Chapter 11

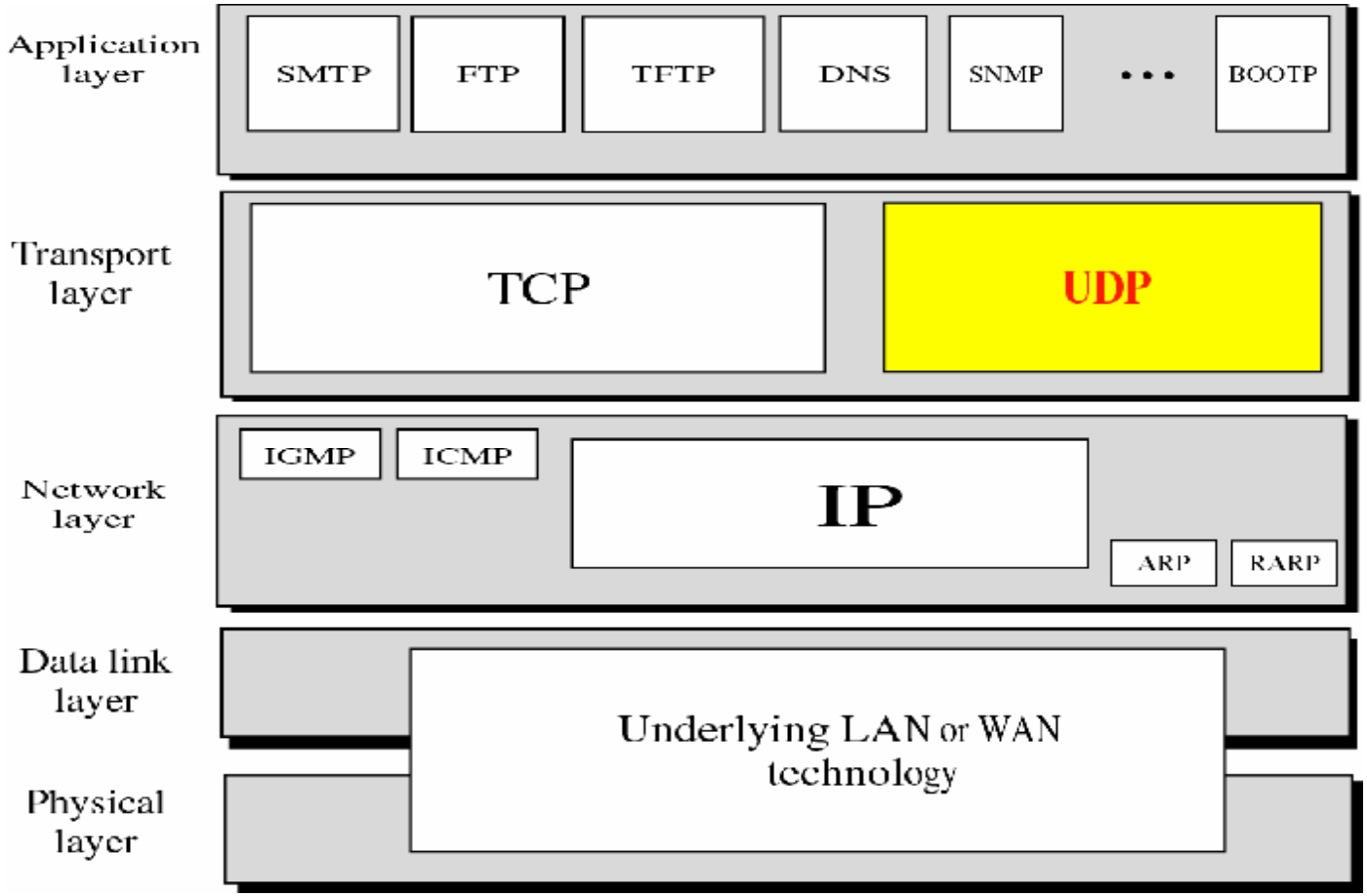
User Datagram Protocol (UDP)

Outline

- Process-to-process communication
- User datagram
- Checksum
- UDP operation
- Use of UDP
- UDP packet

Figure 11.1

Position of UDP in the TCP/IP Protocol Suite



Introduction

- A transport layer protocol usually has two responsibilities
  - Create a *process-to-process* communications
  - Provide *control mechanism* at the transport layer
    - UDP does this task at a very minimal level
    - It only provides error control to some extent

Introduction (Cont.)

- Thus, UDP is a connectionless, unreliable transport protocol
  - Only add process-to-process to the IP
  - And perform very limited error checking

- Advantages
  - UDP is very simple protocol using a minimum of overhead

11.1

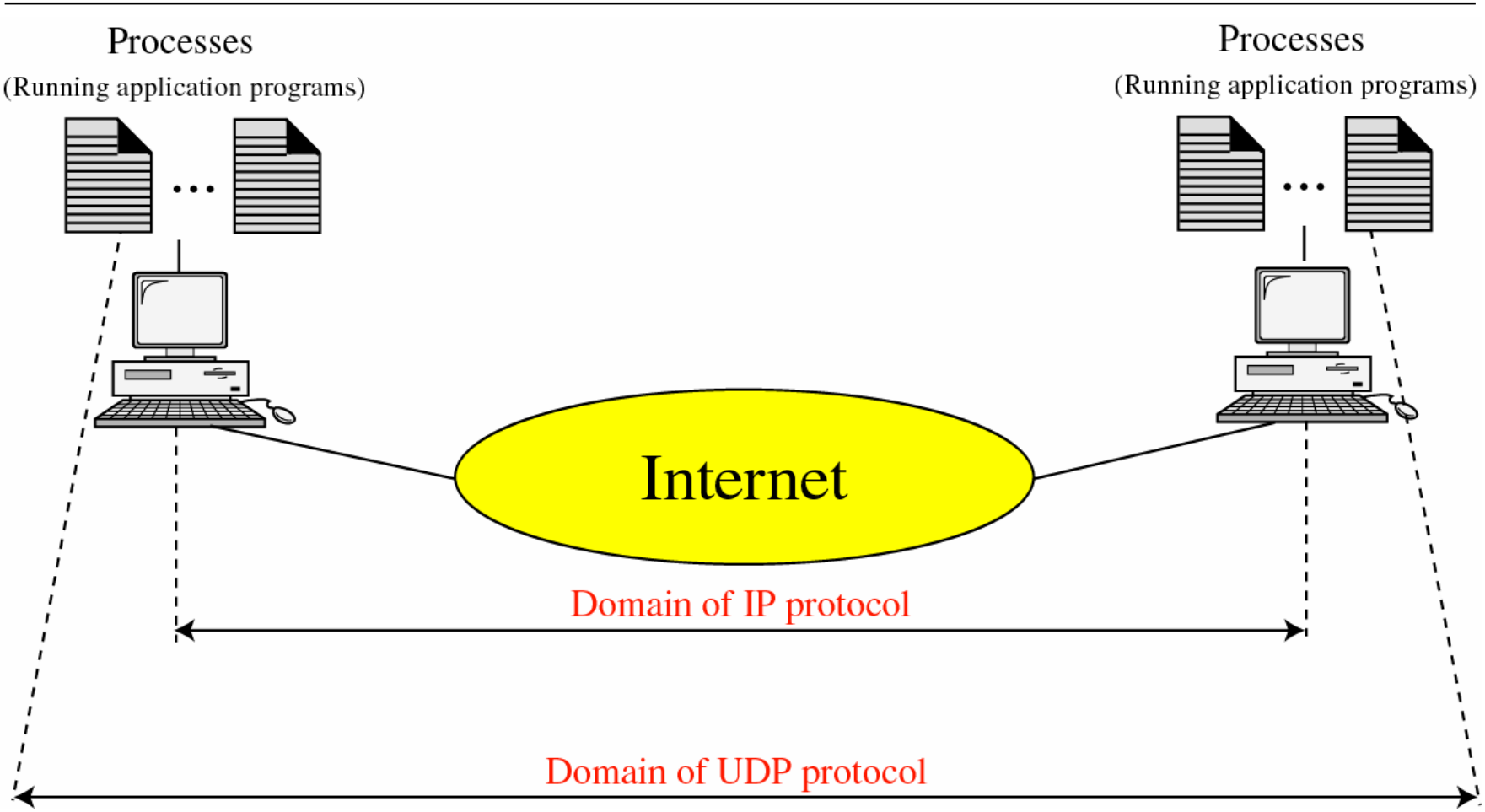
PROCESS TO PROCESS COMMUNICATION

Process-to-Process Communication

- Host-to-host communication
  - Network layer protocol: IP
  - Deliver the message only to the destination computer
- Process-to-process communication
  - Transport layer protocol: UDP and TCP
  - Deliver the message to the appropriate process

Figure 11.2

UDP Versus IP



Port Numbers

- Used to define the process
- 0~65535
- Well-known port numbers
  - Some serves needs to be assign a universal port number
- However, client's port number can be defined randomly
  - *Ephemeral port number*

Port Numbers

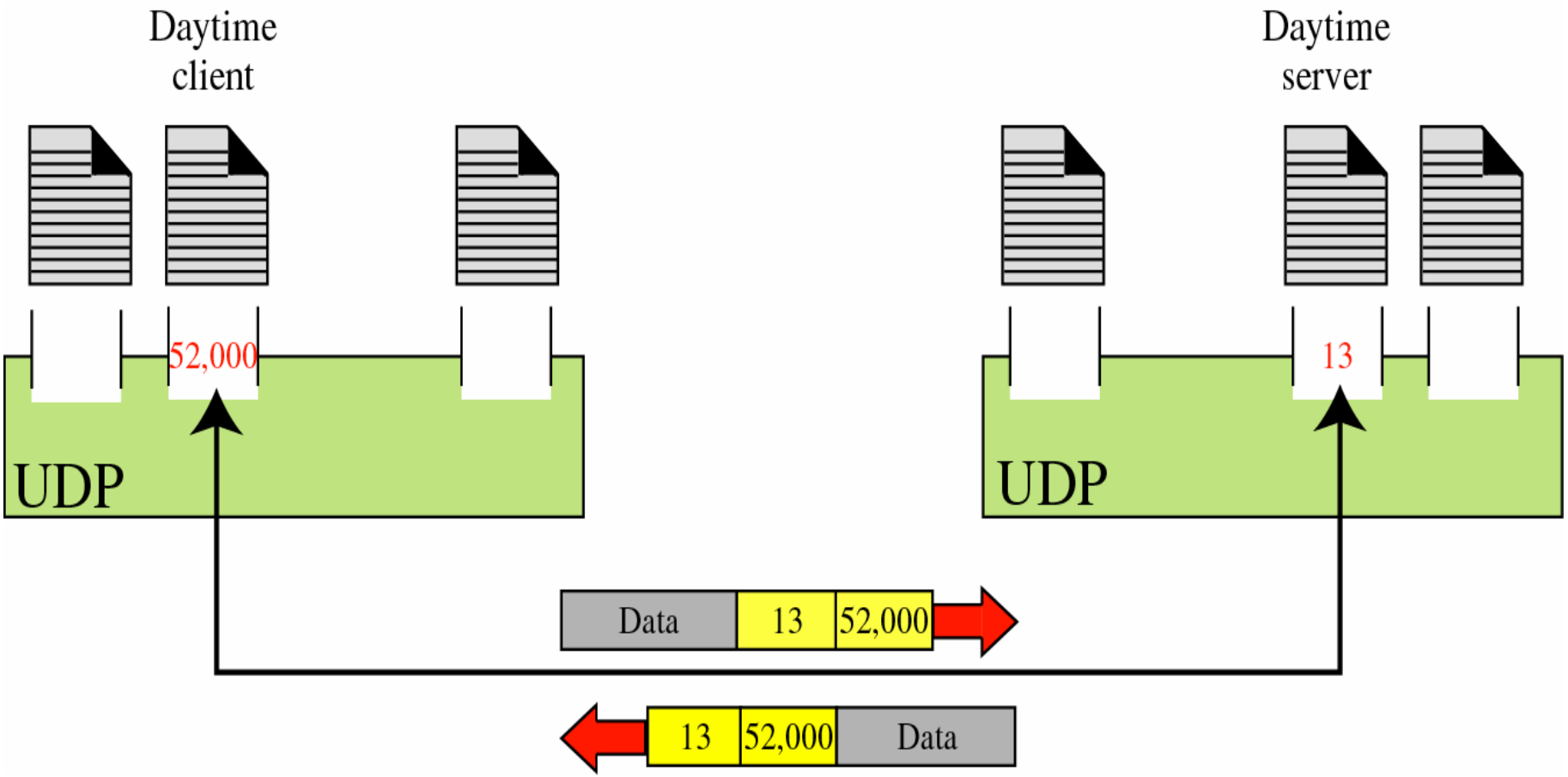
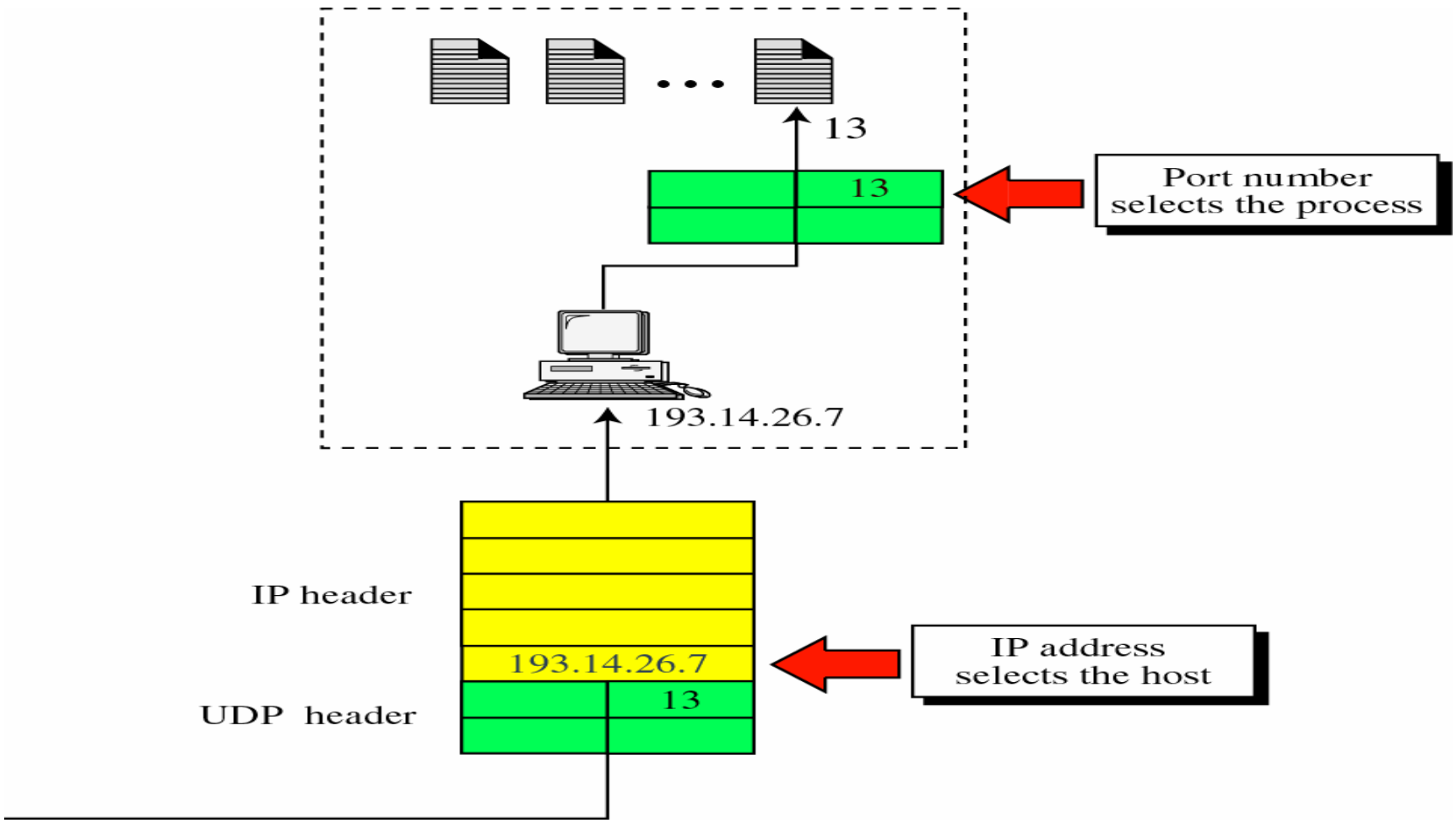


Figure 11.4

IP Addresses Versus Port Numbers

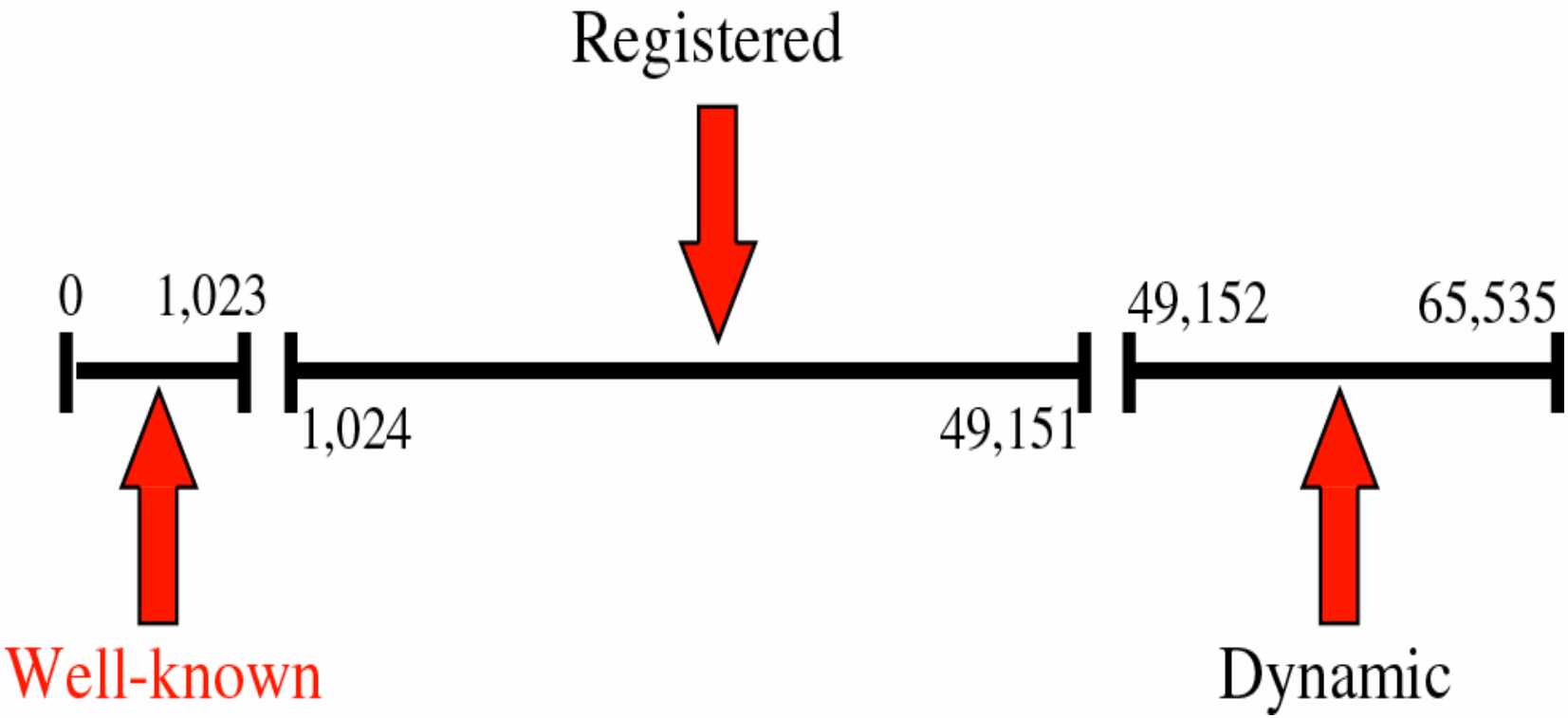


IANA Ranges

- The IANA divides the port numbers into three ranges
  - ***Well-known***: assigned and controlled by IANA
    - 0~1,023
  - ***Registered***: not assigned or controlled by IANA
    - 1,024~49,151
    - Can only be registered with IANA to prevent duplication
  - ***Dynamic*** (or ***private***): neither controlled nor registered
    - 49152~65535
    - Ephemeral ports and can by used by any process

Figure 11-5

IANA Ranges



Well-Known Ports for 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	Return the data and the time
17	Quote	Returns a quote of a day
19	Chargen	Return a string of characters
53	Nameserver	Domain Name Service
67	Bootps	Server port to download bootstrap information
68	Bootpc	Client port to download bootstrap information

Well-Known Ports for UDP (Cont.)

Port	Protocol	Description
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)

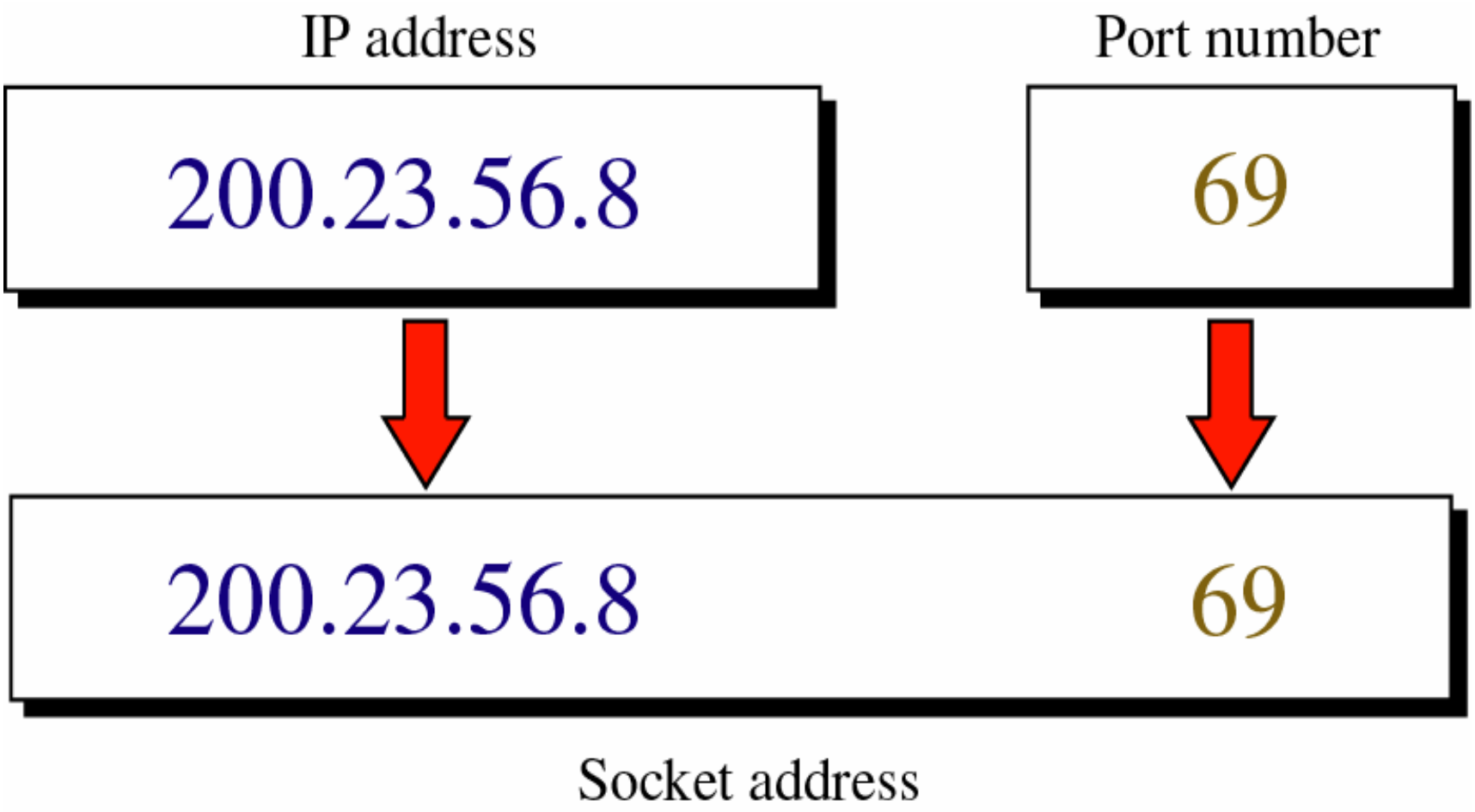
Socket Addresses

- UDP needs two identifiers
  - The IP address
  - The port number

- Socket address
  - The combination of an IP address and a port number

Figure 11.6

Socket Addresses



11.2

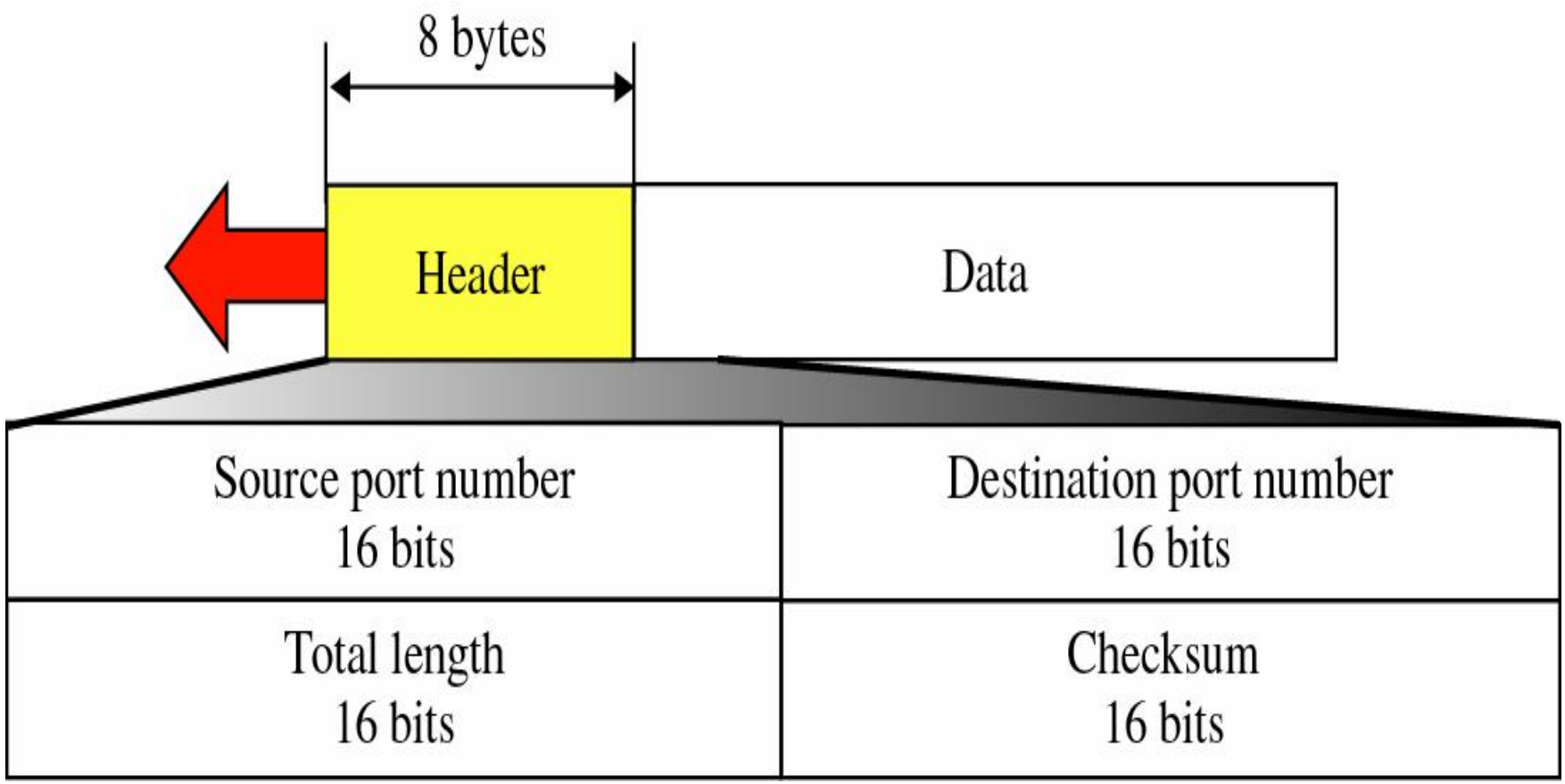
USER DATAGRAM

User Datagram

- UDP packets: called *user datagrams*
  - Have a fixed-size header of 8 bytes
- The fields are
  - Source port number: 16-bits: 0~65535
    - Usually chosen by the UDP software
  - Destination port number: 16-bits: 0~65535
  - Length: 16 bits:
    - Total length (*header plus data*) of the user datagram
    - Minimum length must be 8 bytes (contains only header)
  - Checksum: 16 bits
    - Detect errors over the entire user datagram (*header plus data*)

Figure 11.7

User Datagram Format



User Datagram (Cont.)

- However, the UDP length can also be derived from the IP header
  - UDP length = IP length – IP header's length
- Thus, theoretically, the length field in a UDP datagram is not necessary
  - However, it is more efficient to derive the length of a UDP packet by the UDP package itself
    - Ask the IP layer is time consuming

Note

UDP length =
IP length - IP header's length

11.3

CHECKSUM

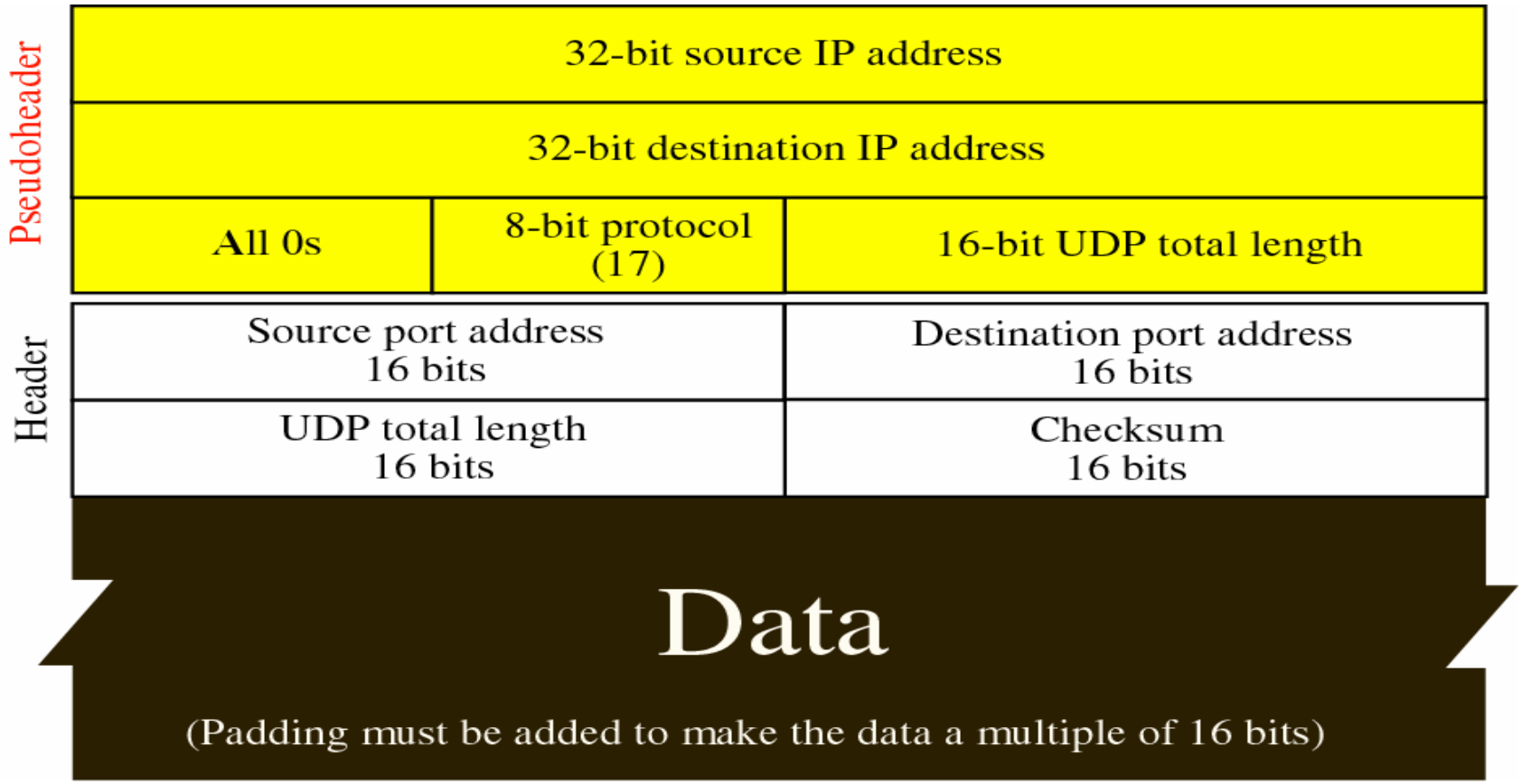
Checksum

- UDP checksum calculation is different from the one for IP and ICMP
- The checksum includes three sections
  - The *Pseudoheader*
    - However, it will not appear in the UDP datagram
  - The UDP header
  - The *data* coming from the application layer

Pseudoheader

- Some part of the header of the IP packet
  - With some fields are filled with 0s
- Fields
  - Source and destination IP address
    - To prevent that UDP datagram is correct but IP header is corrupted and be delivered to the wrong host
  - Protocol
    - To prevent the packet to deliver to the TCP
    - UDP has the value of 17

Pseudoheader Added to the UDP Datagram



Checksum Calculation at Sender

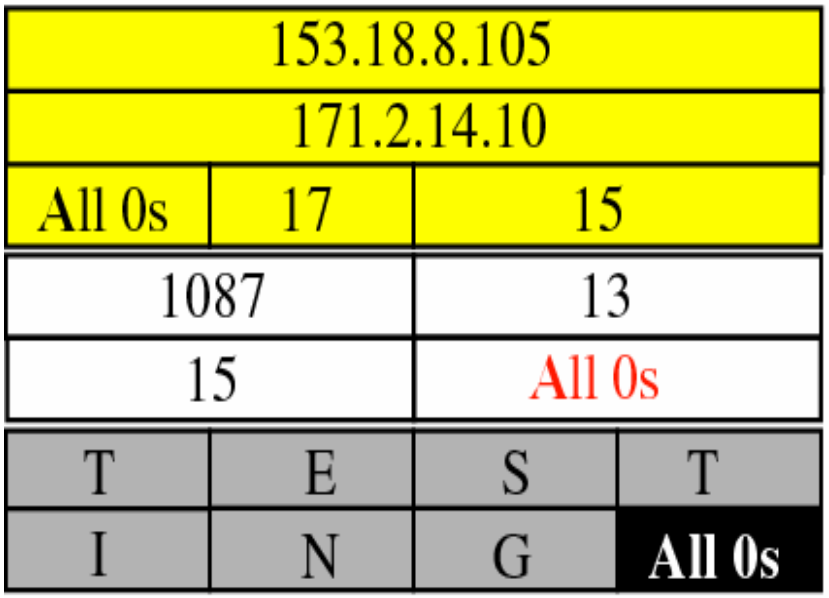
- Add the pseudoheader to the UDP datagram
- Fill the checksum field with zeros
- Divide the total bits into 16-bits words
- If the total number of bytes is not even
  - Add 1 byte of padding (all 0s)
- Add all 16-bit section using one's complement arithmetic
- Complement the result and insert it in the checksum field
- Drop the pesudoheader and any added padding
- Deliver the UDP datagram to the IP software

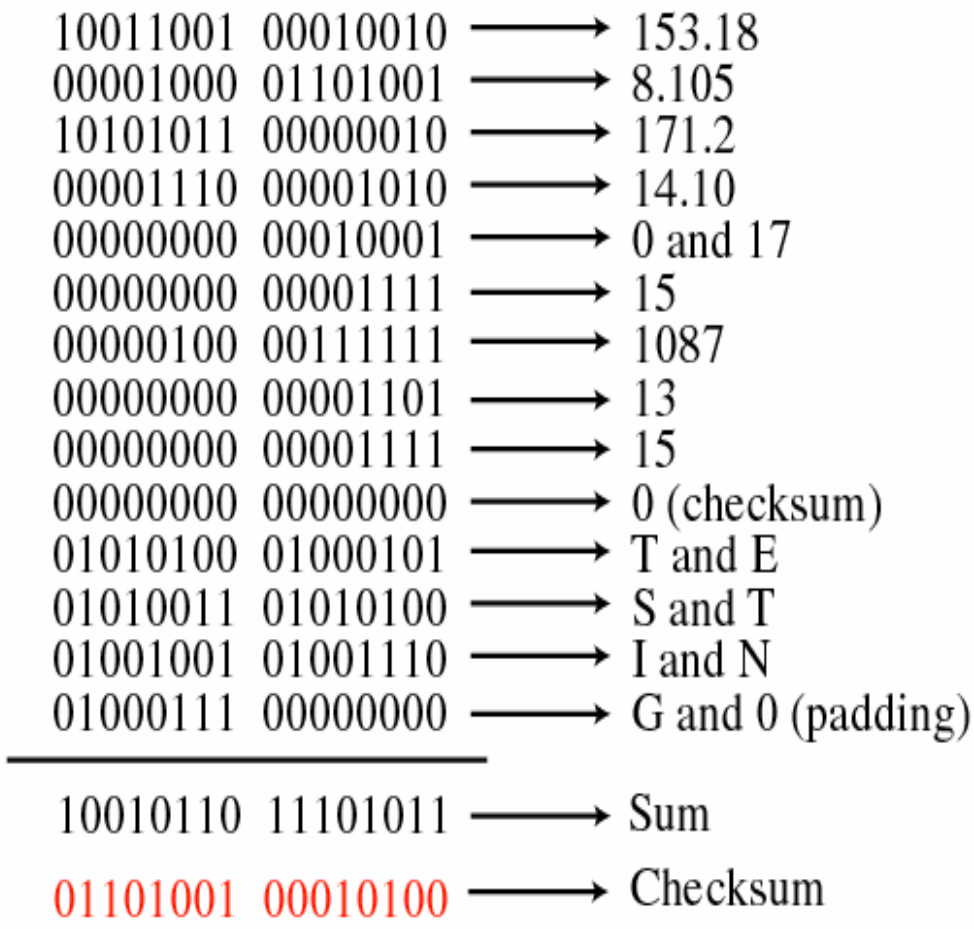
Checksum Calculation at Receiver

- Add the pseudoheader to the UDP user datagram
- Add padding if needed
- Divide the total bits into 16-bit sections
- Add all 16-bit sections using one's complement arithmetic
- Complement the result
- If the result is all 0s
  - Drop the pseudoheader and any added padding
  - Accept the user datagram
- Otherwise
  - Discard the user datagram

Figure 11.8

Checksum Calculation of a Simple UDP User Datagram





Optional Use of the Checksum

- The calculation of the checksum in UDP is ***optional***

- If the checksum is not calculated
  - The field is filled with 0s

11.4

UDP OPERATION

UDP Operation

- Connectionless Services
- Flow and Error Control
- Encapsulation and Decapsulation
- Queuing

Connectionless Services

- Each user datagram is an independent datagram
  - Even coming from the same source process
- The user datagram is not numbered

- No connection establishment and no connection termination

Flow and Error Control

- No flow control
  - The receiver may overflow with incoming messages
- No error control except for the checksum
  - The sender does not know if a message has been lost or duplicated

Figure 11-19

Encapsulation and Decapsulation

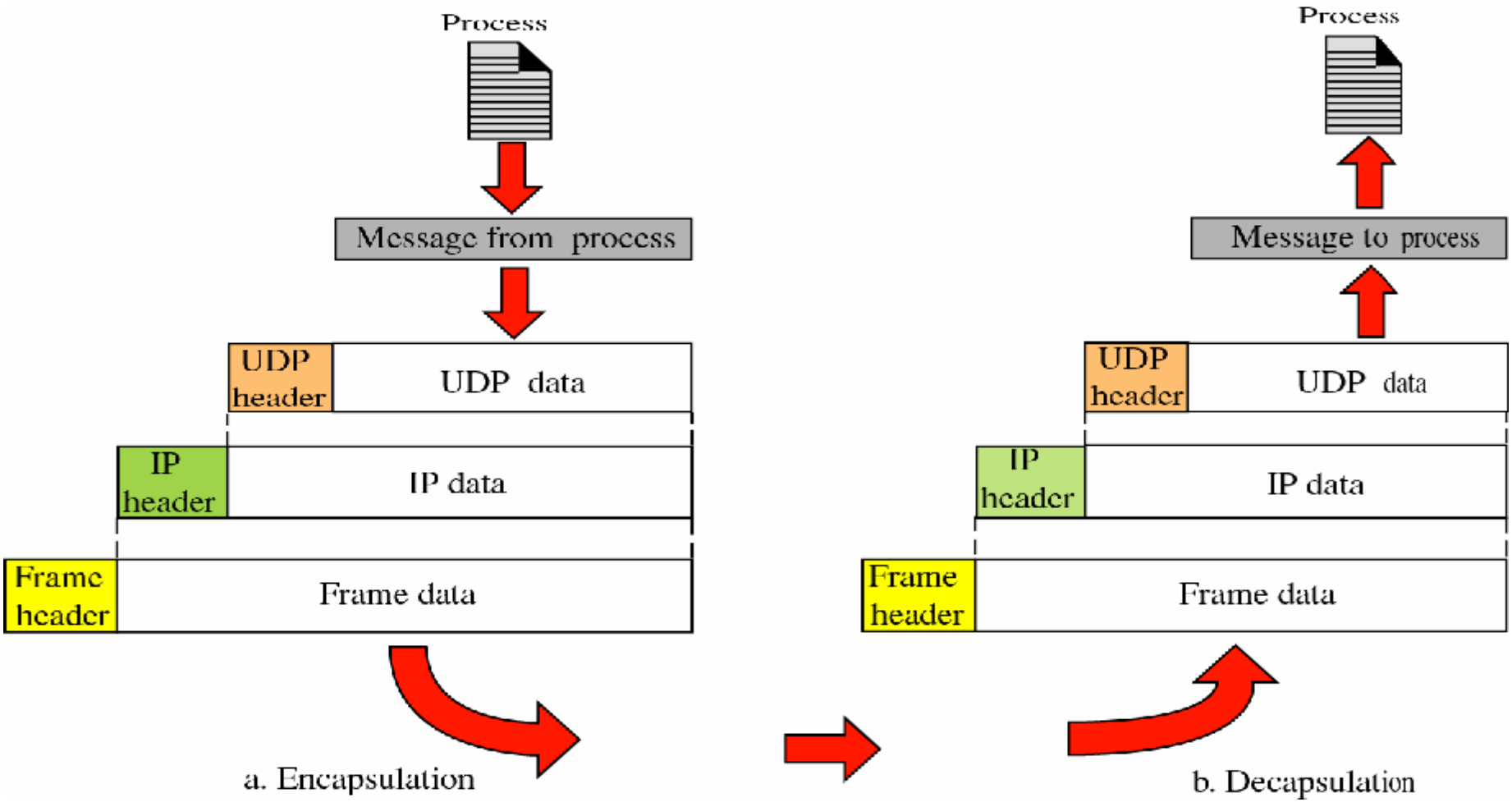


Figure 11.11

Queues in UDP

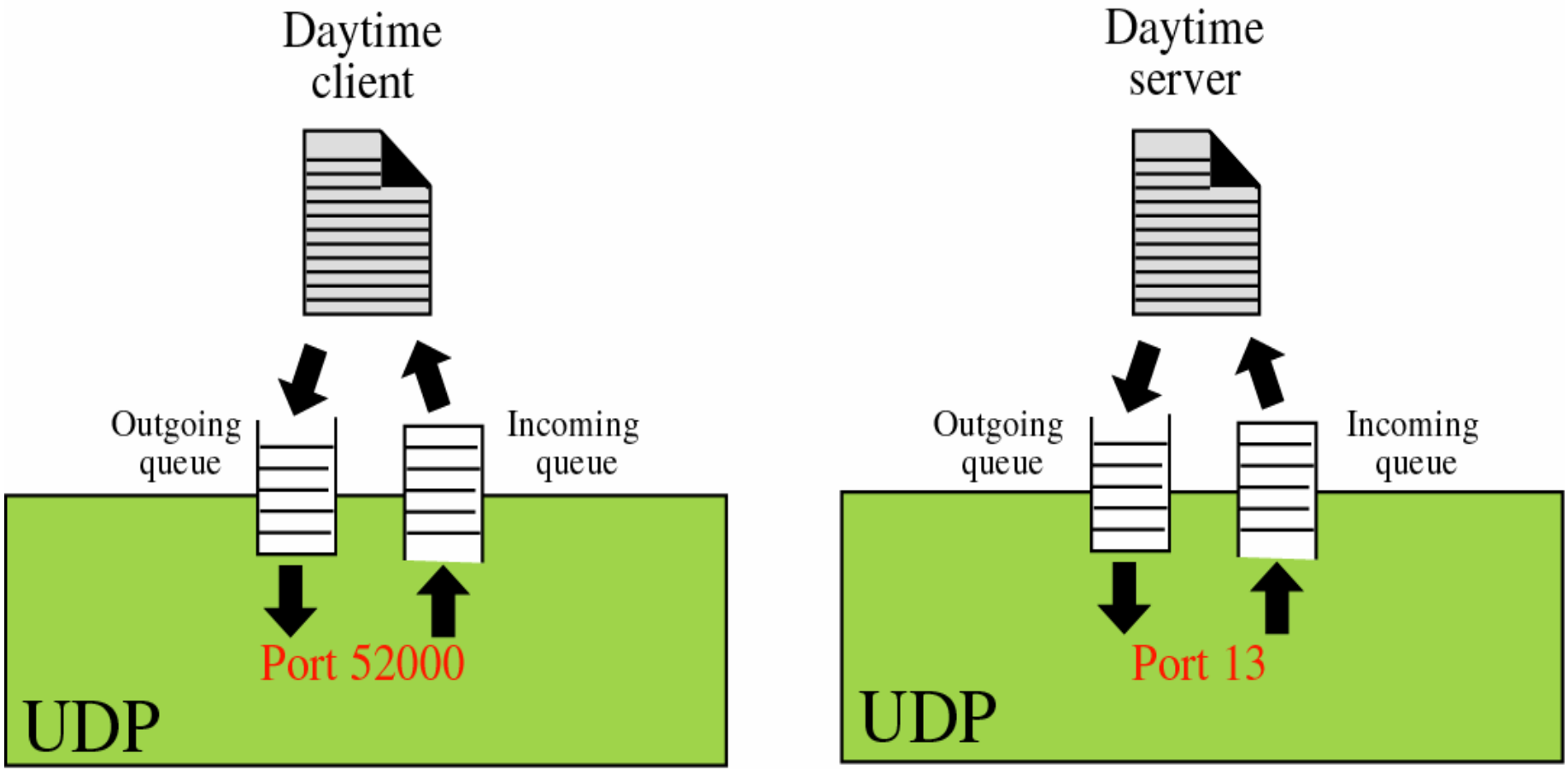
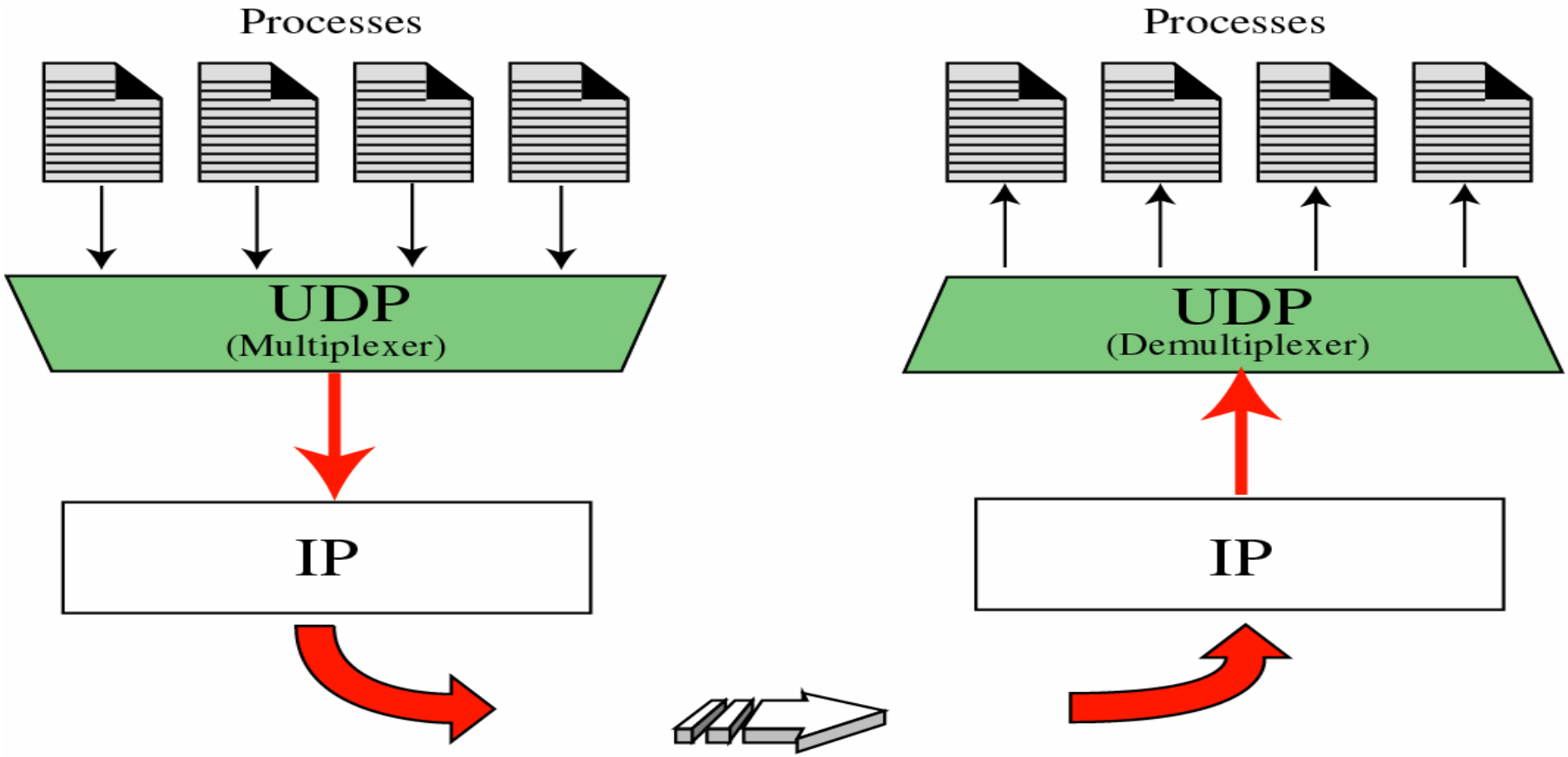


Figure 11-12

Multiplexing and Demultiplexing



11.5

USE OF UDP

Uses of UDP

- For a process that require simple request-response communication with little concern for flow and error control
- For a process with internal flow and error-control
  - TFTP (Trivial File Transfer Protocol)
- For muticasting and broadcasting
  - Embedded in UDP but not in the TCP software
- For some management processes
- For some route updating protocol
  - RIP (Routing Information Protocol)

11.6

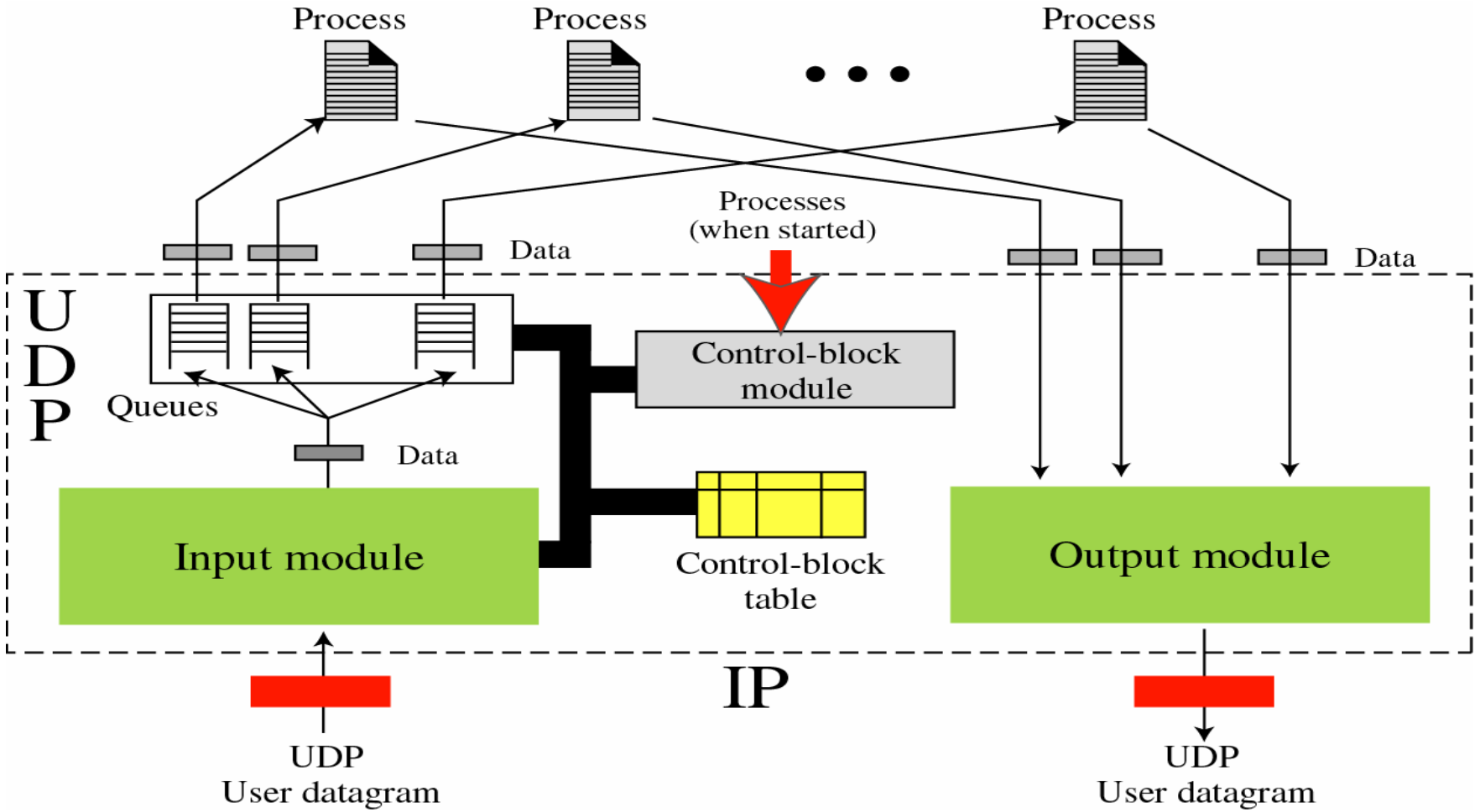
UDP PACKAGE

UDP Package

- Five components
  - A control-block table
  - Input queues
  - A control-block module
  - An input module
  - An output module

Figure 11.12

UDP Package



Control-Block Table

- Keep track of the open ports

- Each entry has a minimum of four fields
  - State: FREE or IN-USE
  - Process ID
  - Port number
  - Corresponding queue number

Input Queues

- A set of input queues
  - One for each process

- In the book's design
  - They do not use output queues

Control-Block Module

- Manage the control-block table
- Pseudo code
  - Receive: a *process ID* and a *port number*
  - Search the control block table for a FREE entry
    - If (not found)
      - Delete an entry using a predefined strategy
    - Create a new entry with the state IN-USE
    - Enter the process ID and the port number
  - Return

Input Module

- Receive: a user datagram from IP
- Look for the corresponding entry in the control-block table
  - If (found)
    - Check the queue field to see if a queue is allocated
    - If (no)
      - Allocate a queue
    - Enqueue the data in the corresponding queue
  - If (not found)
    - Ask the ICMP module to send an “unreachable port” message
    - Discard the user datagram
- Return

Output Module

- Receive: data and information from a process
- Create a UDP data datagram
- Send the user datagram
- Return

Example: Control-Block Table at the Beginning

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 1

- 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
  - The port has been previously used
- The input module sends the data to queue 38
- The control-block table does not change

Example 2

- 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 does not allocate a queue at this moment
  - because no user datagram have arrived for this destination

Modified Table After Example 2

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 3

- 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
  - This is the first time a user datagram has arrived for this destination
- The module creates a queue and gives it a number (43)

Modified Table After Example 3

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 4

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

Example 5

- After a few seconds, a process needs to send a user datagram
  - It delivers the data to the output module which adds the UDP header and sends it