OPERATING SYSTEMS

Subject code : CSC 404

Subject In-charge

Nidhi Gaur
Assistant Professor
email: nidhigaur@sfit.ac.in

Module 6 : I/O Management and Disk scheduling

I/O Management

- A large portion of OS code is used to manage I/O
- I/O is important due to performance and because of varying nature of devices, i/o management is challenging task.
- The software which deals with I/O is known as I/O software.

I/O Management

The I/O software consists of two modules:

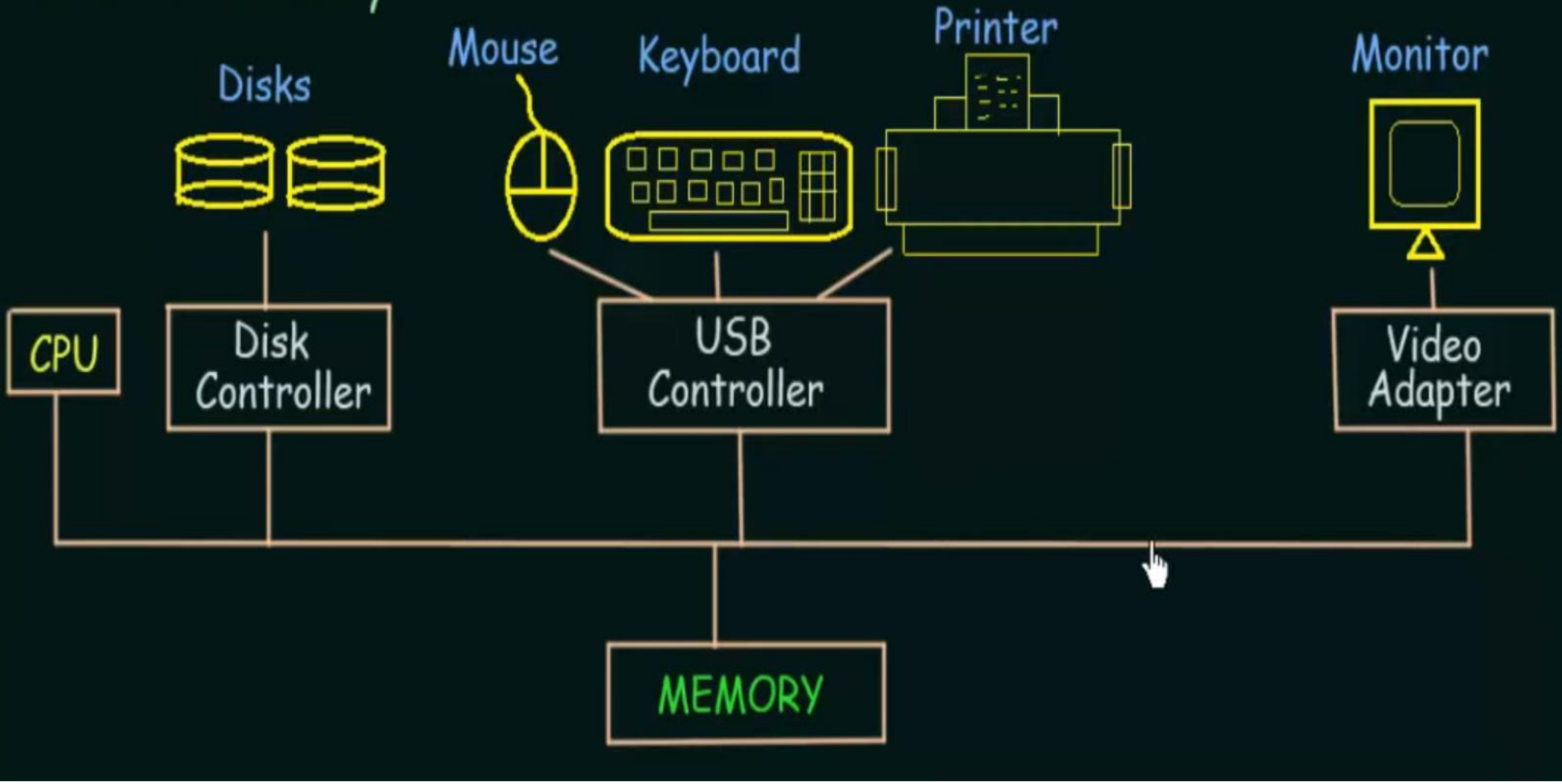
- First deals with general functionalities when interfacing with any type of device, known as device independent software.
- Second module provides device specific code for controlling it and is known as device driver.

Hence OS need not change its code again and again to incorporate any new device.

Basics of Operating System (Computer System Operation)

Some basic knowledge of the structure of Computer System is required to understand how Operating Systems work.

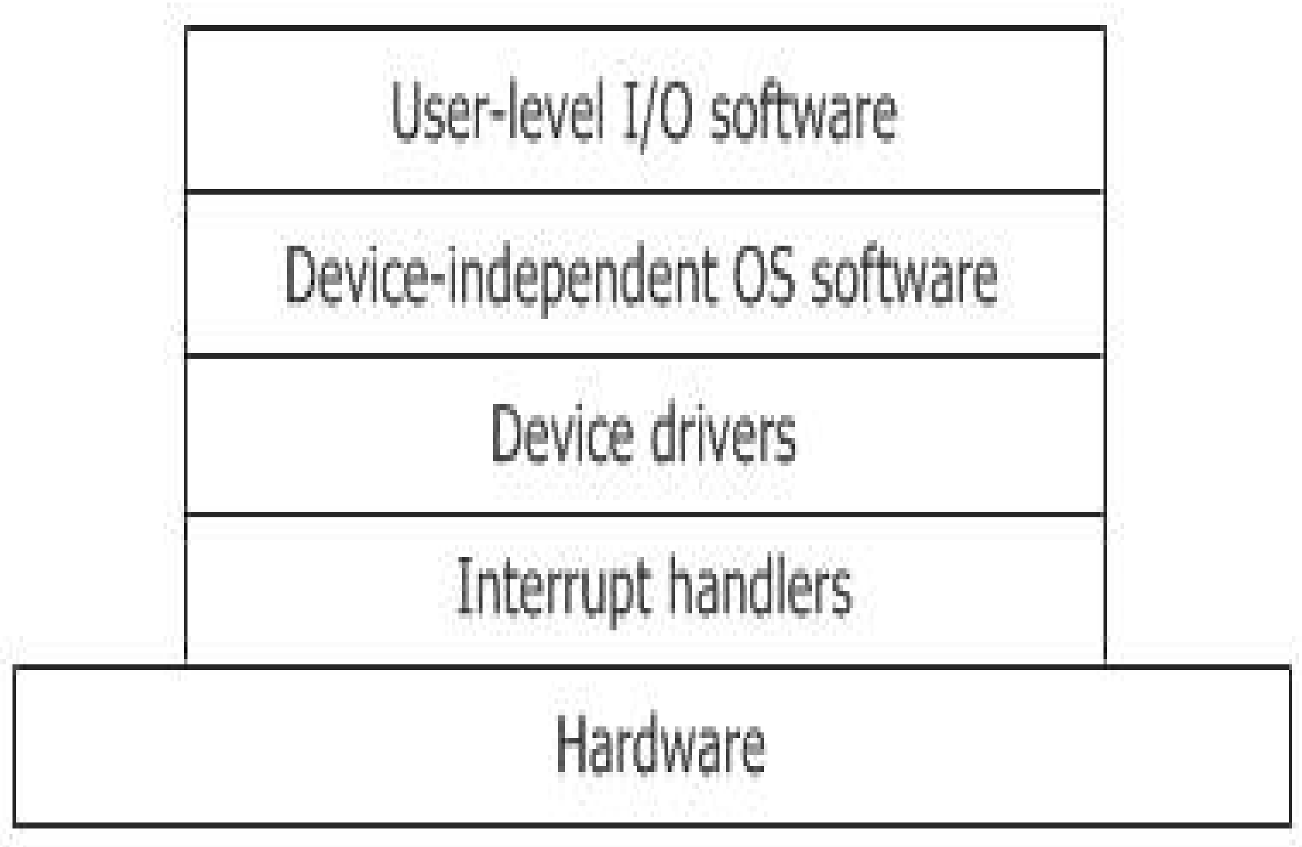
→ A modern general-purpose computer system consists of one or more CPUs and a number of device controllers connected through a common bus that provides access to shared memory.



I/O Management

- Each device controller is incharge of specific type of device.
- Device controller maintains local buffer storage and special purpose registers.
- OS has device driver for each device controller
- Device driver understands device controller and presents uniform interface to device, to rest of the OS.

Layered structure of I/O



User-level I/O software

Device-independent OS software

Device drivers

Interrupt handlers

Hardware

I/O management

- I/O software consists of user I/O software, kernel I/O subsystem and device driver.
- Kernel I/O subsystem provides I/O scheduling, caching, spooling and buffering.
- Device driver acts as an interface with the hardware.

Application I/O interface

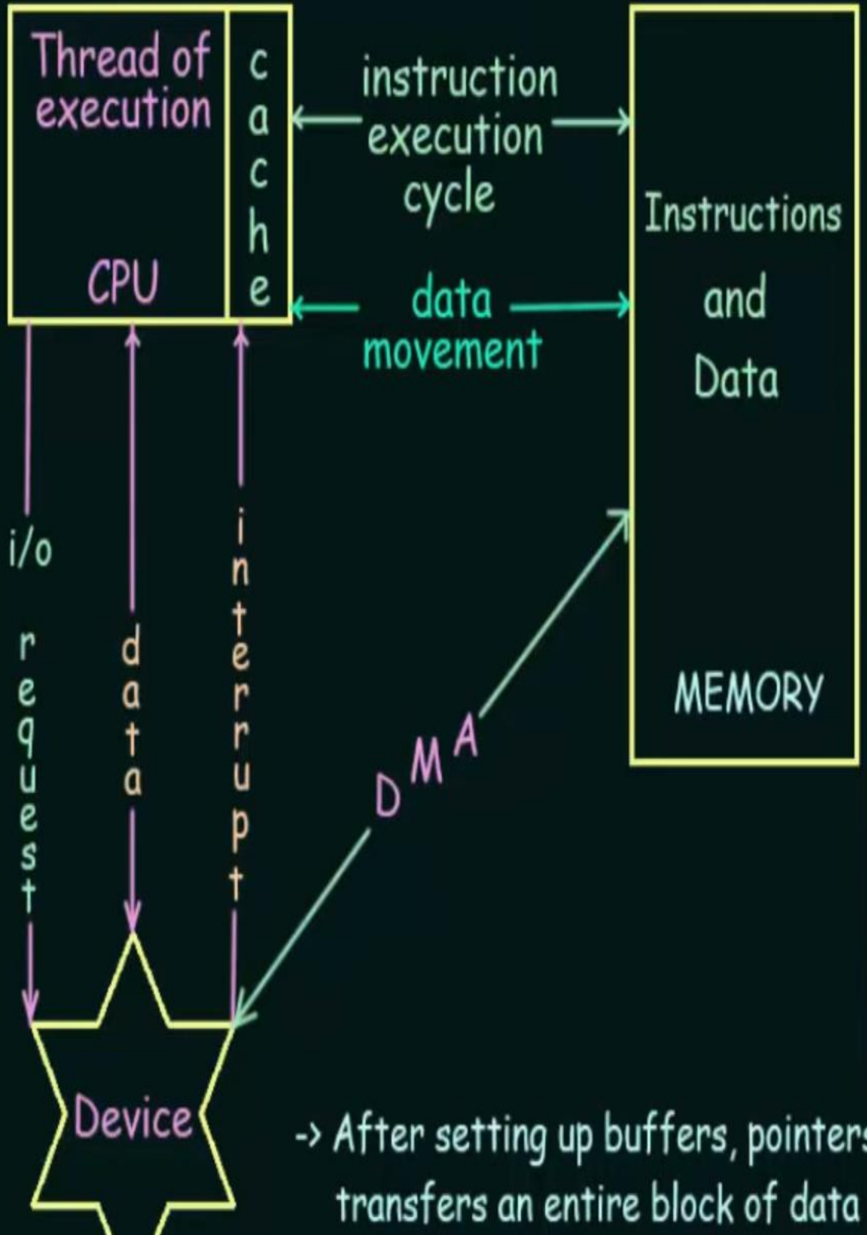
- Interface for OS that enable I/O devices to be treated in a uniform way.
- Detailed differences are abstracted
- Differences are encapsulated in kernel modules called device drivers
- The purpose of the device driver layer is to hide the differences among device controllers from I/O subsystem of the kernel.

I/O management

- I/O hardware consists of two parts:

Device controller and hardware device.

Working of an I/O Operation:



-> To start an I/O operation, the device driver loads the appropriate registers within the device controller

-> The device controller, in turn, examines the contents of these registers to determine what action to take

-> The controller starts the transfer of data from the device to its local buffer

-> Once the transfer of data is complete, the device controller informs the device driver via an interrupt that it has finished its operation

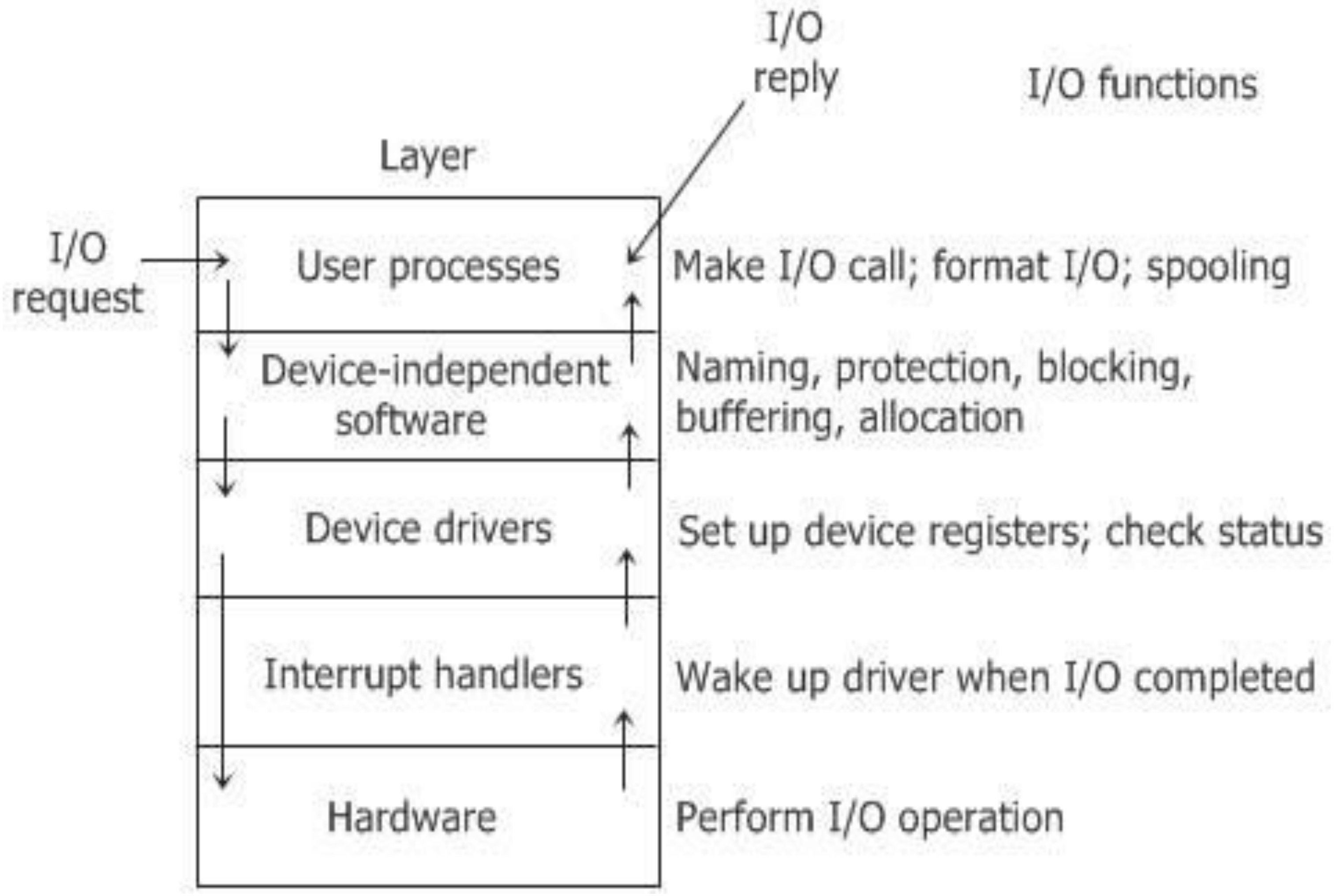
-> The device driver then returns control to the operating system

This form of interrupt-driven I/O is fine for moving small amounts of data but can produce high overhead when used for bulk data movement

To solve this problem, Direct Memory Access (DMA) is used

-> After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data directly to or from its own buffer storage to memory, with

- Using DMA only one interrupt is generated per block to tell the device driver that operation completed.
- While without DMA interrupt is generated per byte.
- Hence using DMA, CPU is available to accomplish other useful tasks.



I/O reply

I/O functions

Layer

I/O request

Layer	I/O functions
User processes	Make I/O call; format I/O; spooling
Device-independent software	Naming, protection, blocking, buffering, allocation
Device drivers	Set up device registers; check status
Interrupt handlers	Wake up driver when I/O completed
Hardware	Perform I/O operation

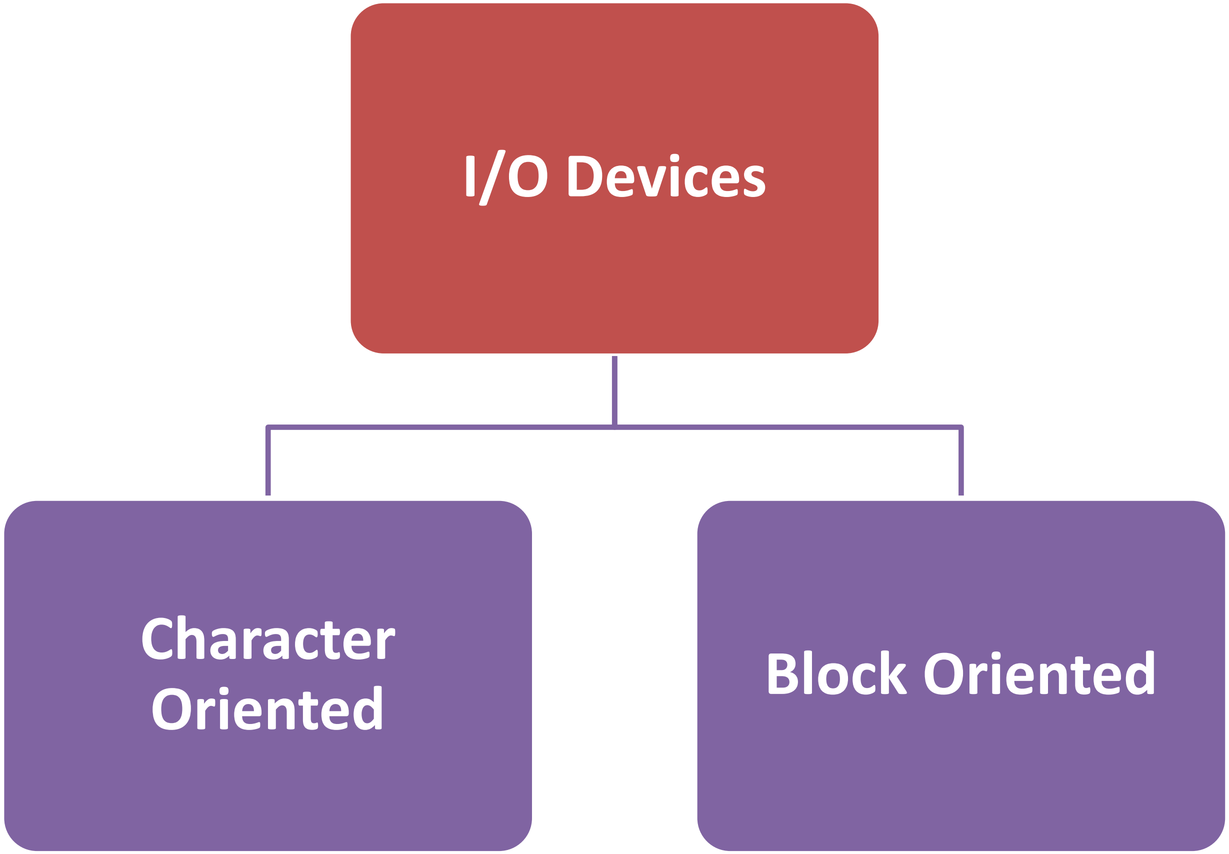
Categories of I/O Devices

- Human readable
  - Used to communicate with the user
  - Printers
  - Video display terminals
    - Display
    - Keyboard
    - Mouse

Categories of I/O Devices

- Machine readable
  - Used to communicate with electronic equipment
  - Disk and tape drives
  - Sensors
  - Controllers

Types of I/O devices


I/O Devices

Character Oriented

Block Oriented

Block and Character Devices

- Block devices include disk drives
  - Commands include read, write, seek
- Character devices include keyboards, mice, serial ports
  - Commands include `get, put`

Differences in I/O Devices

- Data rate
  - – May be differences of several orders of magnitude between the data transfer rates

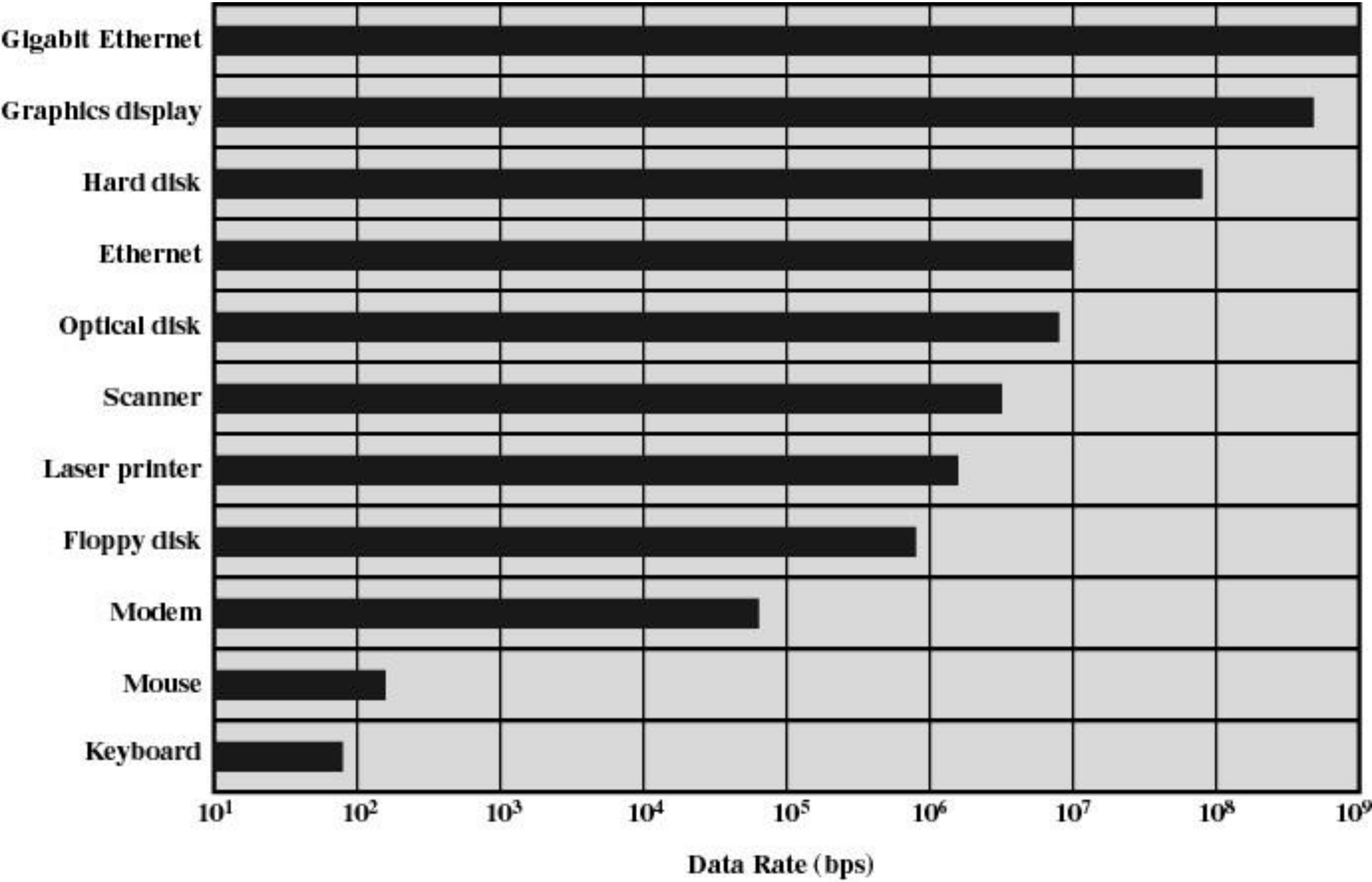
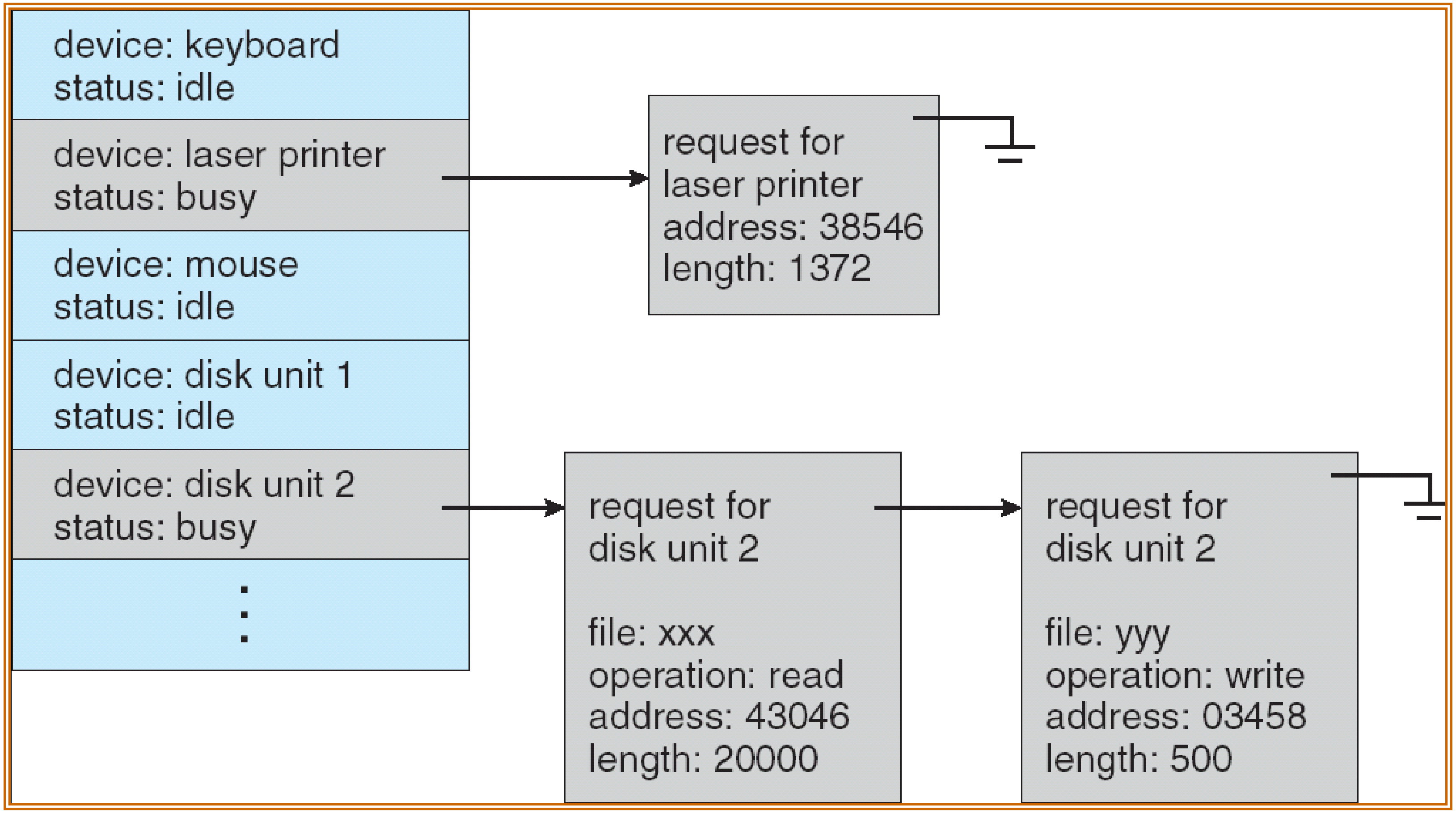
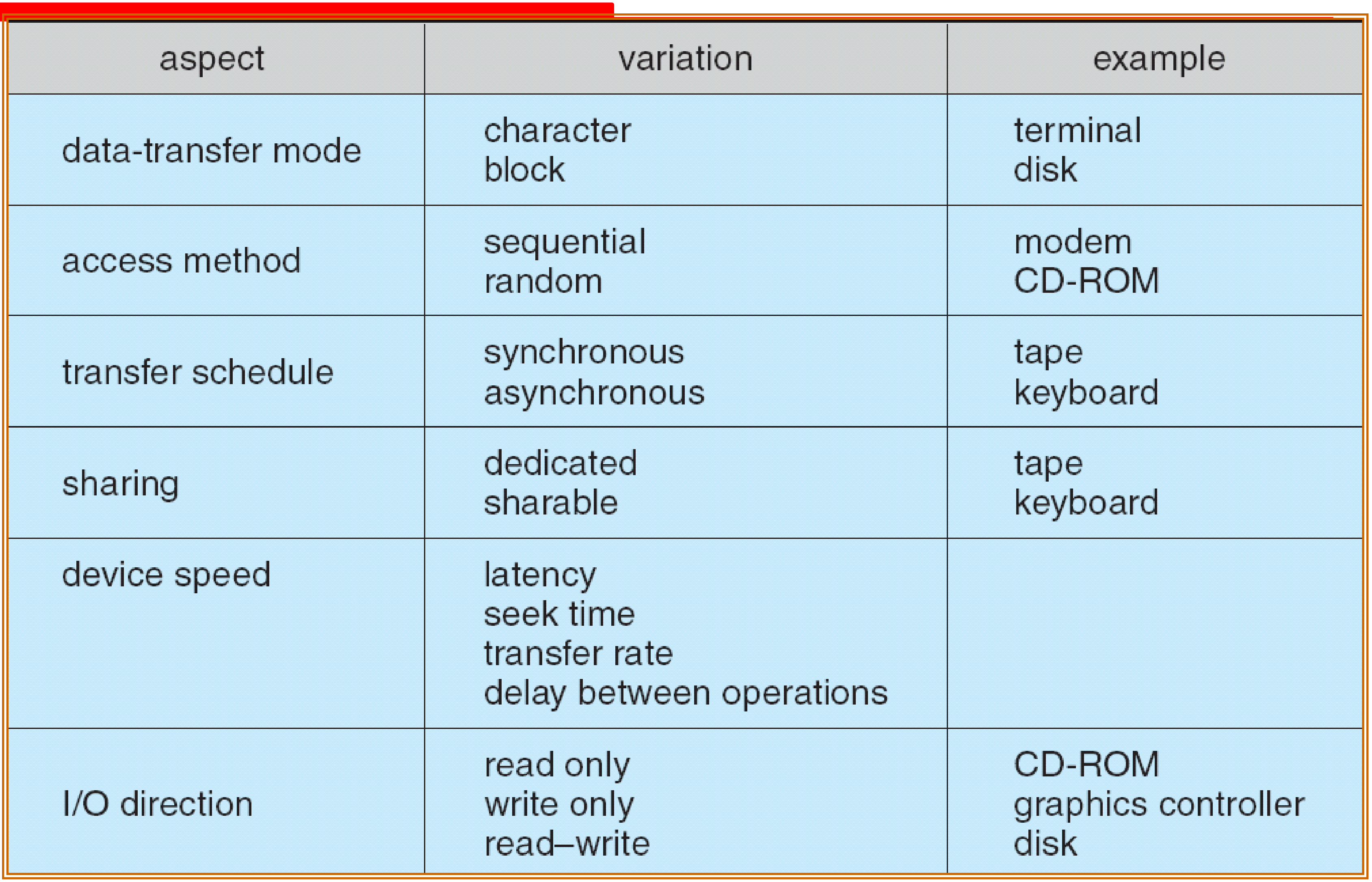


Figure 11.1 Typical I/O Device Data Rates

Device-status Table



Characteristics of I/O Devices



aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

Differences in I/O Devices

- Complexity of control
- Unit of transfer
  - Data may be transferred as a stream of bytes for a terminal or in larger blocks for a disk
- Data representation
  - Encoding schemes
- Error conditions
  - Devices respond to errors differently

Differences in I/O Devices

- Programmed I/O
  - Process is busy-waiting for the operation to complete
- Interrupt-driven I/O
  - I/O command is issued
  - Processor continues executing instructions
  - I/O module sends an interrupt when done

Evolution of the I/O Function

- Processor directly controls a peripheral device
- Controller or I/O module is added
  - Processor uses programmed I/O without interrupts
  - Processor does not need to handle details of external devices

Evolution of the I/O Function

- Controller or I/O module with interrupts
  - Processor does not spend time waiting for an I/O operation to be performed
- Direct Memory Access
  - Blocks of data are moved into memory without involving the processor
  - Processor involved at beginning and end only

Evolution of the I/O Function

- I/O module is a separate processor
- I/O processor
  - – I/O module has its own local memory
  - – Its a computer in its own right

I/O Buffering

Buffering is done **to deal effectively with a speed mismatch between the producer and consumer of the data stream**.

An I/O buffer is a memory area where data is stored temporarily during I/O operation.

- Reasons for buffering
  - Processes must wait for I/O to complete before proceeding
  - Certain pages must remain in main memory during I/O

I/O Buffering

- Block-oriented
  - Information is stored in fixed sized blocks
  - Transfers are made a block at a time
  - Used for disks and tapes
- Stream-oriented
  - Transfer information as a stream of bytes
  - Used for terminals, printers, communication ports, mouse, and most other devices that are not secondary storage

Single Buffer

- Operating system assigns a buffer in main memory for an I/O request
- Block-oriented
  - Input transfers made to buffer
  - Block moved to user space when needed
  - Another block is moved into the buffer
    - Read ahead

I/O Buffering

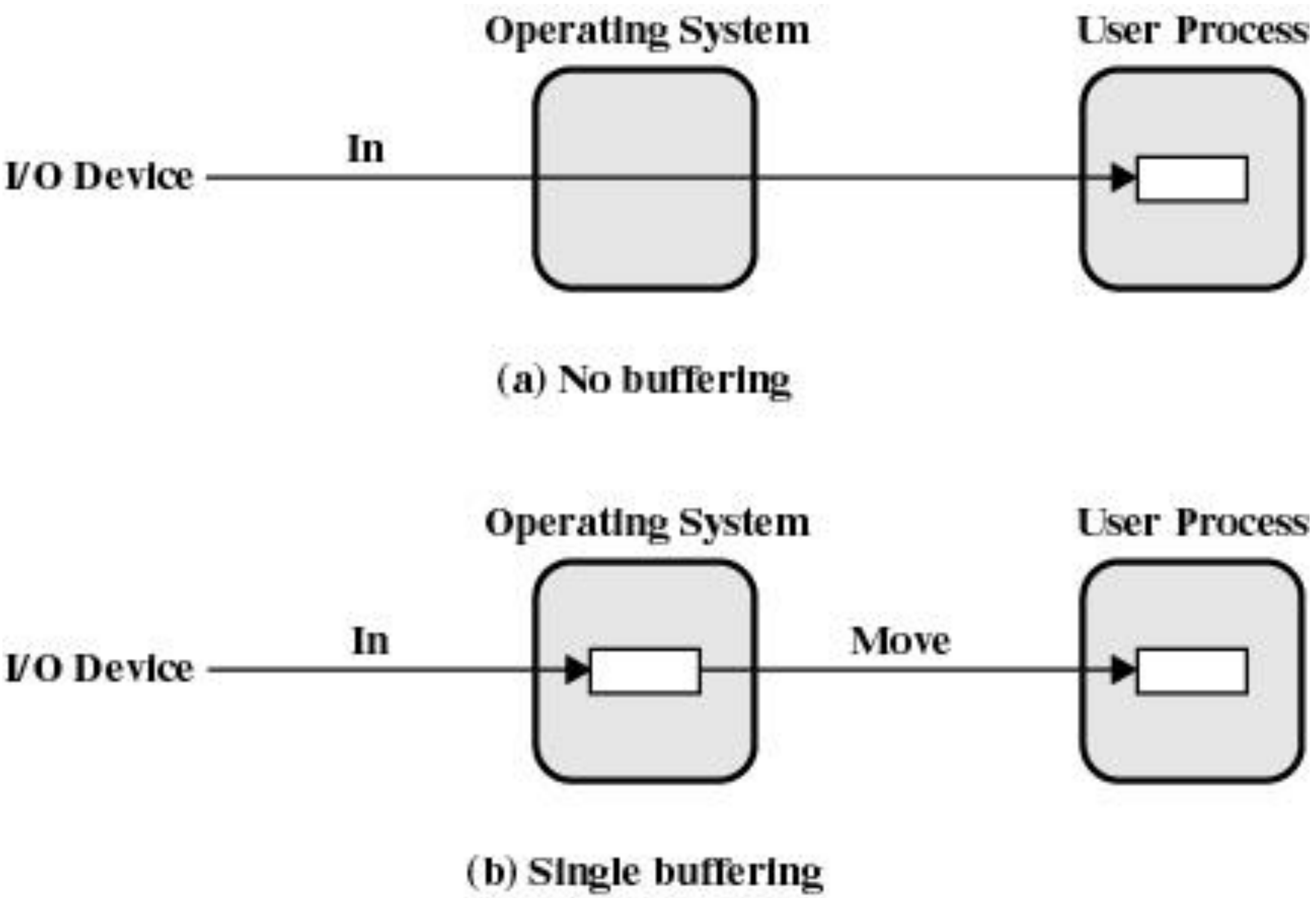


Figure 11.6 I/O Buffering Schemes (input)

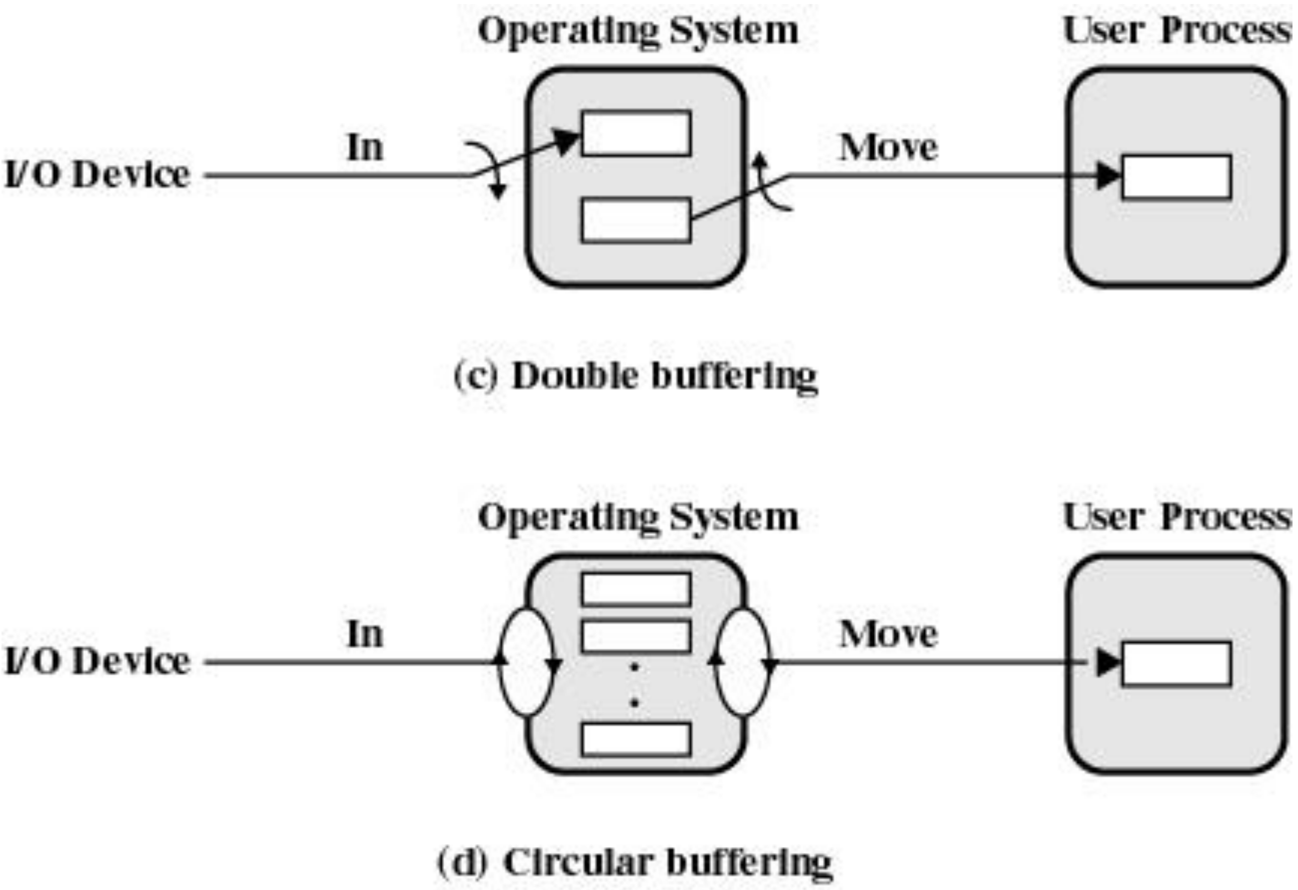
Single Buffer

- Block-oriented
  - User process can process one block of data while next block is read in
  - Operating system keeps track of assignment of system buffers to user processes

Single Buffer

- Stream-oriented
  - Used a line at time
  - User input from a terminal is one line at a time with carriage return signaling the end of the line
  - Output to the terminal is one line at a time

I/O Buffering



(c) Double buffering

(d) Circular buffering

Figure 11.6 I/O Buffering Schemes (input)

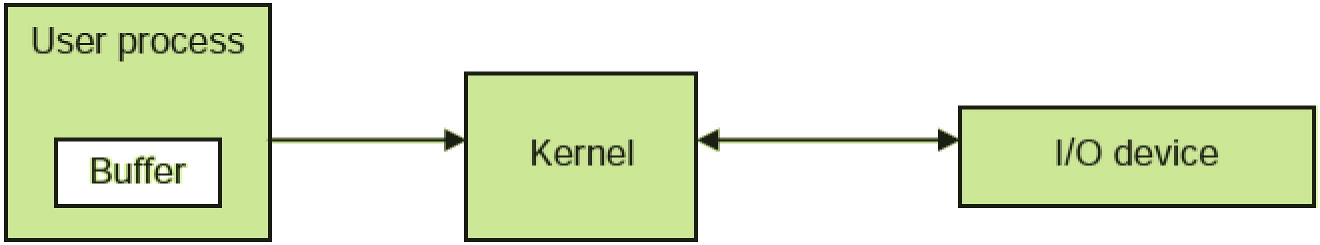
Double Buffer

- Use two system buffers instead of one
- A process can transfer data to or from one buffer while the operating system empties or fills the other buffer

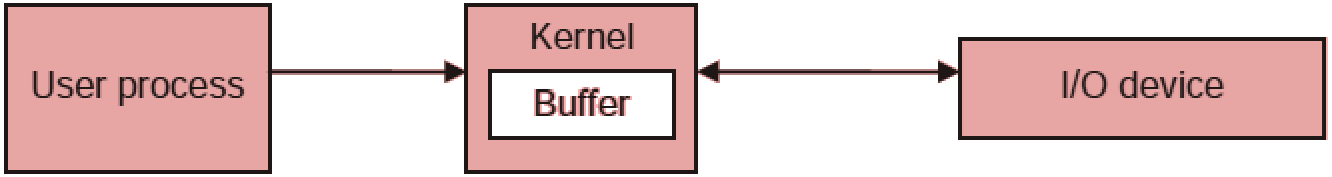
Circular Buffer

- More than two buffers are used
- Each individual buffer is one unit in a circular buffer
- Used when I/O operation must keep up with process

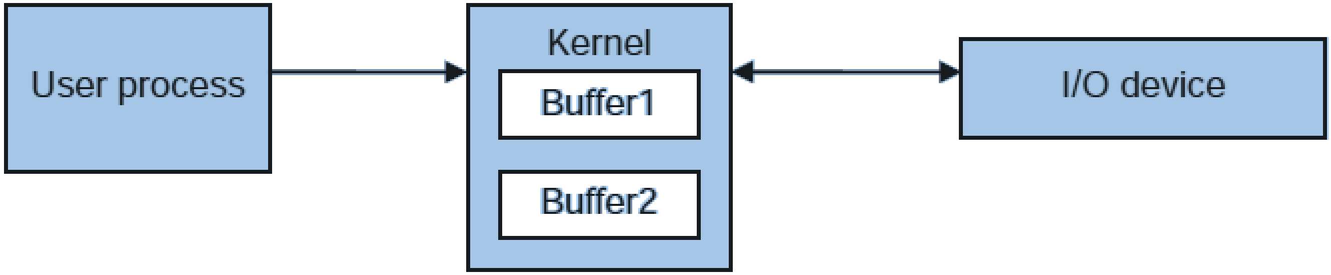
Types of buffering



Buffering in user space



Buffering in kernel space



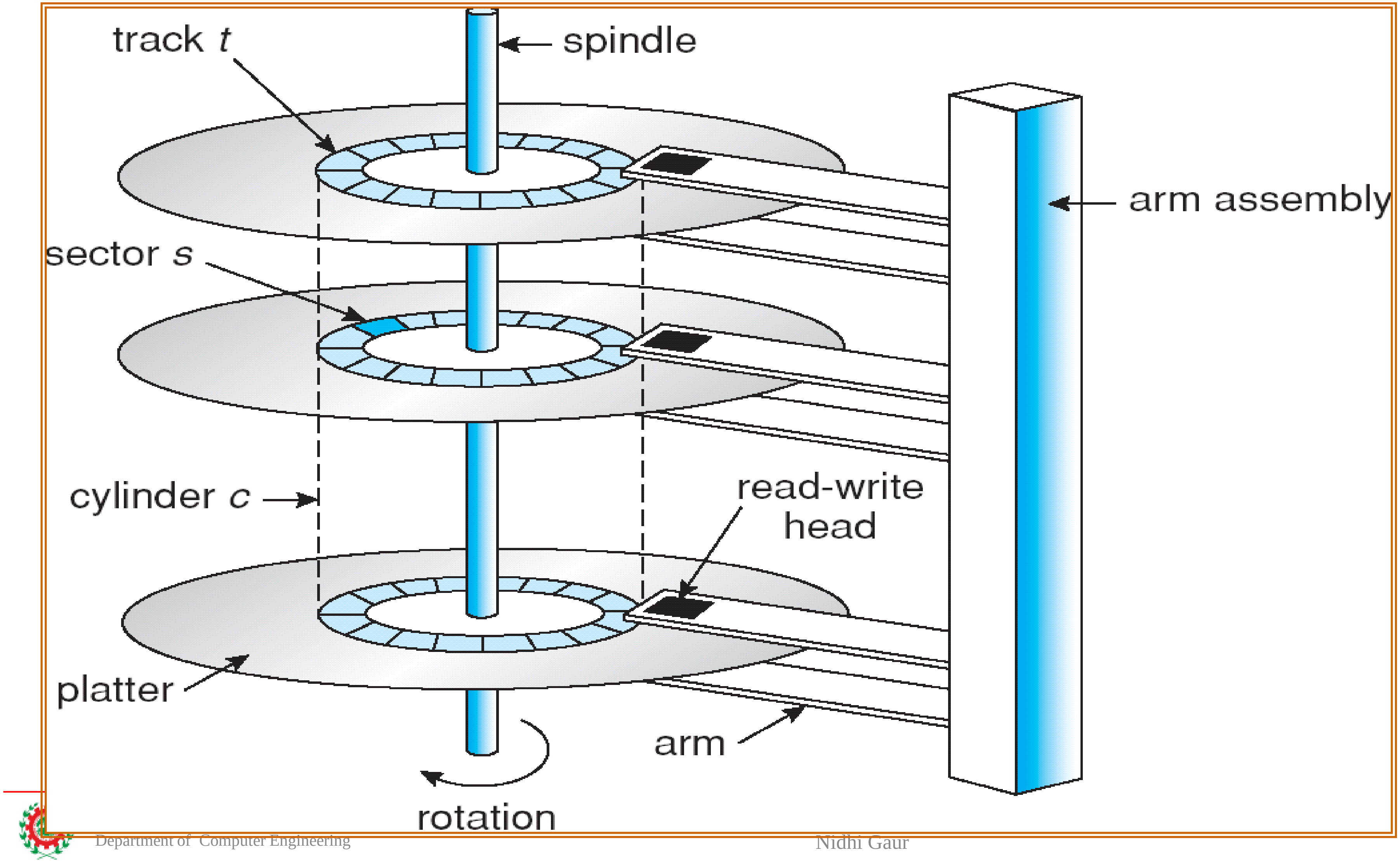
Double buffering

- DISK MANAGEMENT

Disk Layout

- “Platter” – Circular Piece of Magnetic Material – Coated Aluminum
- One Platter Has 2 “Sides”
- Each Side is Divided into Concentric Rings Called “Tracks”
- Each Track is Divided into Arcs Called “Sectors”
- Each Sector Can Store Some Number of “Bytes” of Data
- “Media Capacity” Determines Amount of Storage per Sector

Moving-head Disk Mechanism



Disk management

- A Platter can be “Double Sided” or “Single Sided”
- A Single Fixed Disk can Contain Multiple Platters
- A Stack of Tracks On Top of Each Other is a “Cylinder”
- Each Side of a Platter Must Have a “Read/Write Head”

Contains Induction Coil – Floats Over Surface

- Position of Read/Write Head Given by “Cylinder” Number
- Physical Location on Disk: <cylinder, head, sector>

The first commercial hard-disk drive, the IBM 305 RAMAC, used forced air to maintain a 0.002-inch (51-micron) spacing between the head and disk.

Disk management

- Magnetic disks provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 200 times per second
  - **Transfer rate** is rate at which data flow between drive and computer
  - **Head crash** results from disk head making contact with the disk surface

Disk Performance Parameters

- To read or write, the disk head must be positioned at the desired track and at the beginning of the desired sector
- Seek time
  - time it takes to position the head at the desired track
- Rotational delay or rotational latency
  - time it takes for the beginning of the sector to reach the head

Timing of a Disk I/O Transfer

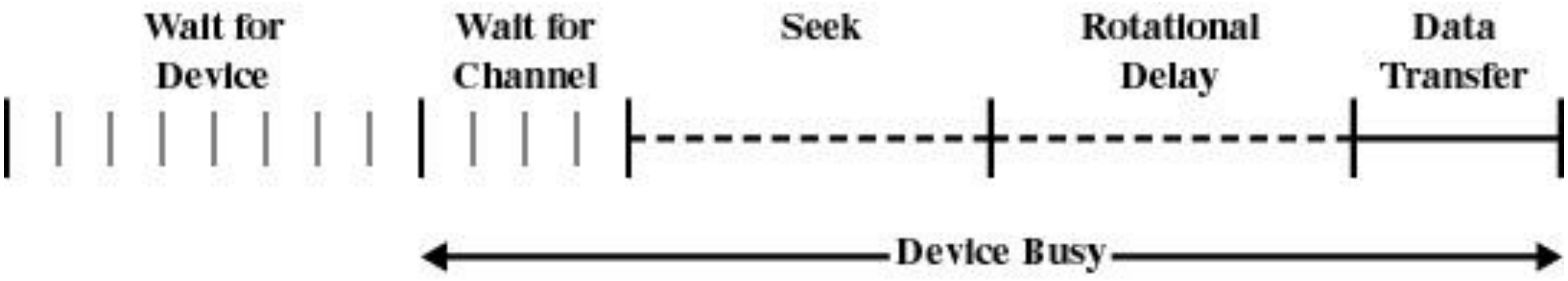


Figure 11.7 Timing of a Disk I/O Transfer

Disk Performance Parameters

- Access time
  - Sum of seek time and rotational delay
  - The time it takes to get in position to read or write
- Minimize seek time
- Data transfer occurs as the sector moves under the head

- Each Disk Drive Has a Controller Associated With It
- Purpose of Controller:

1) Accept Commands from Software (READ, WRITE, FORMAT)
2) Control Arm Motion
3) Detect/Correct Errors
4) Convert parallel data into Serial Bit-Stream (vice-versa)
5) Remap bad Sectors to Spares (each track has spares)

Disk Scheduling Policies

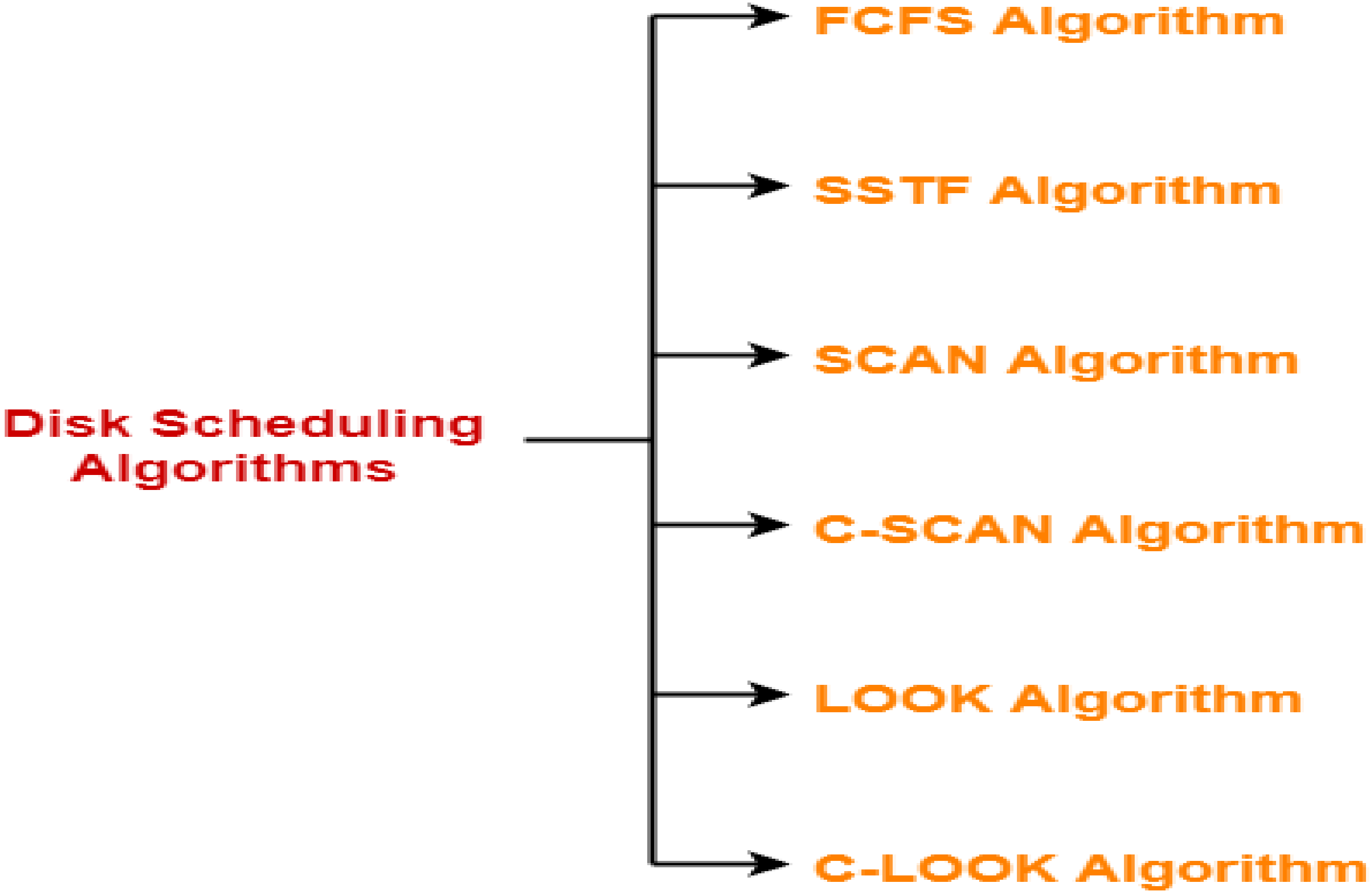
- Seek time is the reason for differences in performance
- For a single disk there will be a number of I/O requests
- If requests are selected randomly, we will get the worst possible performance

Disk Scheduling (Cont.)

- Several algorithms exist to schedule the servicing of disk I/O requests.
- We illustrate them with a request queue (0-199).

98, 183, 37, 122, 14, 124, 65, 67

- Head pointer 53



Disk Scheduling Algorithms

- FCFS Algorithm
- SSTF Algorithm
- SCAN Algorithm
- C-SCAN Algorithm
- LOOK Algorithm
- C-LOOK Algorithm

Desired disk performance parameters

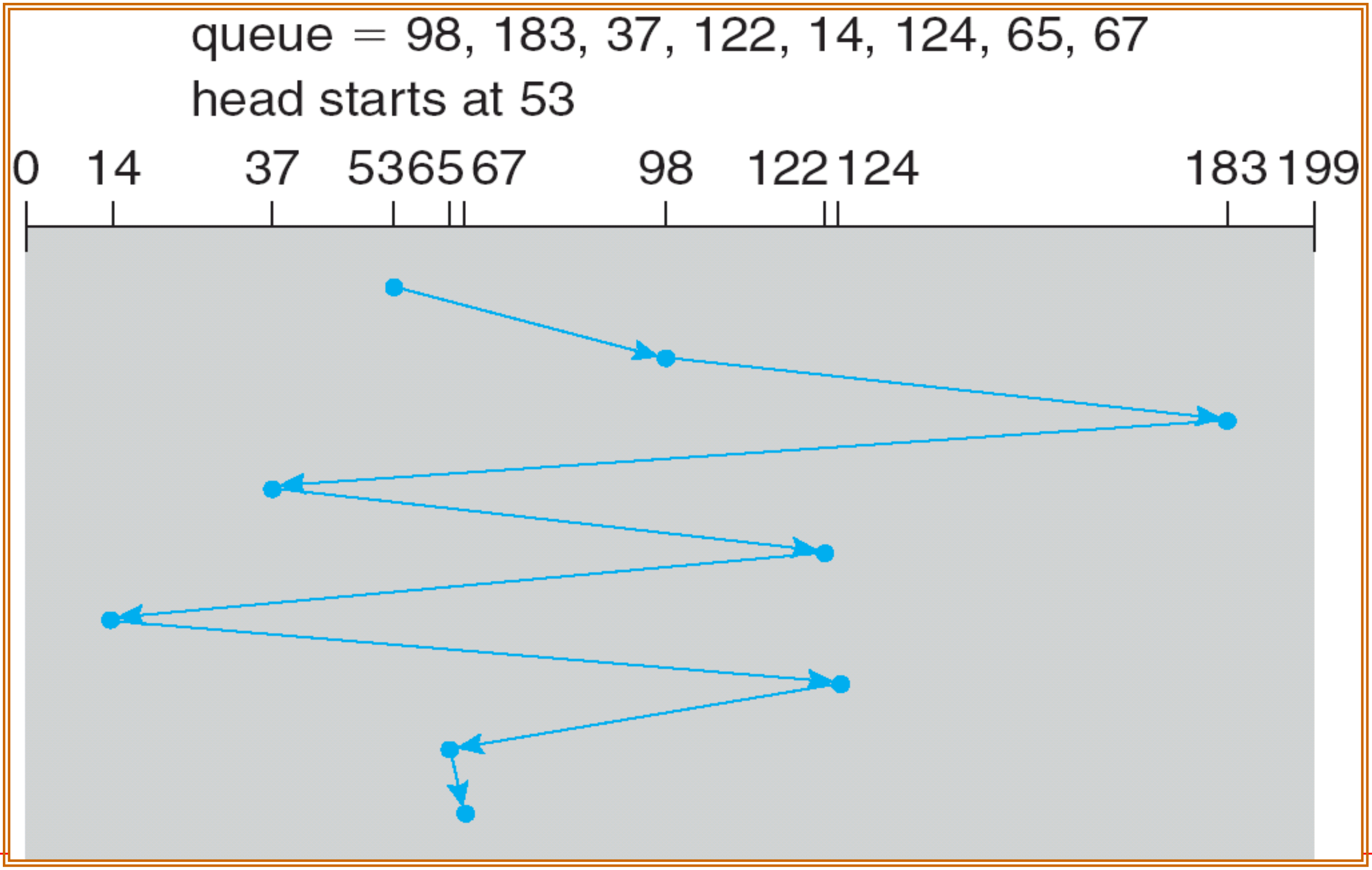
Parameter	Desired performance
Seek time	Minimize
Rotational latency	Minimize
Transfer time	Minimize
Bandwidth	Maximize
Throughput	Maximize
Average Response time	Minimize
Variance of Response time	Minimize

Disk Scheduling Policies

- First-in, first-out (FIFO/FCFS)
  - Process request sequentially
  - Fair to all processes
  - Approaches random scheduling in performance if there are many processes

FCFS

Find out total head movement and average seek length.

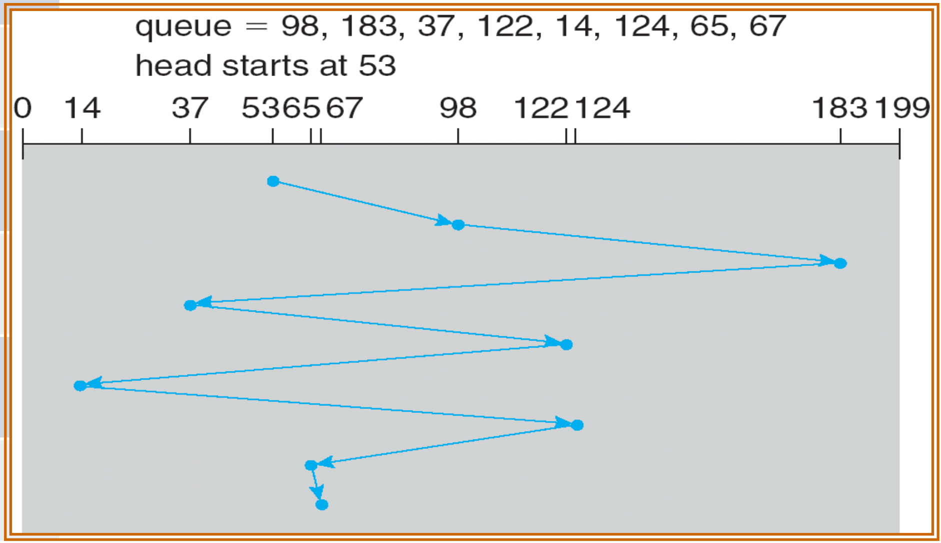

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53

FCFS

I/O REQUESTS: 98, 183, 37, 122, 14, 124, 65, 67
Head starts at 53

	Next cylinder to be serviced	Total head movement for request
1	98	45
2	183	85
3	37	146
4	122	85
5	14	108
6	124	110
7	65	59
8	67	2
	Total head movement	**640**

Average seek length for FCFS
= total head movement/ number of requests
= 640/8 =80


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53

FCFS

Consider a disk queue with I/O requests on following cylinders in their arriving order: 54, 97, 73, 128, 15, 44, 110, 34, 45 (0 to 128 cylinders)
Head starts at 23. Calculate and show with diagram disk head movement.

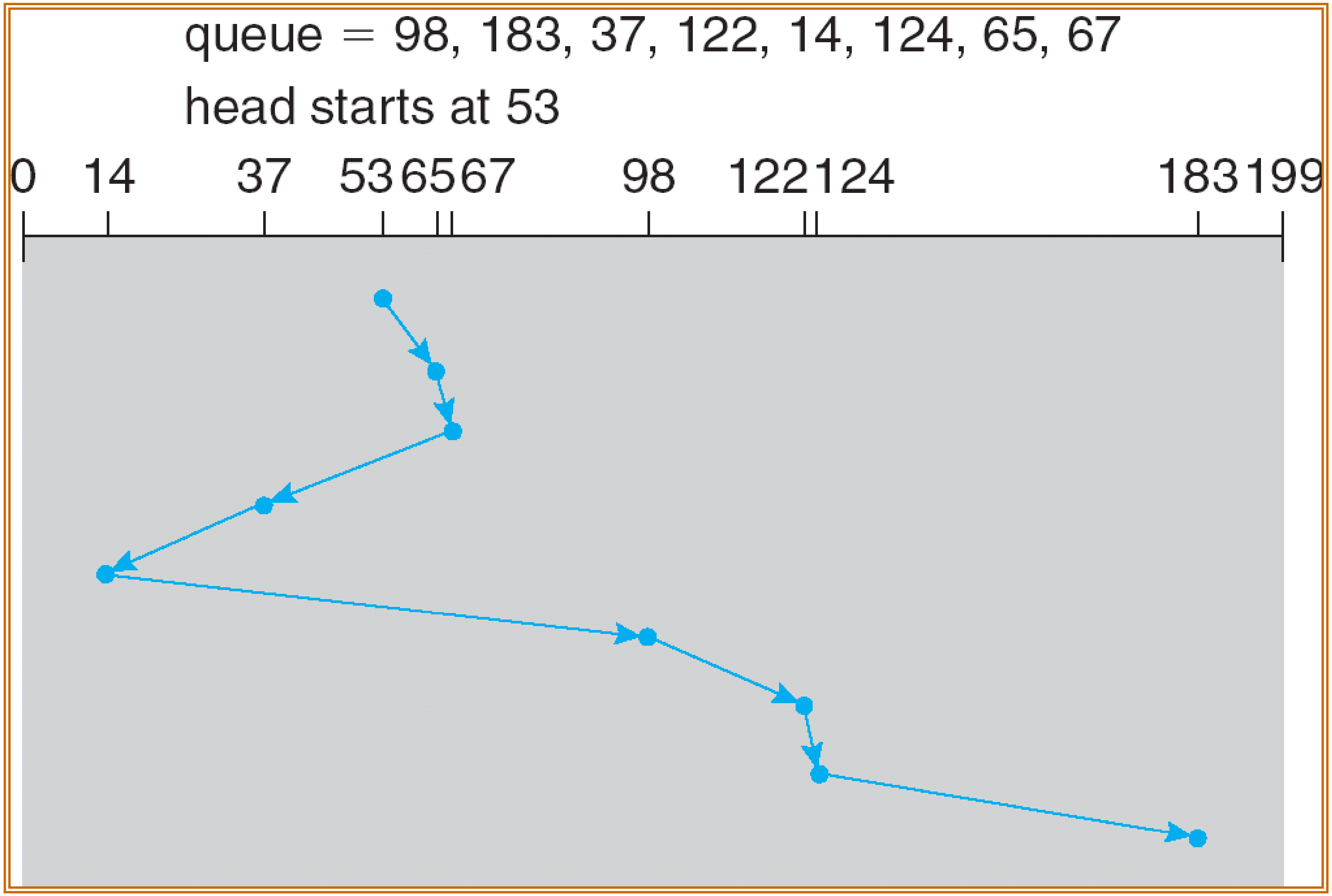
Next cylinder to be serviced	Total head movement for request
54	31
97	43
73	24
128	55
15	113
44	29
110	66
34	76
45	11 **Total = 448**

Average seek length = 49.7

SSTF

- Selects the request with the minimum seek time from the current head position.
  - Select the disk I/O request that requires the least movement of the disk arm from its current position
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.

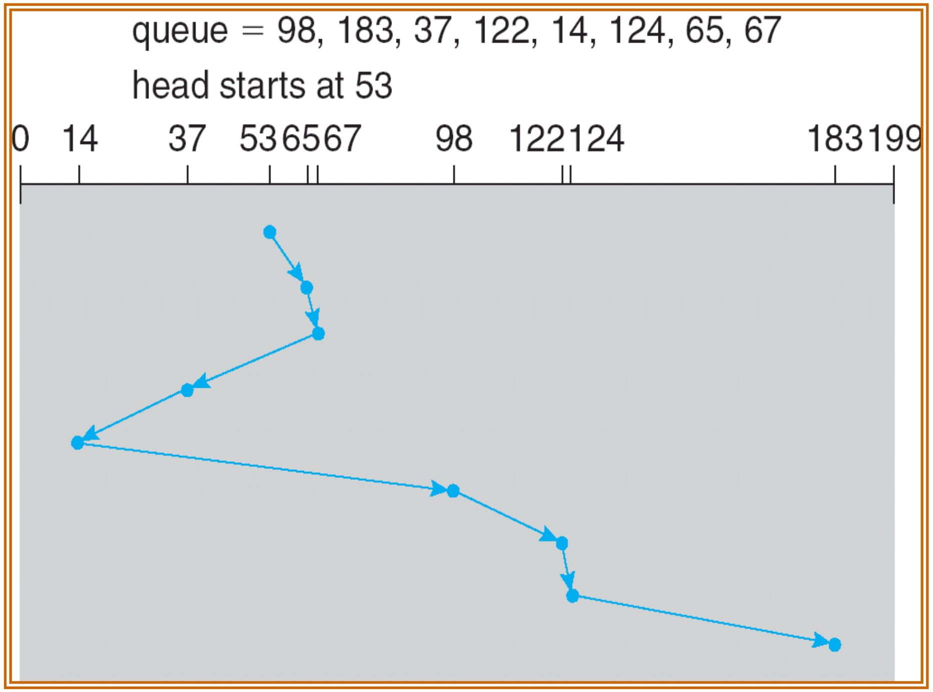
SSTF (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

SSTF

Consider a disk queue with I/O requests on following cylinders in their arriving order: I/O REQUESTS: 98, 183, 37, 122, 14, 124, 65, 67
Head starts at 53. Calculate and show with diagram disk head movement.

Next cylinder to be serviced	Total head movement for request
65	12
67	2
37	30
14	23
98	84
122	24
124	2
183	59
total	**236**



SSTF

Consider a disk queue with I/O requests on following cylinders in their arriving order: 54, 97, 73, 128, 15, 44, 110, 34, 45

Head starts at 23. Calculate and show with diagram disk head movement.

Next cylinder to be serviced	Total head movement for request
15	8
34	19
44	10
45	1
54	9
73	19
97	24
110	13
128	**18** **Total = 121**

Total head movement = 121

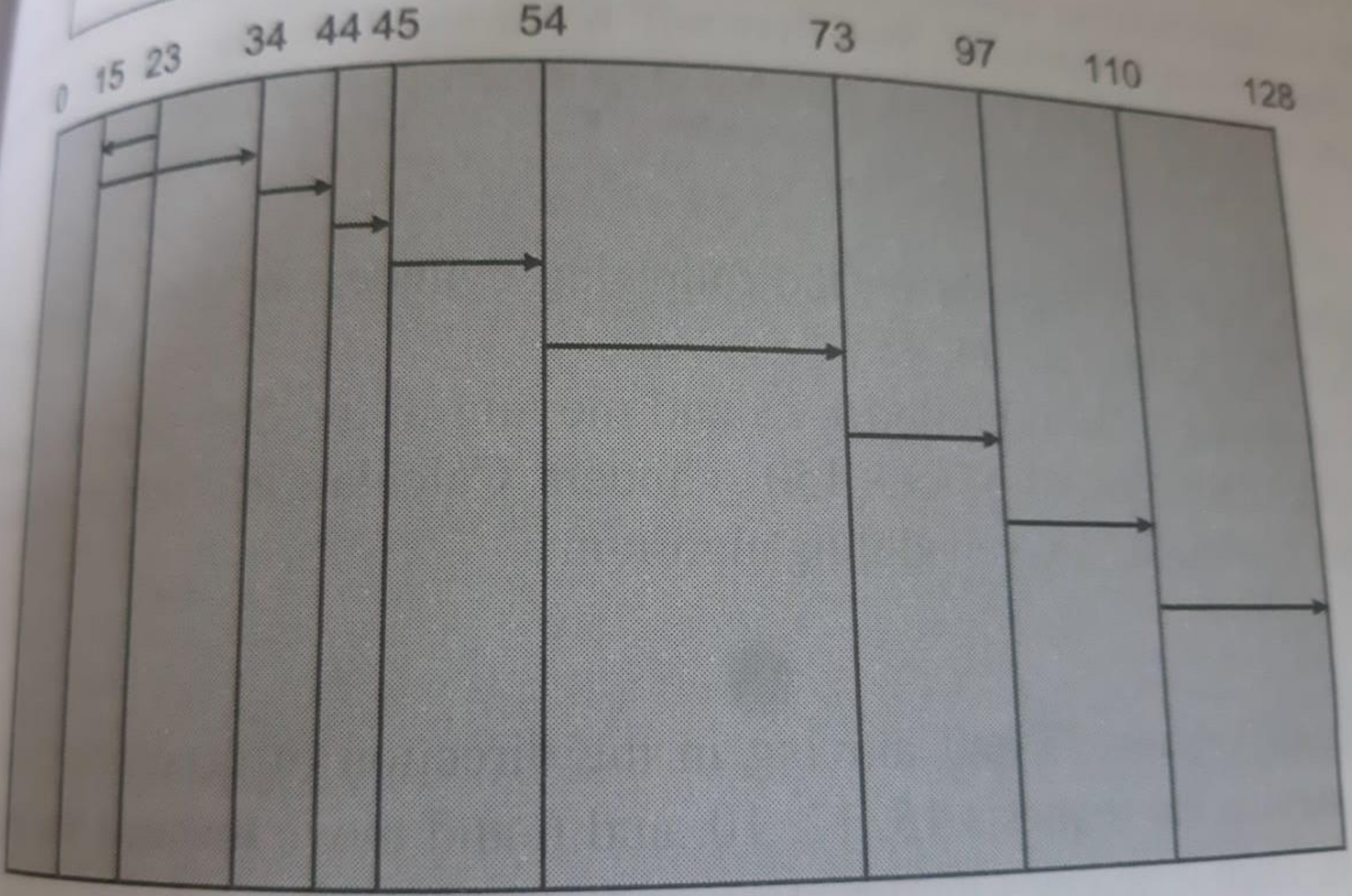
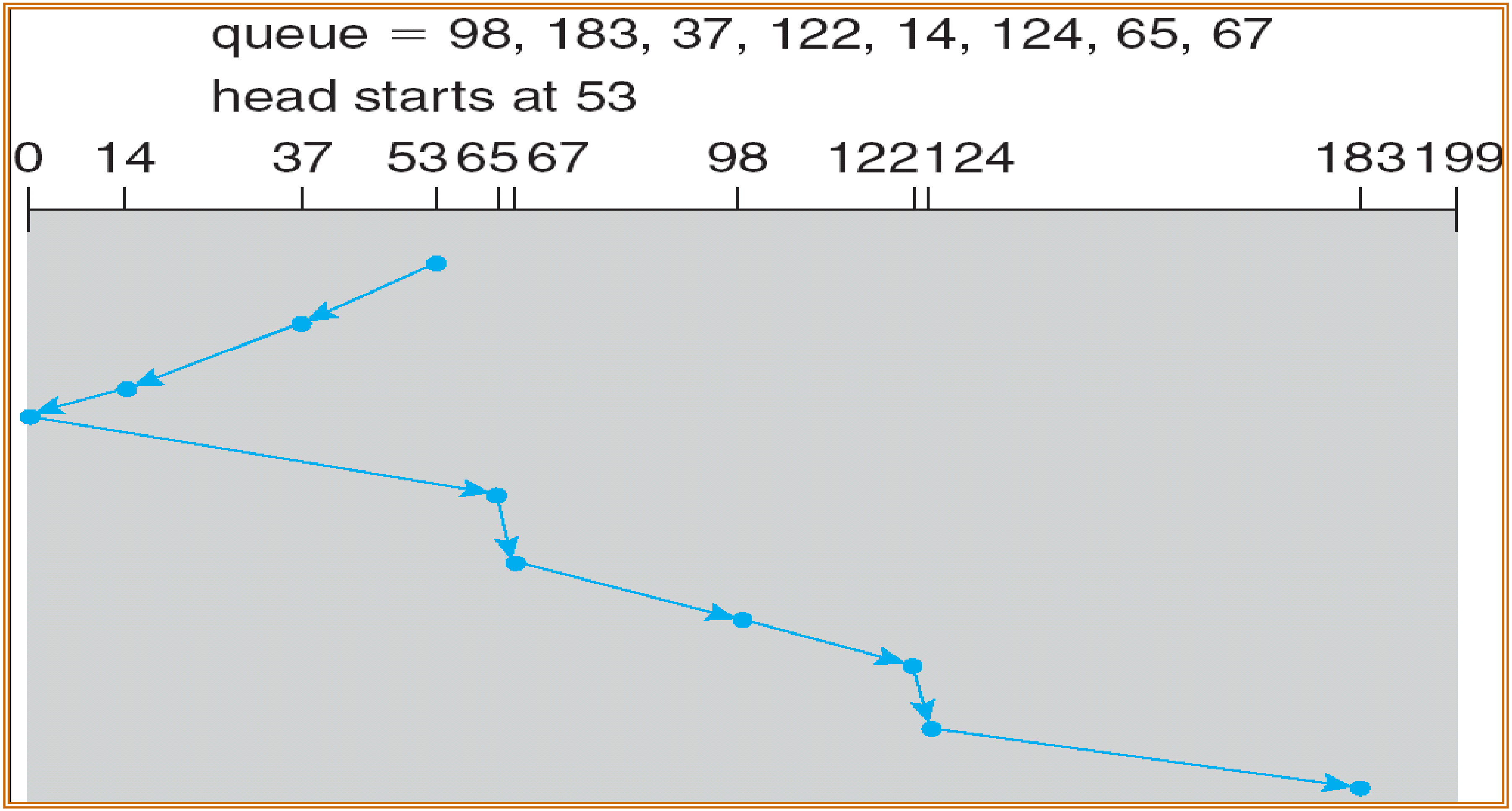


Fig. 15.3 SSTF disk scheduling for Example 15.2

SCAN

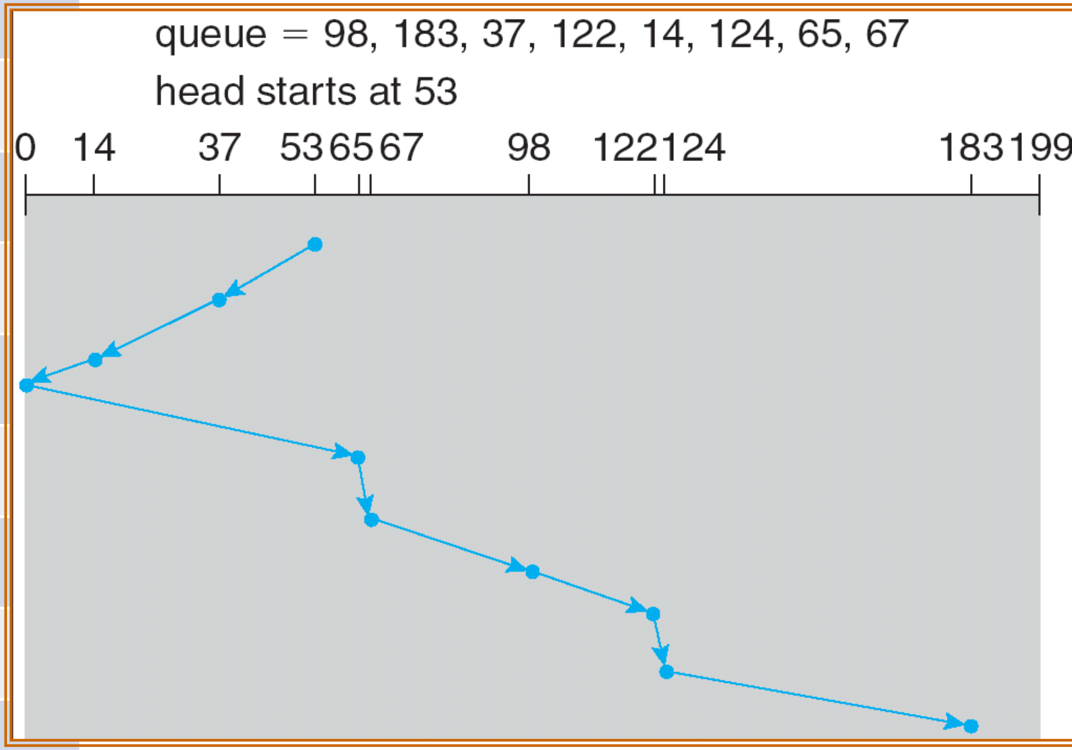
- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- Sometimes called the *elevator algorithm*.
- It is popularly known as ELEVATOR algorithm as the head moves back and forth across the disk and services the requests in its path.
- This algorithm is bidirectional as it satisfies all requests until there is no more pending request in that direction, the service direction is then reversed.

SCAN (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

SCAN

Next cylinder to be serviced	Total head movement for request
37	16
14	23
0	14
65	65
67	2
98	31
122	24
124	2
183	59
199	16
Total	**252**



SCAN

Consider a disk queue with I/O requests on following cylinders in their arriving order: 6,10,12, 54, 97, 73, 128, 15, 44, 110, 34, 45 Head starts at 23 moving in direction of decreasing number of cylinders. Calculate and show with diagram disk head movement. The disk consists total 0 to 150 cylinders.

Ans: 173 cylinders

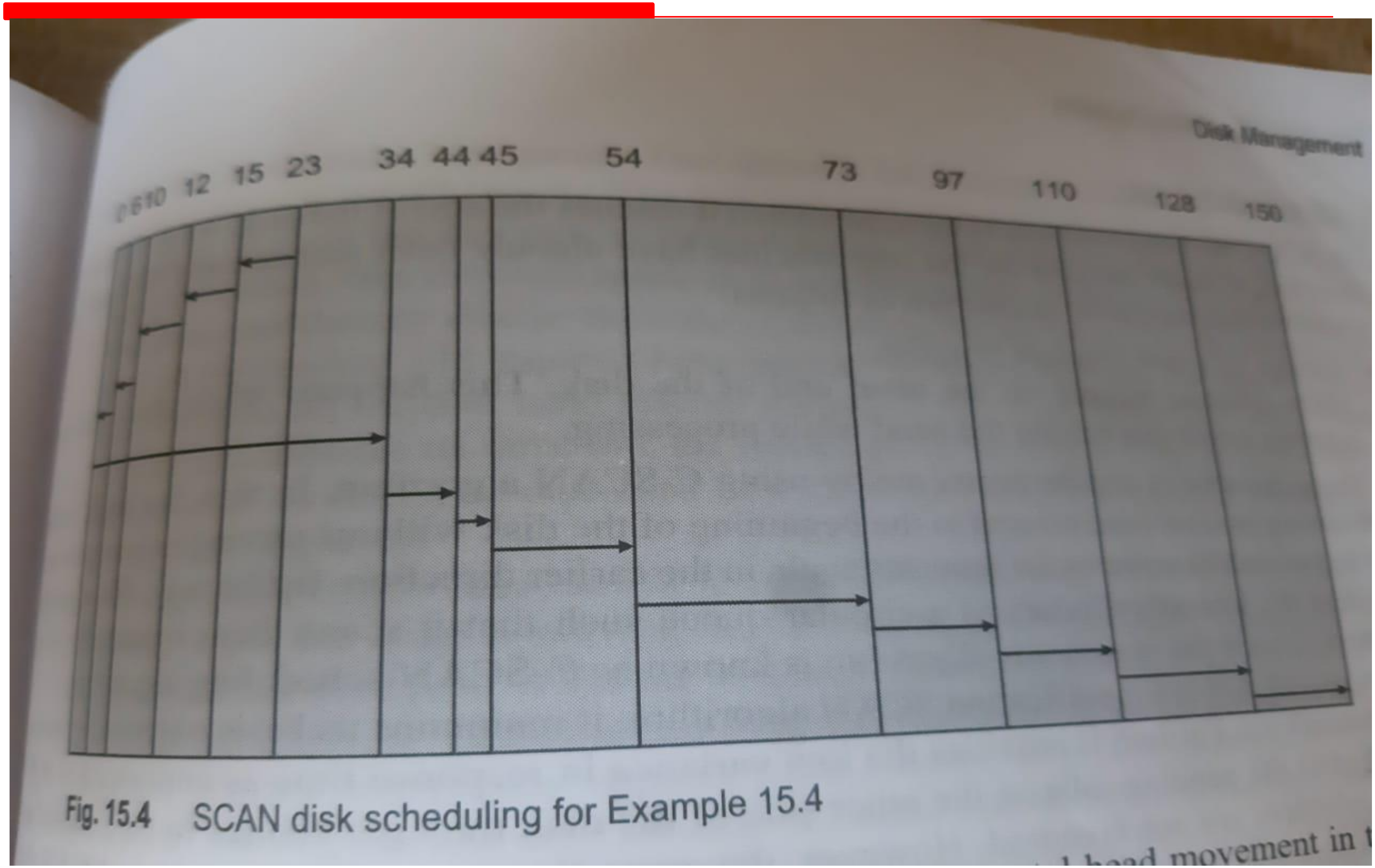


Fig. 15.4 SCAN disk scheduling for Example 15.4

head movement in t

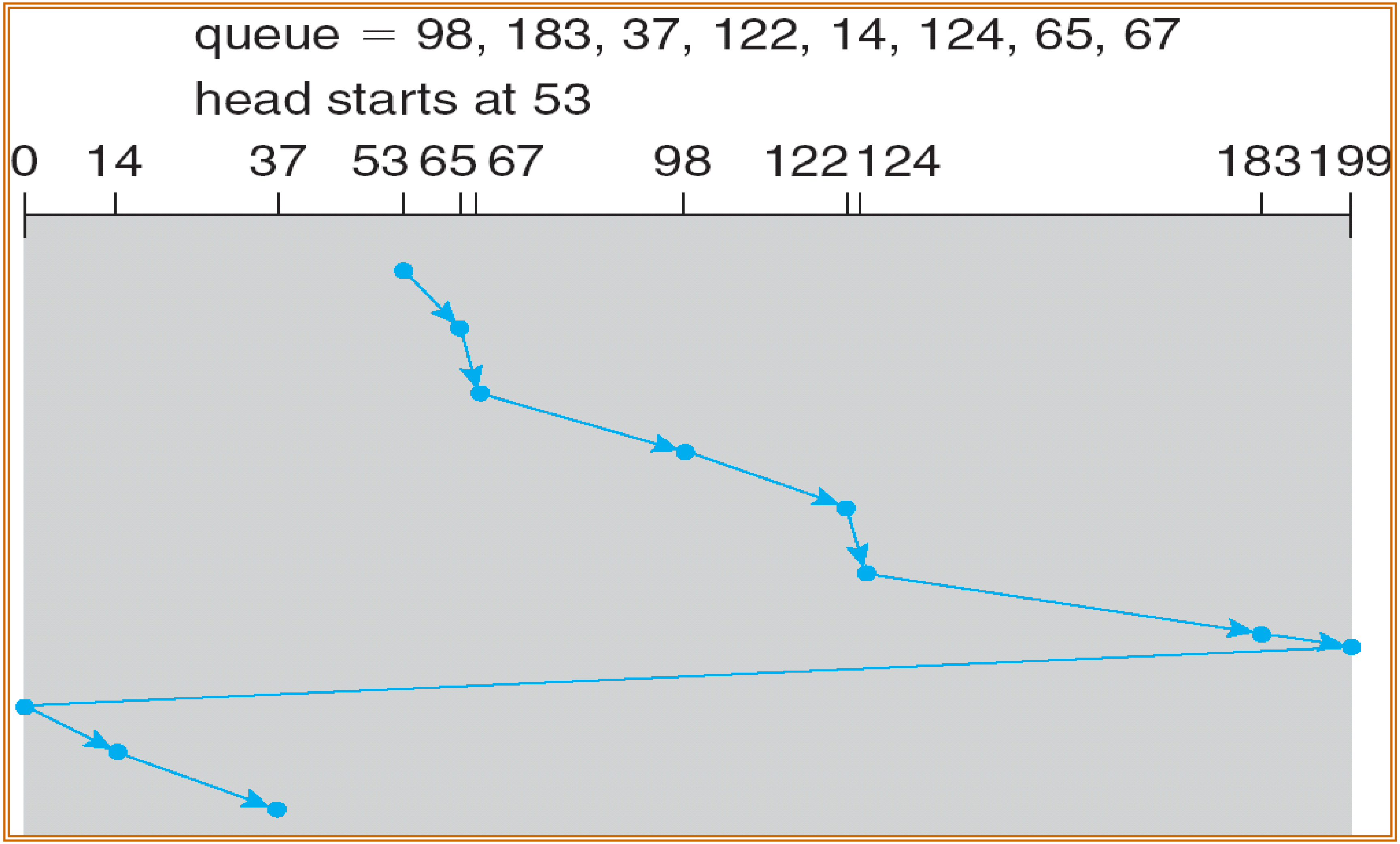
C-SCAN

- Provides a more uniform wait time than SCAN.
- The head moves from one end of the disk to the other servicing requests as it goes.
- When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.

Disk Scheduling Policies

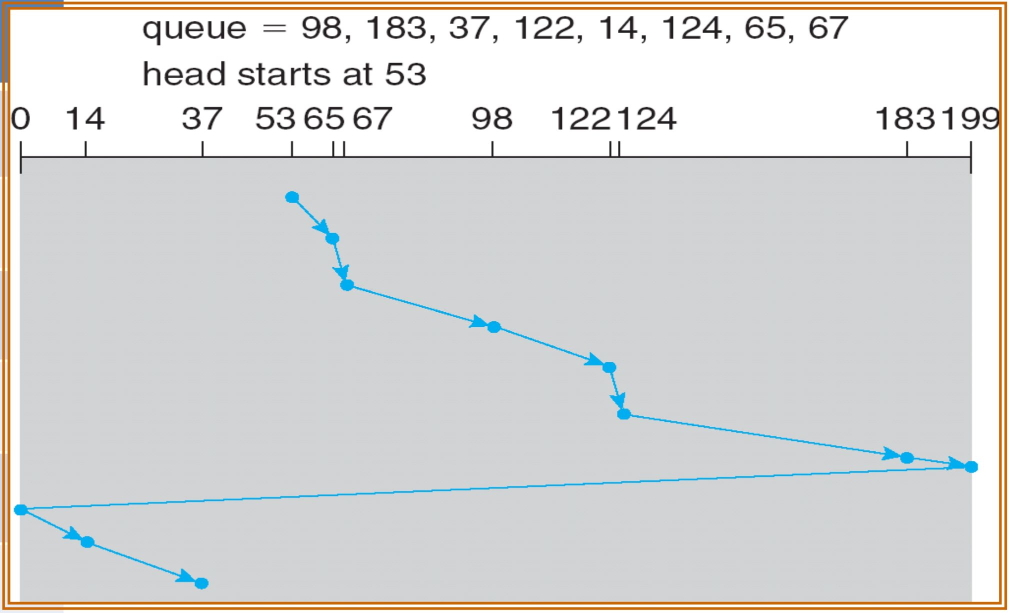
- C-SCAN
  - Restricts scanning to one direction only
  - When the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again

C-SCAN (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-SCAN

Next cylinder to be serviced	Total head movement for request
65	12
67	2
98	31
122	24
124	2
183	59
199	16
0-14	14
37	23
Total	183 cylinders


queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53

C-SCAN

Consider a disk queue with I/O requests on following cylinders in their arriving order: 6,10,12, 54, 97, 73, 128, 15, 44, 110, 34, 45 Head starts at 23 moving in direction of decreasing number of cylinders. Calculate and show with diagram disk head movement. The disk consists total 0 to 150 cylinders.

Ans: 139 cylinders

LOOK

- In scan disk head moves to last cylinder and then reverses its direction, the head increases the delay. So modification is made in this algorithm such that the head moves in one direction only till the last request in that direction.
- This is called LOOK algorithm.
- This is bidirectional algorithm like SCAN
- The disk head looks ahead in direction of movement

LOOK

Consider a disk queue with I/O requests on following cylinders in their arriving order: 6,10,12, 54, 97, 73, 128, 15, 44, 110, 34, 45 Head starts at 23 moving in direction of decreasing number of cylinders. Calculate and show with diagram disk head movement. The disk consists total 0 to 150 cylinders.

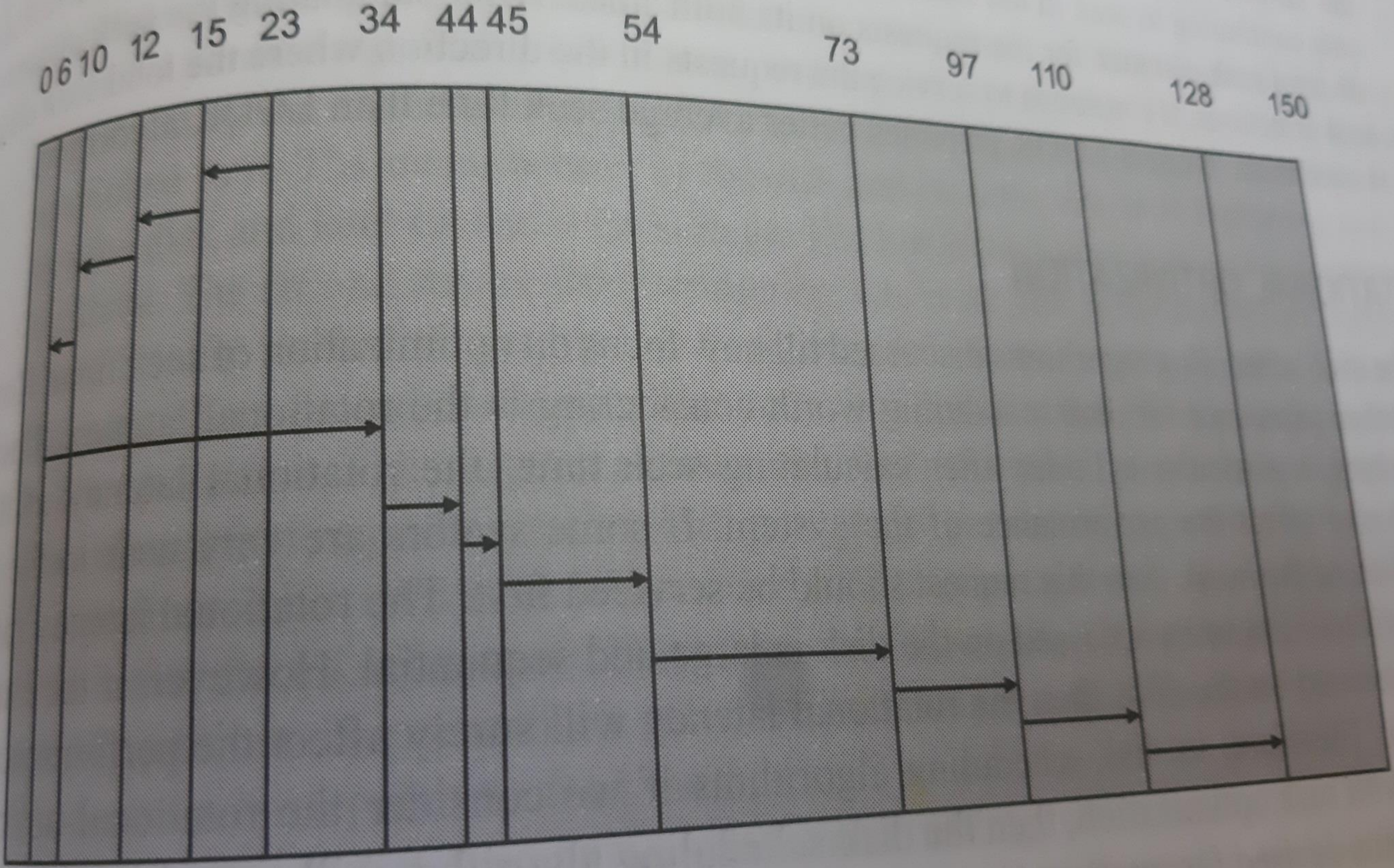


Fig. 15.13 LOOK disk scheduling for Example 15.9

Ans: 139 cylinders

LOOK

I/O REQUESTS: 98, 183, 37, 122, 14, 124, 65, 67
Head starts at 53 moving towards decreasing number of cylinders. Find out total head movement using LOOK disk scheduling algorithm.(Total cylinders 0 to 199).

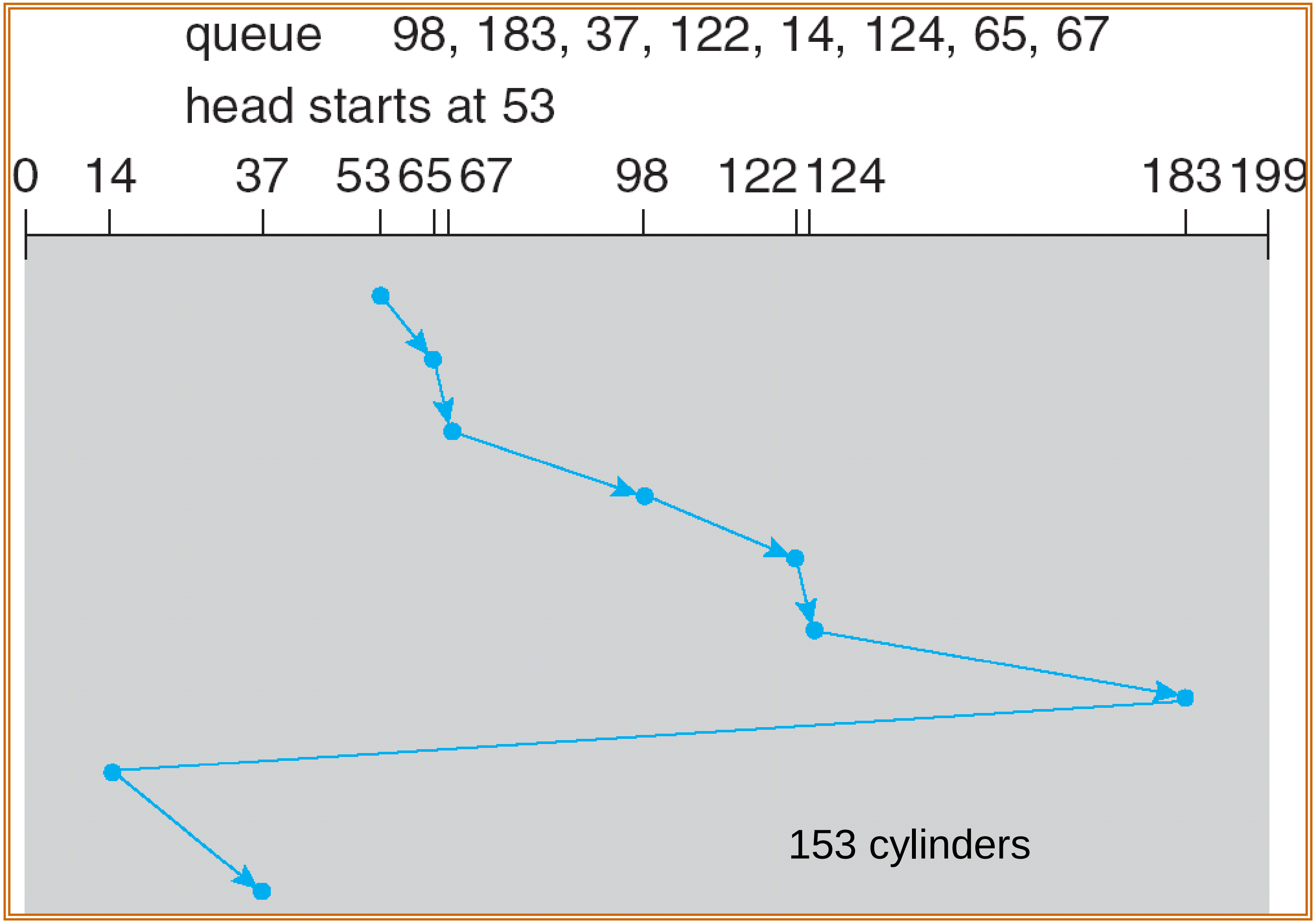
LOOK

Next cylinder to be serviced	Total head movement for request
37	16
14	23
0	14 NA
65	51
67	2
98	31
122	24
124	2
183	59
199	16 NA
Total	**208**

C-LOOK

- Version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.
- This algorithm works only in one direction like C-SCAN.

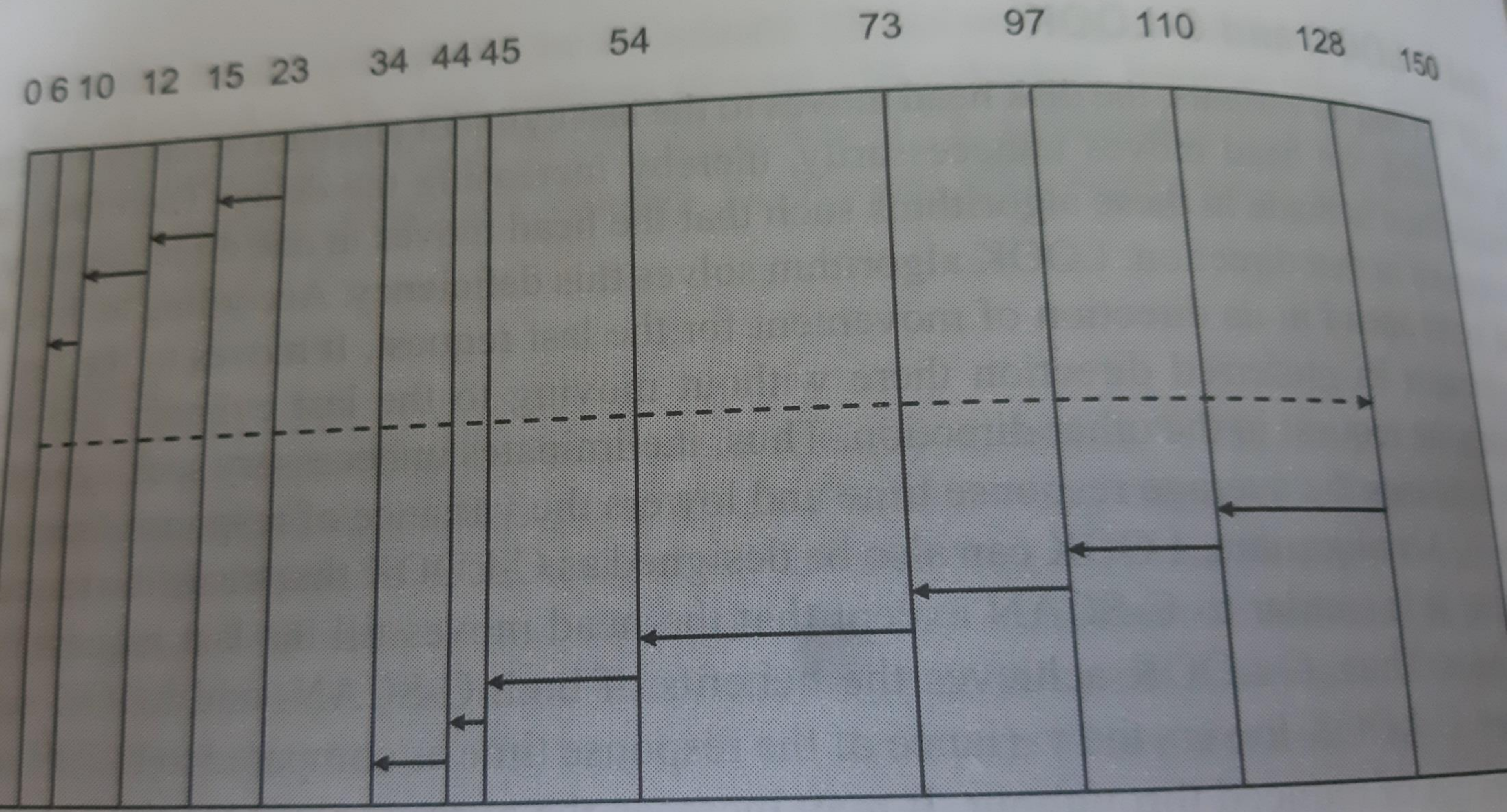
C-LOOK (Cont.)


queue 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199
153 cylinders

C-LOOK

Consider a disk queue with I/O requests on following cylinders in their arriving order: 6,10,12, 54, 97, 73, 128, 15, 44, 110, 34, 45 Head starts at 23 moving in direction of decreasing number of cylinders. Calculate and show with diagram disk head movement. The disk consists total 0 to 150 cylinders.

Ans: 111 cylinders



ig. 15.14 C-LOOK disk scheduling for Example 15.10

11

Various disk scheduling algorithms

Algorithm	Criteria	Pros/cons
FCFS	The requests are scheduled in the order as they have arrived in the disk queue.	Fair Low throughput High average response time High seek time
SSTF	The request whose seek time is shortest will be executed first.	Reduced seek time High throughput Starvation Unfair
SCAN	Services the requests in the direction of its head movement and continues until it reaches at the end of the disk	Good average response time High throughput Starvation
C-SCAN	Considers the queue of cylinders in a circular queue such that it starts scanning them in the queue and after reaching at the last cylinder it starts with the first one again.	High throughput Starvation

Various disk scheduling algorithms

LOOK	the disk head looks ahead in its direction of movement for the last request, moves to the last request and changes its preferred direction there only and continues servicing the request.	Reduces seek time High throughput Good average response time
C-LOOK	treats the request queue as a circular queue, i.e. the disk head after reaching to the last request in its direction of movement returns back to the other end of the disk and reverses its direction and continues servicing the requests.	Reduces seek time Good average response time

End of Chapter