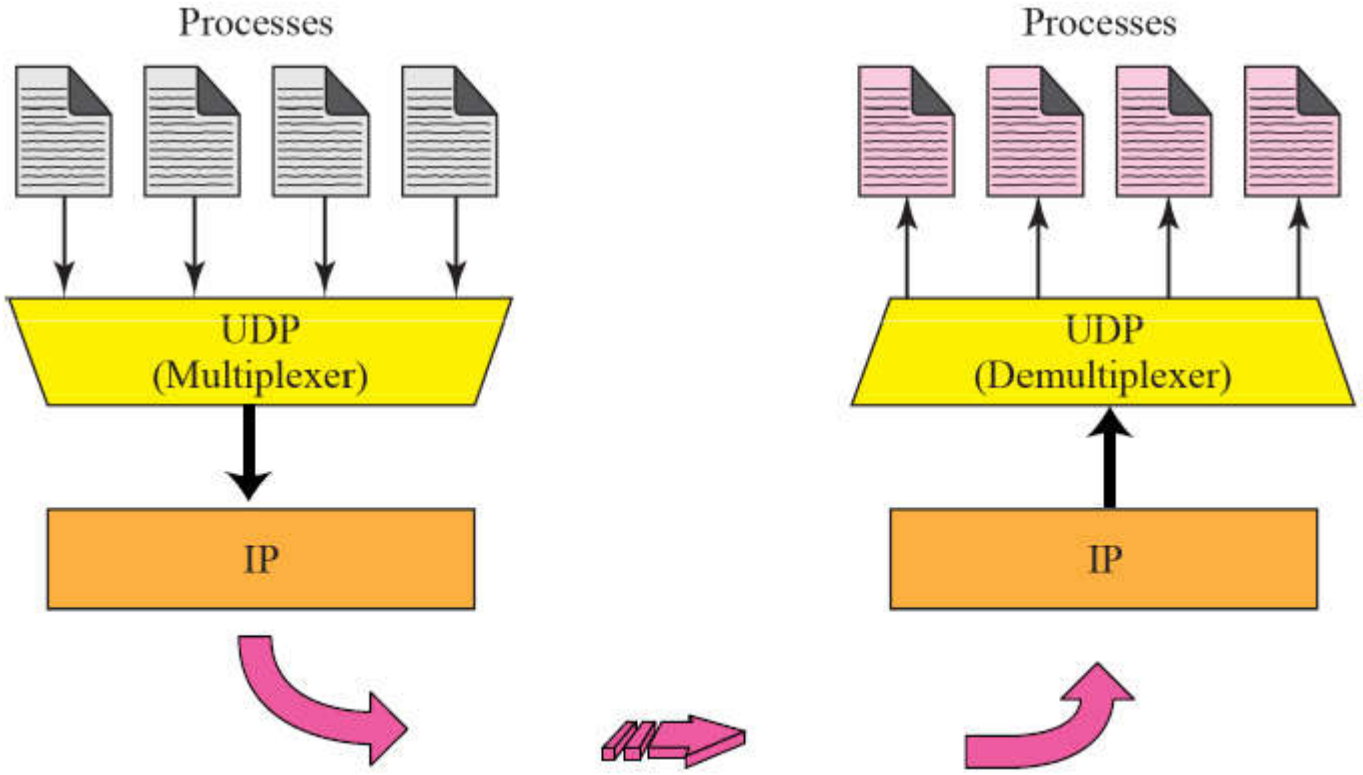
Encapsulation and Decapsulation



Queues in UDP



Multiplexing and Demultiplexing



UDP Applications

Overview

- Although UDP meets almost none of the criteria 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 needs sometimes to compromise to get the optimum.

UDP Application 1 (DNS)

- A client-server application such as Domain Name System (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.

UDP Application 2 (SNMP)

- 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

UDP Application 3 (Downloading)

- 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 delivery of the parts are 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.