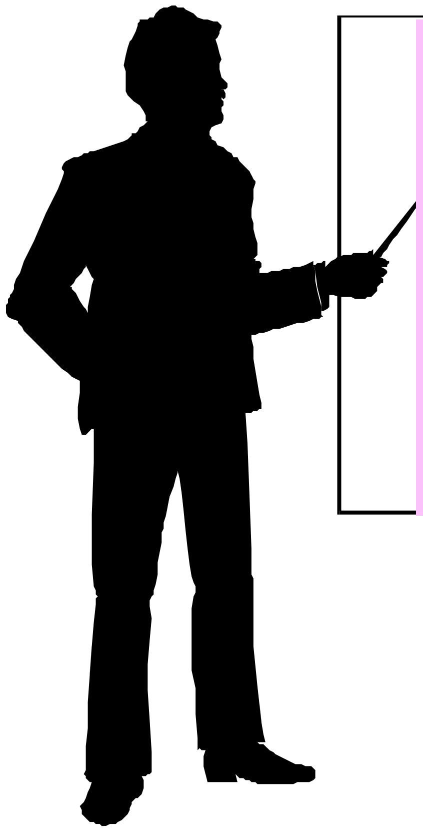


Chapter 14 User Datagram Protocol (UDP)

Kyung Hee University

14.1 Introduction

- Responsibilities of Transport Layer
  - to create a process-to-process communication
    - using port numbers in case of UDP
  - to provide a flow-and-error control mechanism at the transport level
    - But, no flow control mechanism and no acknowledgment for received packets in UDP
    - If UDP detects an error in the received packets, it silently drops it.
  - to provide a connection mechanism for the processes
    - sending streams of data to the transport layer by process
    - making the connection, chopping the stream into transportable units, numbering them and sending them one by one

14.1 Introduction (cont'd)

- Normally, at the receiving end, waiting until all the different units belonging to the transport layer have received, checking, passing those that are error free and delivering them to the receiving process as a stream

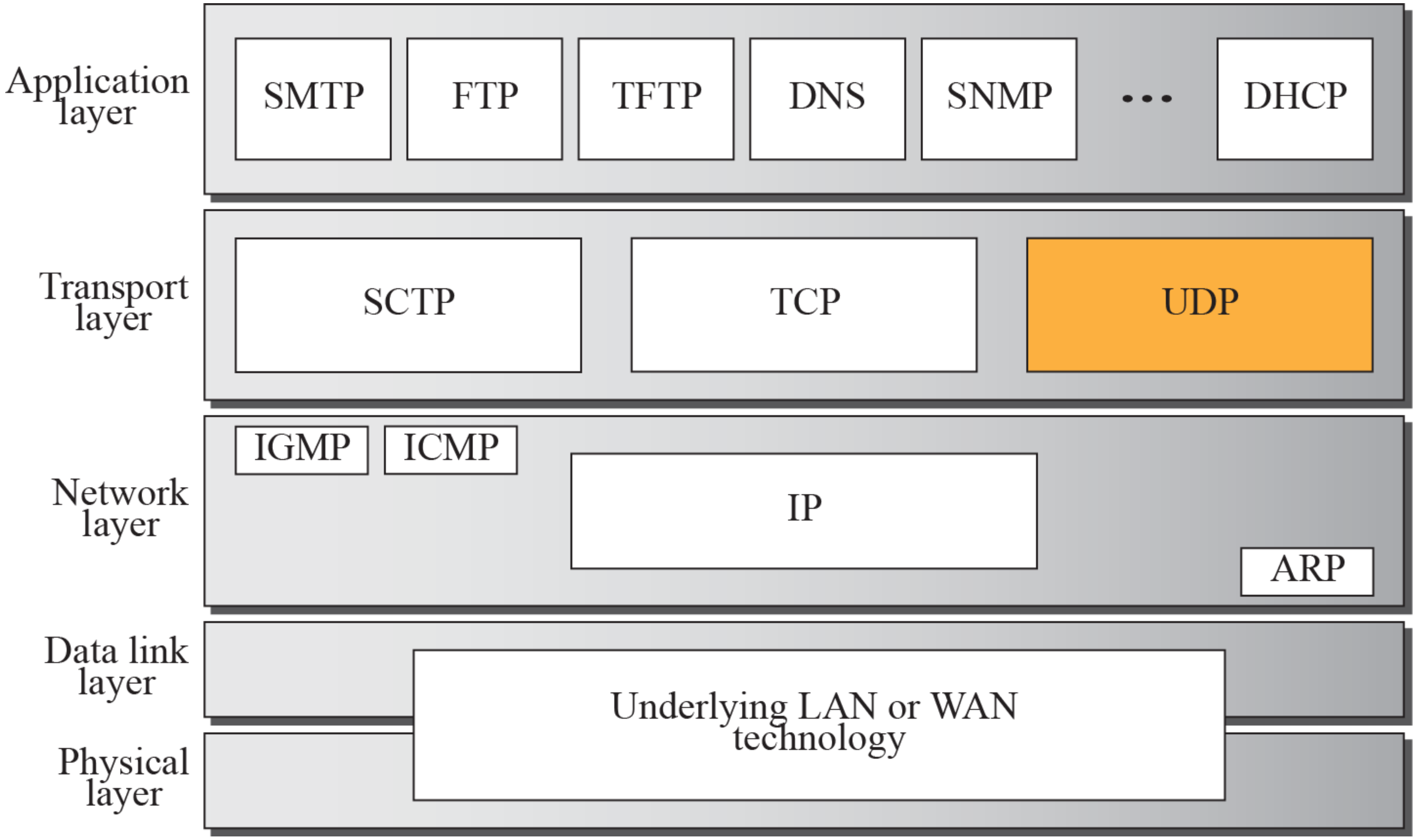
- But, UDP
  - does not do any of the above
    - can only receive a data unit from the process and deliver it, unreliably, to the receiver
  - data unit must be small enough to fit in a UDP packet
- UDP is called a connectionless, unreliable transport protocol
  - providing process-to-process communication instead of host-to-host communication
  - performing very limited error checking

14.1 Introduction (cont'd)

- If UDP is so powerless, why would a process want to use it ?
  - very simple protocol using a minimum of overhead
    - if a process wants to send a small message and does not care much about reliability, it can use UDP
    - if it sends a small message, taking much less interaction between the sender and receiver than it does using TCP

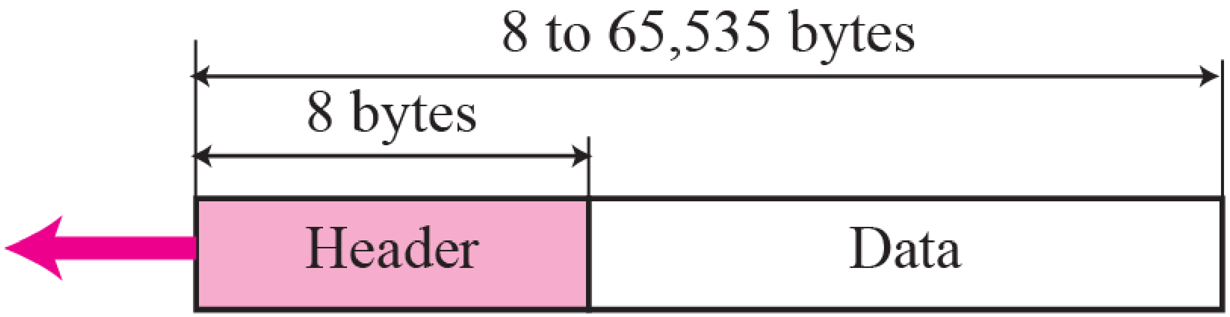
14.1 Introduction (cont'd)

- Position of UDP in the TCP/IP protocol suite

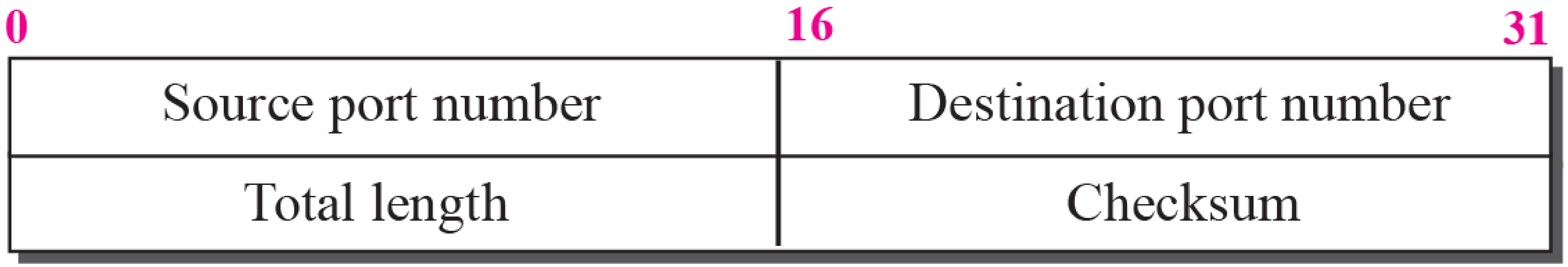


14.2 User Datagram

- User datagram format



a. UDP user datagram



b. Header format

14.2 User Datagram

Source port number

- In case of the client (a client sending a request), having ephemeral port number requested by the process
- In case of the server (a sever sending response), having a well-known port number

Destination port number

- Used by the process running on the destination host

Length

- Defining the total length of the user diagram, header + data
- Minimum 8 bytes. (header + no data)
- the length of data : 0 to 65,507 (65,535 – 20 – 8) bytes
- UDP length = IP Length – IP header's length

Checksum

- used to detect errors over the entire user datagram

Example 14.1

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?

Solution of Example 14.1

a. The source port number is the first four hexadecimal digits ($CB84_{16}$), which means that the source port number is 52100.

b. The destination port number is the second four hexadecimal digits ($000D_{16}$), which means that the destination port number is 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 14.1)

14.3 UDP Service

- Process to Process Communication
- Connectionless Service
  - There is no relationship between the different user datagrams even they are coming from same source process.
- Flow Control
  - The process using UDP should provide flow control
- Error Control
  - Checksum only

Table 14.1 Well-known Ports used with UDP

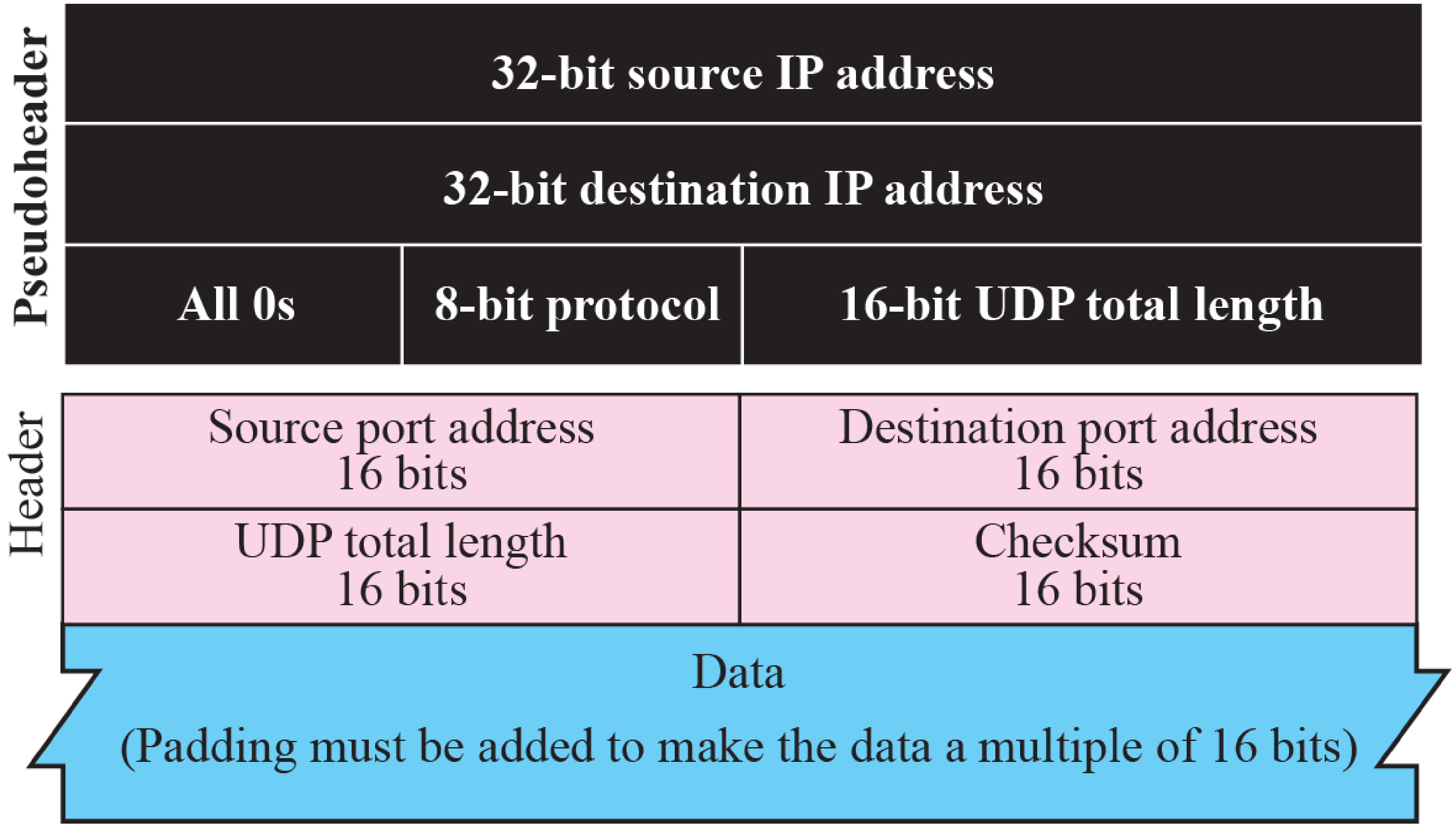
Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Domain	Domain Name Service (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)

Checksum

- UDP checksum calculation is different from the checksum for IP and ICMP
- It includes as follows.
  - pseudoheader : part of the header of the IP packet
  - UDP header
  - data from the application layer

Checksum (cont'd)

- Pseudoheader added to the UDP datagram


Pseudoheader
32-bit source IP address
32-bit destination IP address
All 0s
8-bit protocol
16-bit UDP total length
Header
Source port address 16 bits
Destination port address 16 bits
UDP total length 16 bits
Checksum 16 bits
Data
(Padding must be added to make the data a multiple of 16 bits)

Checksum (cont'd)

- Checksum Calculation at Sender

1. Add the pseudoheader to the UDP datagram
2. Fill the checksum field with zeros
3. Divide the total number of bytes into 16-bit words
4. If the total number of bytes is not even, add one byte of padding (all 0s)
5. Add all 16-bit sections using one's complement arithmetic.
6. Complement the result, and insert it in the checksum field
7. Drop the pseudoheader and added padding
8. Deliver the UDP user datagram to the IP software for encapsulation

Checksum (cont'd)

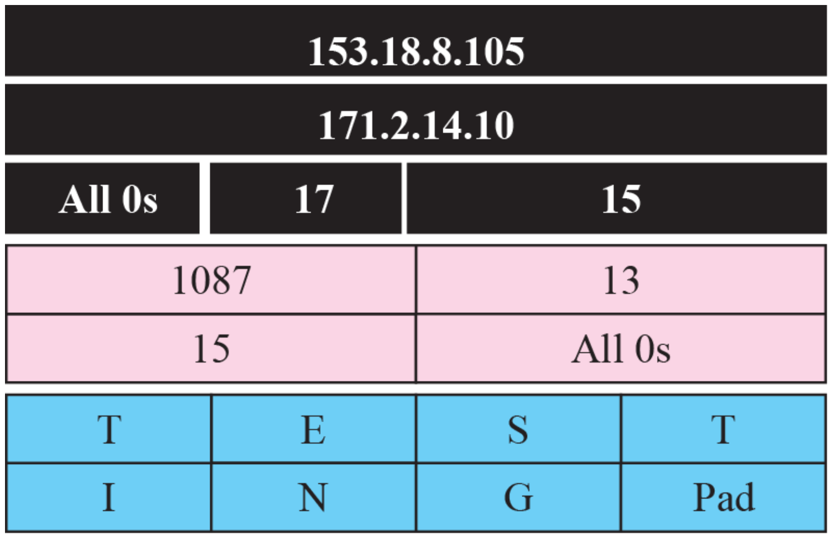
- Checksum Calculation at Receiver

1. Add the pseudoheader to the UDP user datagram
2. Add padding if needed
3. Divide the total bits into 16-bit sections
4. Add all 16-bit sections using one's complement arithmetic
5. Complement the result
6. If the result is all 0s, drop the pseudoheader and any added padding and accept the user datagram.

Example 14.2

- Following figure shows the checksum 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

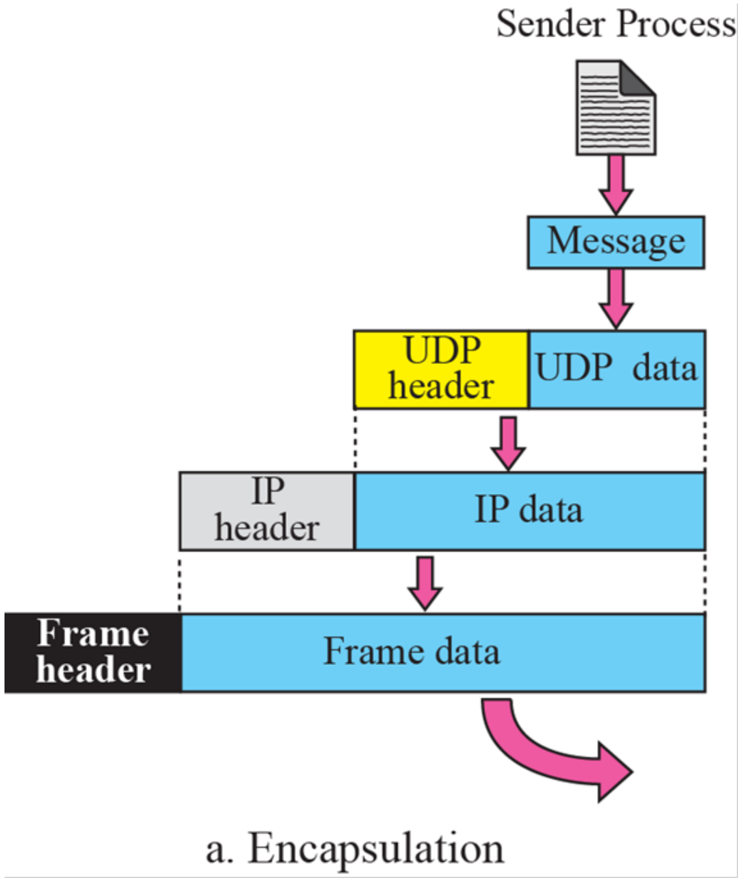
Checksum calculation of a simple UDP user datagram



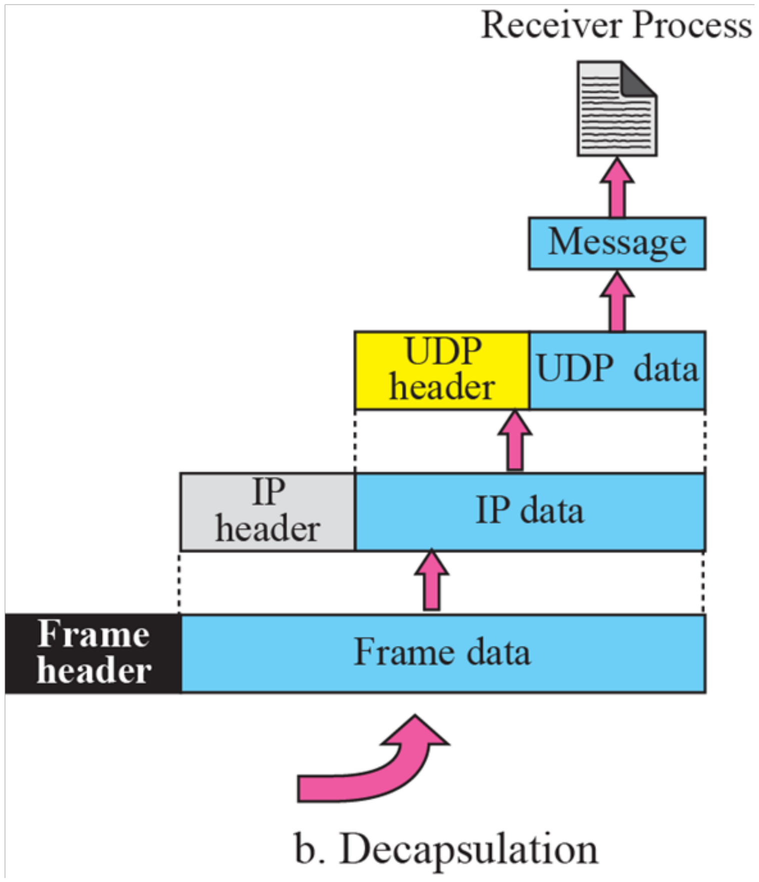
10011001 00010010	$\longrightarrow$	153.18
00001000 01101001	$\longrightarrow$	8.105
10101011 00000010	$\longrightarrow$	171.2
00001110 00001010	$\longrightarrow$	14.10
00000000 00010001	$\longrightarrow$	0 and 17
00000000 00001111	$\longrightarrow$	15
00000100 00111111	$\longrightarrow$	1087
00000000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

UDP Operation

❑ Encapsulation and Decapsulation



a. Encapsulation

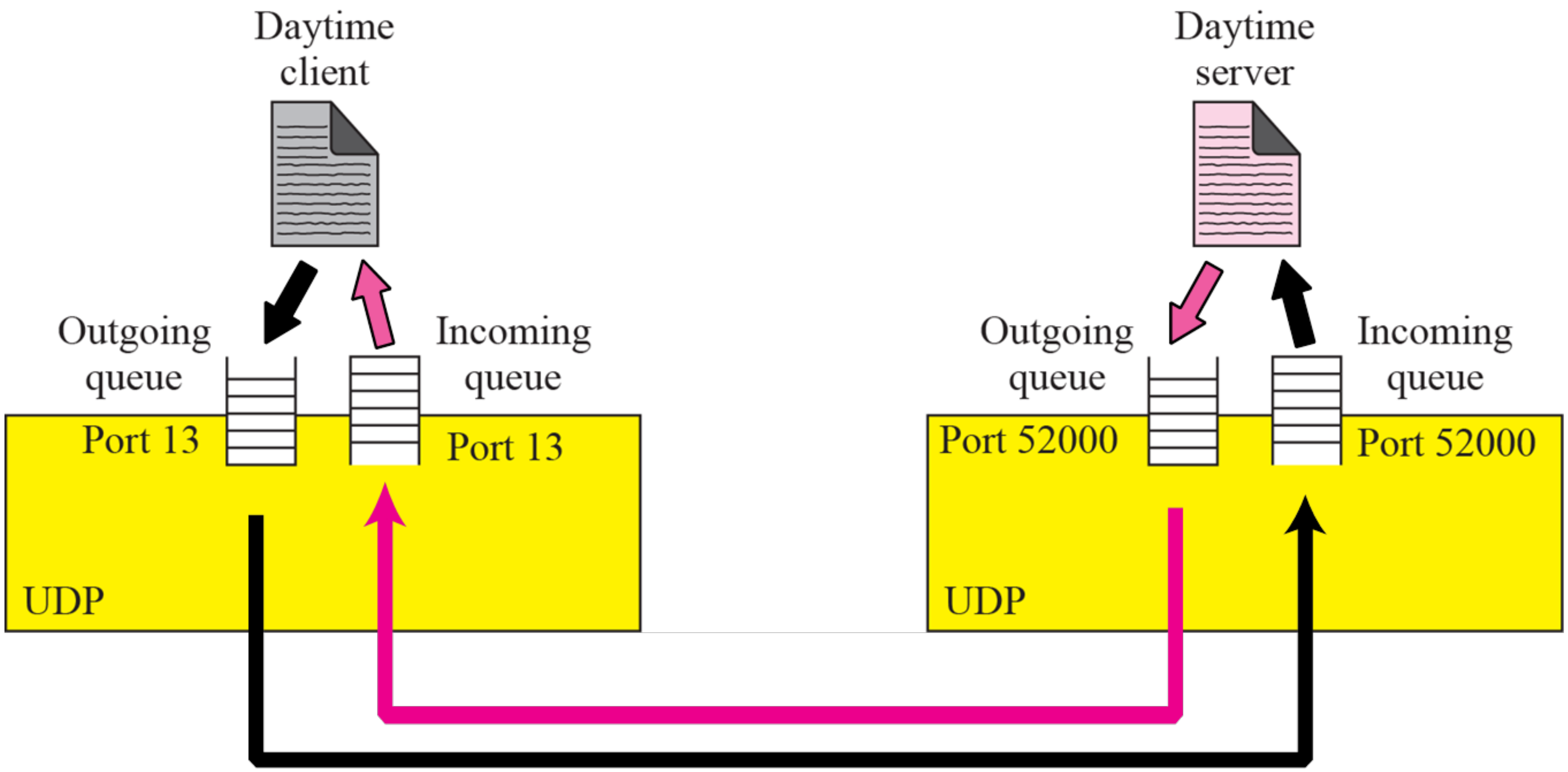


b. Decapsulation

UDP Operation (cont'd)

- Queuing
  - The queues function as long as the process is running.
  - If an outgoing queue is happened overflow, the operating system can ask the client process to wait before sending any more messages.
  - When a message arrives for a client, UDP checks an incoming queue. If there is no such incoming queue, UDP discard the user datagram and ask the ICMP protocols to send a port unreachable message to the server.
  - At the server, the queues remain open as long as the server is running
    - An outgoing queue can overflow. If this happens, the operating system asks the server to wait before sending any more messages

UDP Operation (cont’d)

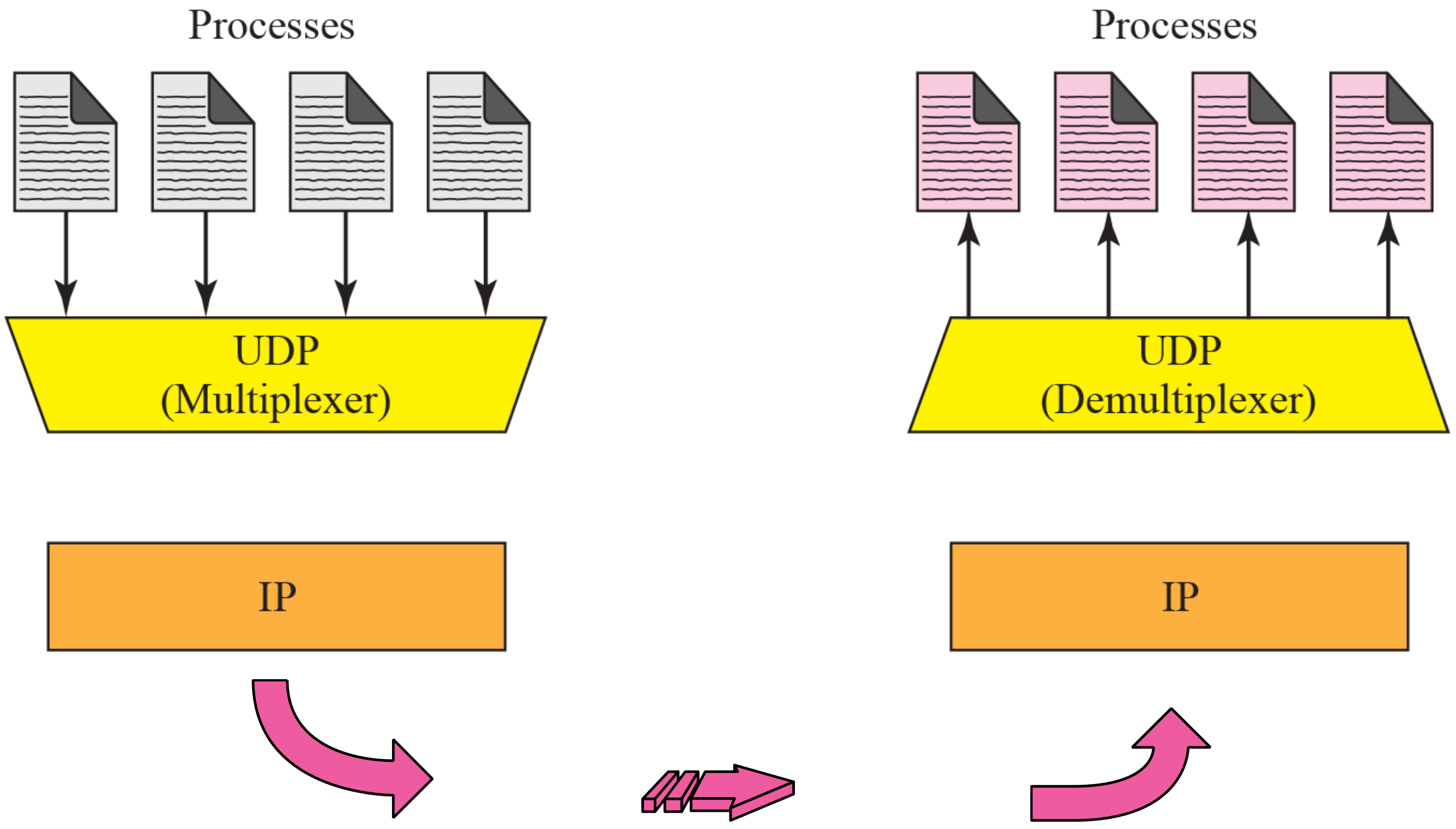


UDP Operation (cont'd)

- Multiplexing
  - At the sender site, there may be several processes that need to send user datagrams
    - differentiating by their assigned port numbers
- Demultiplexing
  - At the receiver site, there is only one UDP
    - UDP receives user datagrams from IP.
    - After error checking and dropping of the header, UDP delivers each message to the appropriate port based on the port numbers

UDP Operation (cont'd)

- Multiplexing and Demultiplexing



14.4 UDP Application

- The following lists some uses of the UDP protocols
  - suitable for a process that requires simple request-response communication and with little concern for flow and error control
    - not used for a process that needs to send bulk data, such as FTP
  - suitable for a process with internal flow and error control mechanisms
    - TFTP process including flow and error control
  - suitable transport protocol for multicasting and broadcasting
    - multicasting and broadcasting capabilities are embedded in the UDP software, but not in the TCP software
- used for management protocol such as SNMP
- used for some route updating protocol such as RIP

Example 14.4

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.

Example 14.5

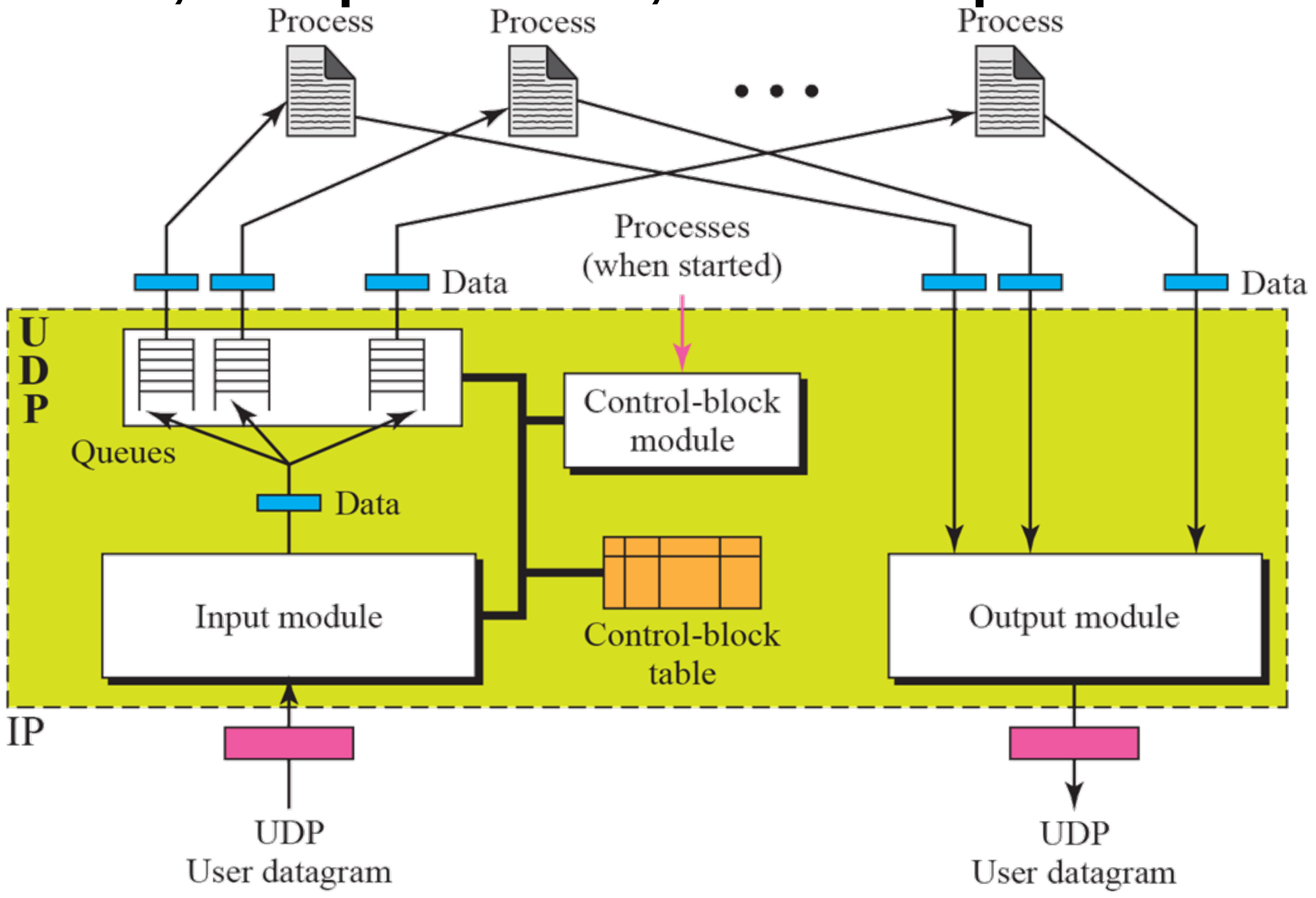
- A client-server application such as SMTP, which is used in electronic mail, cannot use the service of UDP because a user can send a long e-mail message, which may include multimedia (images, audio, or video). If the application uses UDP and the message does not fit in one single user datagram, the message must be split by the application into different user datagrams. Here the connectionless service may create problems. The user datagrams may not be able to reorder the pieces. This means the connectionless service has a disadvantage for an application program that sends long message. In SMTP, when one sends a message, one does not expect to receive a response quickly (sometimes no response is required). This means that the extra delay inherent in connection-oriented service is not crucial for SMTP.

Example 14.6

- 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.

14.5 UDP Package

- Involving 5 components
  - a control-block table, input queues, a control-block module, an input module, and an output module



UDP Package (cont'd)

- Control-Block Table
  - Keeping track of the open ports
  - Each entry having 4 field: the state, which can be FREE or IN-USE, the process ID, the port number and the corresponding queue number
- Input Queues
  - One for each process
- Control-block Module
  - Responsible for management of the control-block table
  - When a process starts, it asks for a port number from the OS
  - OS assigns well-known port numbers to servers and ephemeral port number to clients

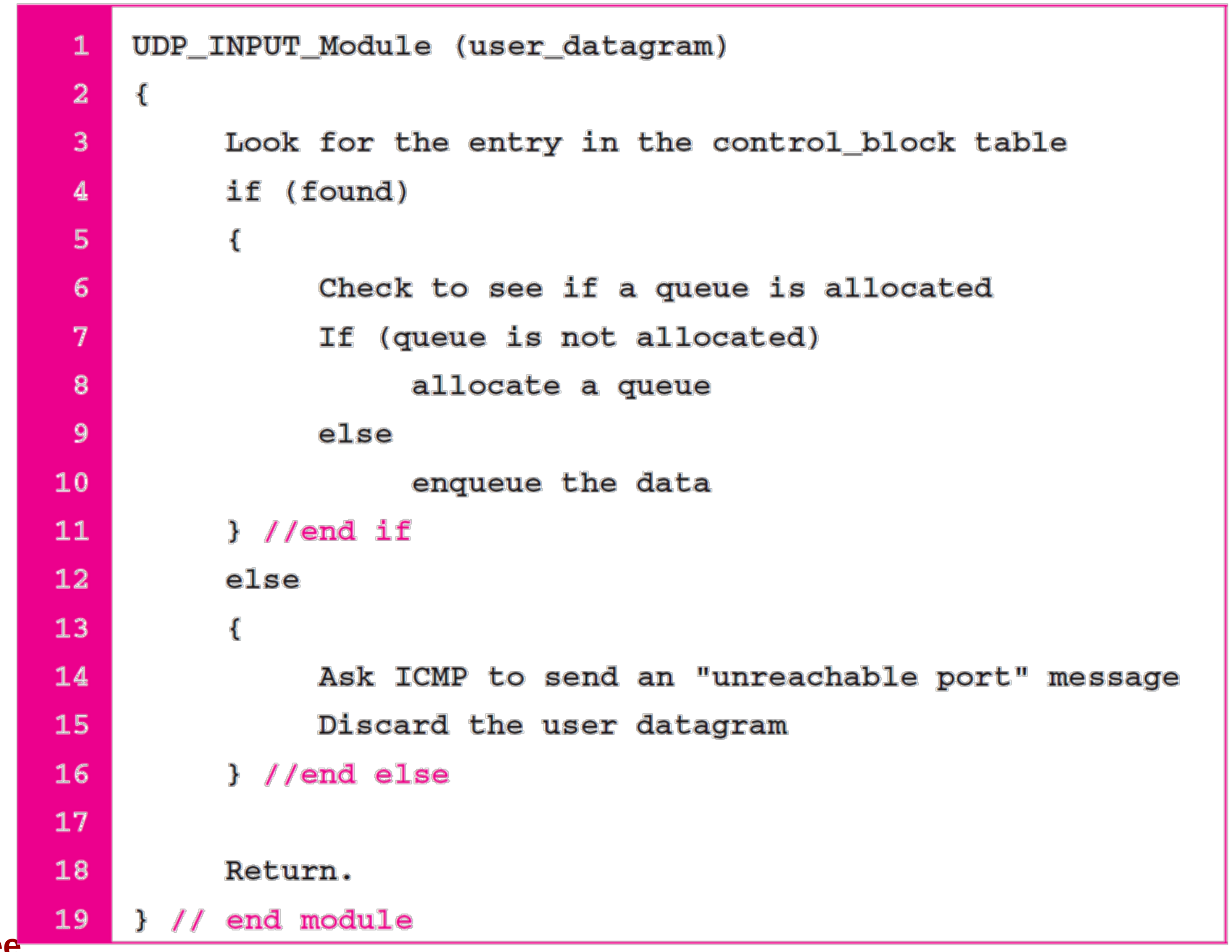
UDP Package (cont'd)

- Control Block Module

```
UDP_Control_Block_Module (process ID, port number)
{
    Search the table for a FREE entry.
    if (not found)
        Delete one entry using a predefined strategy.
    Create a new entry with the state IN-USE
    Enter the process ID and the port number.
    Return.
} // End module
```

UDP Package (cont’d)

- Input Module



```
UDP_INPUT_Module (user_datagram)
{
        Look for the entry in the control_block table
        if (found)
        {
                Check to see if a queue is allocated
                If (queue is not allocated)
                        allocate a queue
                else
                        enqueue the data
        } //end if
        else
        {
                Ask ICMP to send an "unreachable port" message
                Discard the user datagram
        } //end else

        Return.
} // end module
```

UDP Package (cont'd)

- Output Module

```
UDP_OUTPUT_MODULE (Data)
{
    Create a user datagram
    Send the user datagram
    Return.
}
```

UDP Package (cont'd)

- **Examples**
  - **Control block table at the beginning of examples**

State	*Process ID*	*Port Number*	*Queue Number*
IN-USE	2,345	52,010	34
IN-USE	3,422	52,011	
FREE			
IN-USE	4,652	52,012	38
FREE			

Example 14.8

The first activity is the arrival of a user datagram with destination port number 52,012. The input module searches for this port number and finds it. Queue number 38 has been assigned to this port, which means that the port has been previously used. The input module sends the data to queue 38. The control-block table does not change.

Example 14.9

After a few seconds, a process starts. It asks the operating system for a port number and is granted port number 52,014. Now the process sends its ID(4,978) and the port number to the control-block module to create an entry in the table. The module takes the first FREE entry and inserts the information received. The module does not allocate a queue at this moment because no user datagrams have arrived for this destination.

Table 14.6 Control-Block Table after Example 14.9

State	*Process ID*	*Port Number*	*Queue Number*
IN-USE	2,345	52,010	34
IN-USE	3,422	52,011	
IN-USE	4,978	52,014	
IN-USE	4,652	52,012	38
FREE			

Example 14.10

A user datagram now arrives for port 52,011. The input module checks the table and finds that no queue has been allocated for this destination since this is the first time a user datagram has arrived for this destination. The module creates a queue and gives it a number (43).

Control-Block Table after Example 14.10

State	*Process ID*	*Port Number*	*Queue Number*
IN-USE	2,345	52,010	34
IN-USE	3,422	52,011	43
IN-USE	4,978	52,014	
IN-USE	4,652	52,012	38
FREE			

Example 14.11

After a few seconds, a user datagram arrives for port 52,222. The input module checks the table and cannot find an entry for this destination. The user datagram is dropped and a request is made to ICMP to send an unreachable port message to the source.

Summary (1)

- UDP is a transport protocol that creates a process-to-process communication. UDP is a (mostly) unrelible and connectionless protocol that requires little overhead and offers fast delivery. The UDP packet is called a user datagram.

- UDP's only attempt at error control is the checksum. Inclusion of a pseudoheader in the checksum calculation allows source and destination IP address errors to be detected. UDP has no flow-control mechanism.

- A user datagram is encapsulated in the data field of an IP datagram. Incoming and outgoing queue hold message going to and from UDP

Summary (2)

- UDP uses multiplexing to handle outgoing user datagrams from multiple processes on one host. UDP uses demultiplexing to handle incoming user datagrams that go to different processes on the same host.
- A UDP package can involve five components : a control-block table, a control-block module, input queues, an input module, and an output module.