

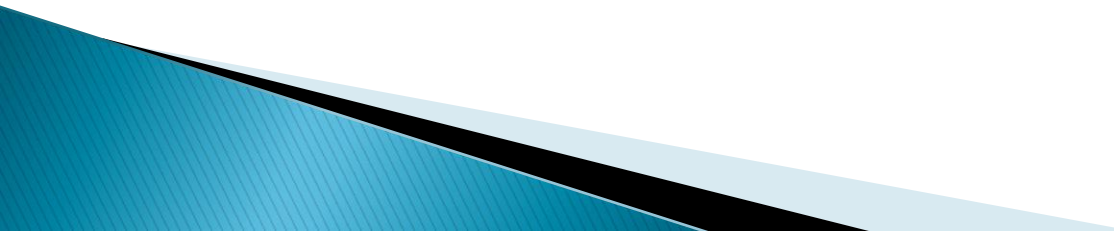
*Operating Systems:
Internals and Design Principles, 6/E*
William Stallings

I/O Management and Disk Scheduling

Dr. Sanam Shahla Rizvi



Contents

- ▶ I/O devices
 - ▶ Organization of I/O function
 - ▶ Operating system design issues
 - ▶ I/O buffering
 - ▶ Disk scheduling
 - ▶ RAID
- 

I/O devices

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
 - Actuators

Categories of I/O Devices

- ▶ Communication
 - Used to communicate with remote devices
 - Digital line drivers
 - Modems

Differences in I/O Devices

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

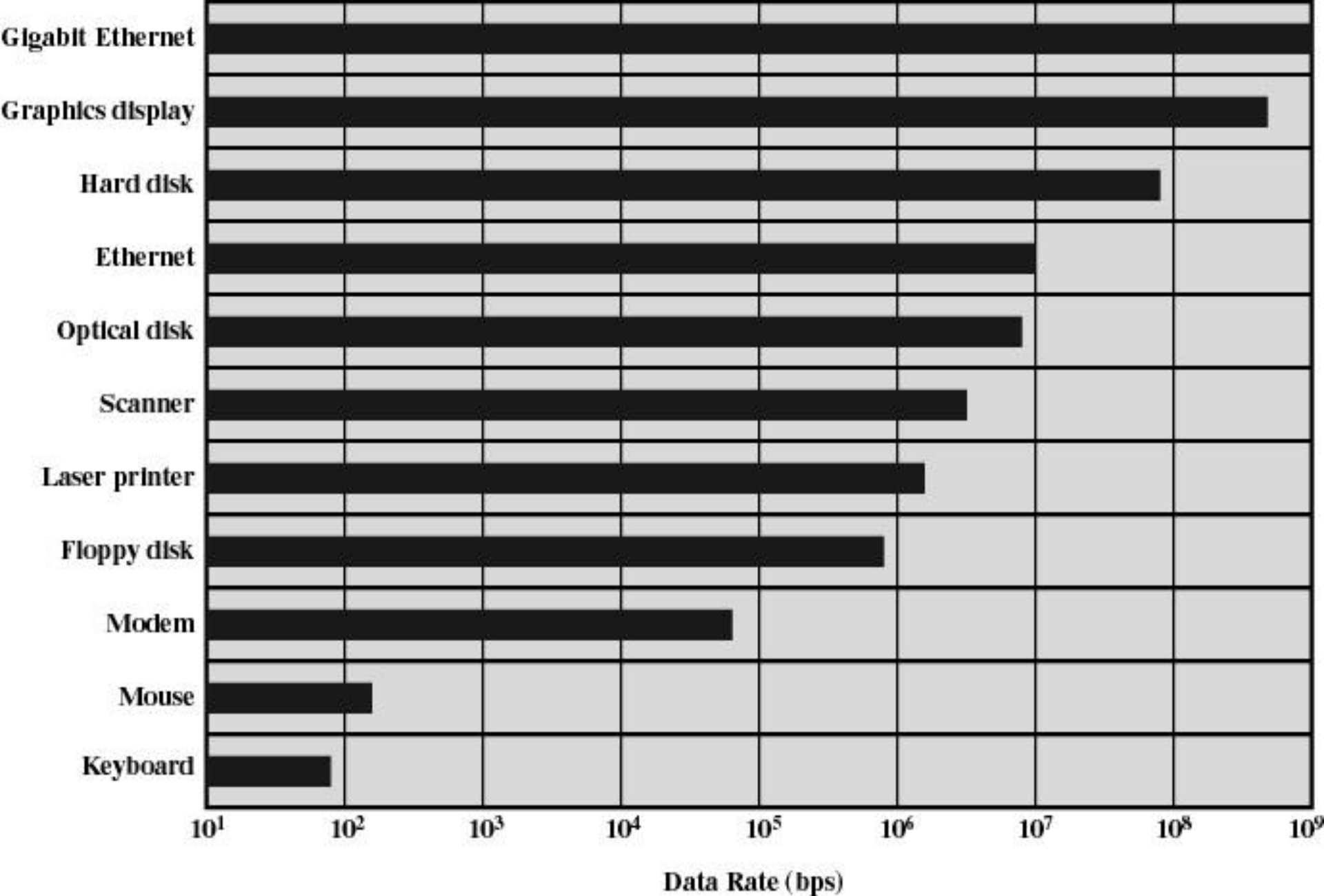


Figure 11.1 Typical I/O Device Data Rates

Differences in I/O Devices

▶ Application

- Disk used to store files requires file-management software
- Disk used to store virtual memory pages needs special hardware and software to support it
- Terminal used by system administrator may have a higher priority

Differences in I/O Devices

- ▶ Complexity of control
 - Printer require simple control interface
 - Disk is much more complex
- ▶ 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

Organization of I/O function



Techniques for Performing I/O

- ▶ Programmed I/O
 - I/O command is issued
 - 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

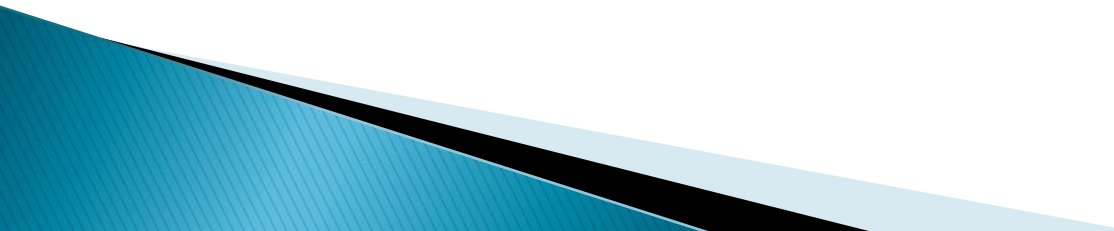
Techniques for Performing I/O

- ▶ Direct Memory Access (DMA)
 - DMA module controls exchange of data between main memory and the I/O device
 - Processor interrupted only after entire block has been transferred

Evolution of the I/O Function

- ▶ Processor directly controls a peripheral device
- ▶ Controller or I/O module is added
 - Processor uses programmed I/O
- ▶ 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 enhanced to become a separate processor
 - Processor specifies sequence of I/O activities
 - Processor interrupts when entire sequence is performed
 - ▶ I/O processor
 - I/O module has its own local memory
 - Its a computer in its own right
 - Minimal processor involvement
- 

Direct Memory Access (DMA)

- ▶ Processor issues a command to DMA module with following information:
 - Read or write request information
 - Address of the I/O device involved, communicated on the data lines
 - Starting location in memory, communicated on the data lines and stored by DMA in its address register
 - Number of words to be read or written, again communicated via the data lines and stored in the data count register

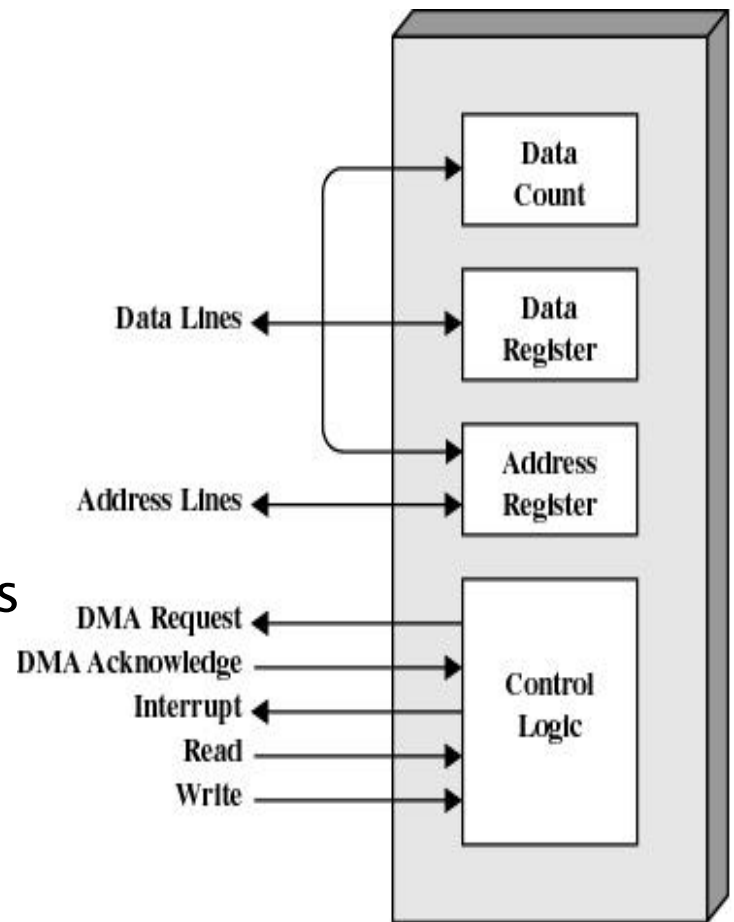


Figure 11.2 Typical DMA Block Diagram

DMA Mechanism

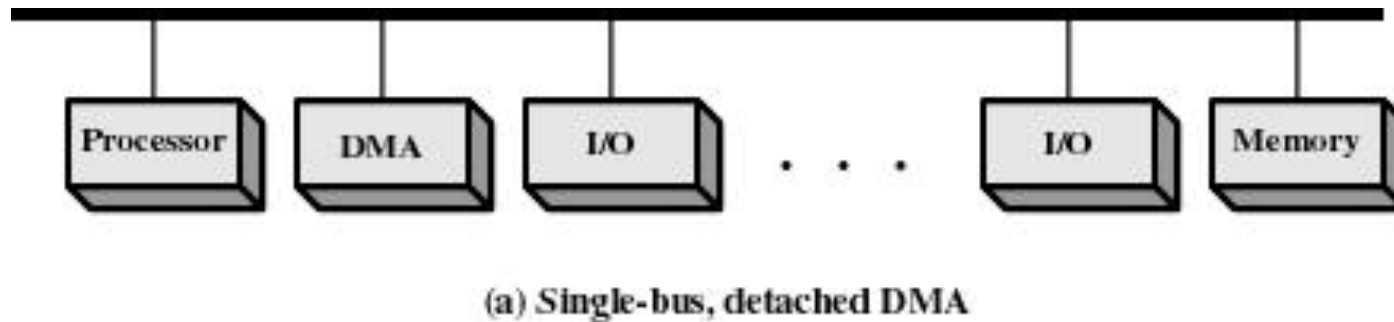
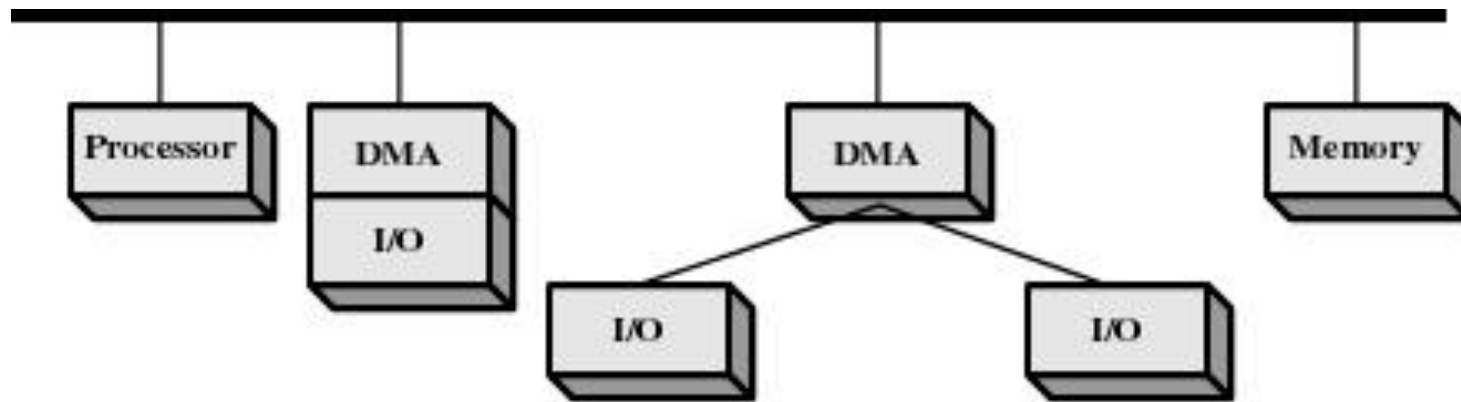


Figure 11.4 Alternative DMA Configurations

DMA Mechanism



(b) Single-bus, Integrated DMA-I/O

Figure 11.4 Alternative DMA Configurations

DMA Mechanism

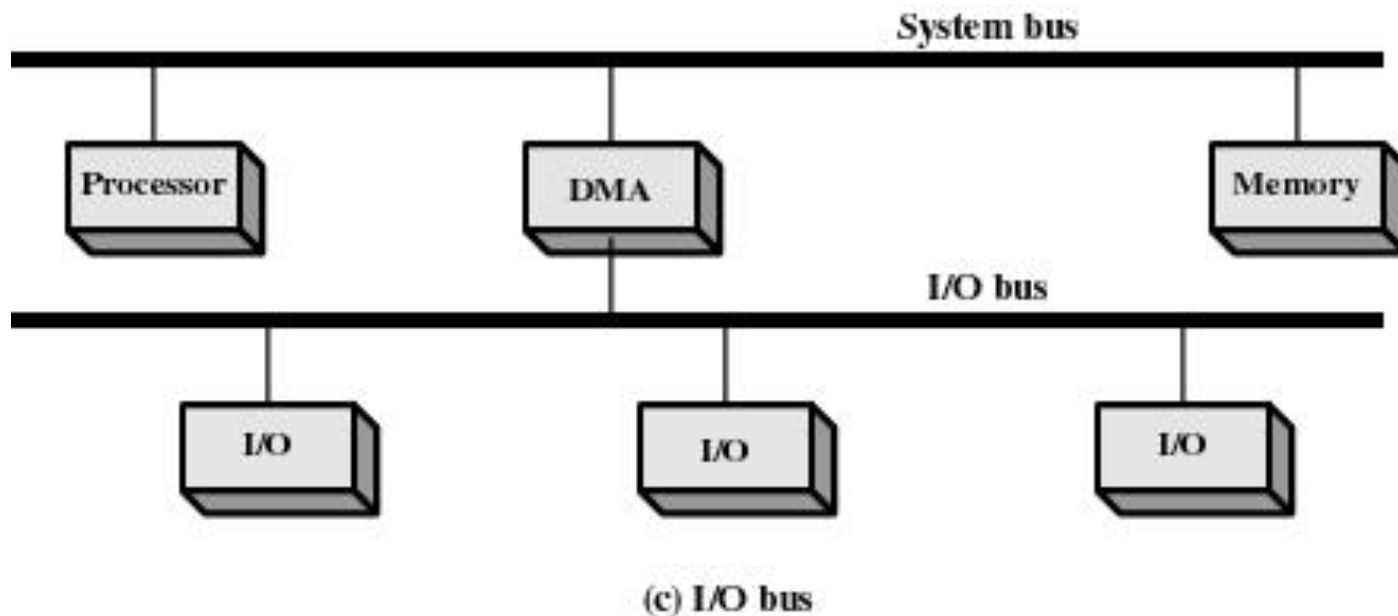


Figure 11.4 Alternative DMA Configurations

Operating System Design Issues



Operating System Design Issues

► Efficiency

- Most I/O devices are extremely slow compared to main memory and processor
- Use of multiprogramming allows for some processes to be waiting on I/O while another process executes
- Swapping is used to bring in additional ready processes, which itself is an I/O operation

Operating System Design Issues

▶ Generality

- Desirable to handle all I/O devices in a uniform manner
- Hide most of the details of device I/O in lower-level routines so that processes and upper levels see devices in general functions such as read, write, open, close, lock, unlock

Logical Structure of I/O Function

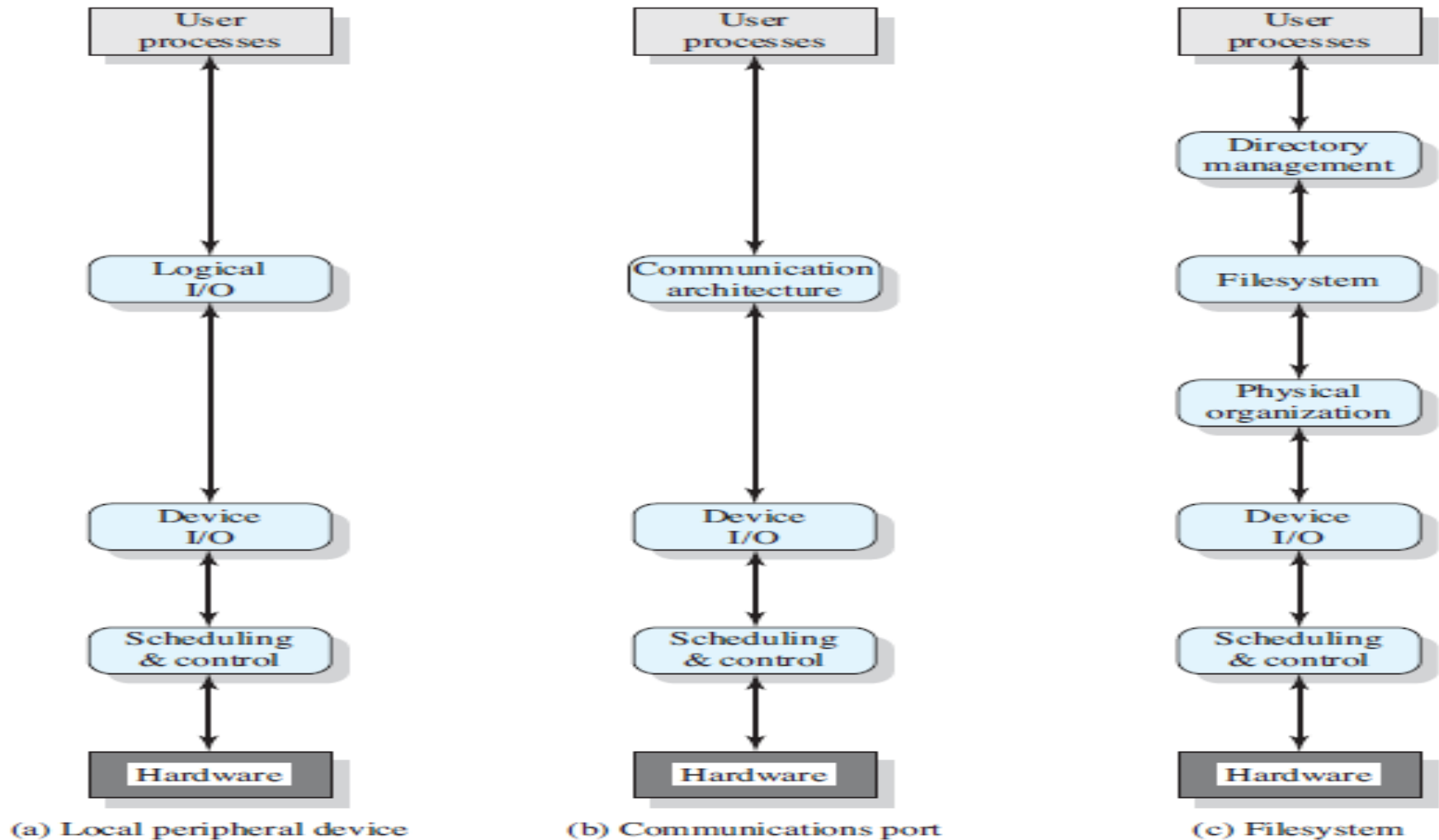


Figure 11.4 A Model of I/O Organization

I/O Buffering

I/O Buffering

- ▶ Reasons for buffering
 - Processes must wait for I/O to complete before proceeding
 - Increase efficiency of OS and performance of individual processes

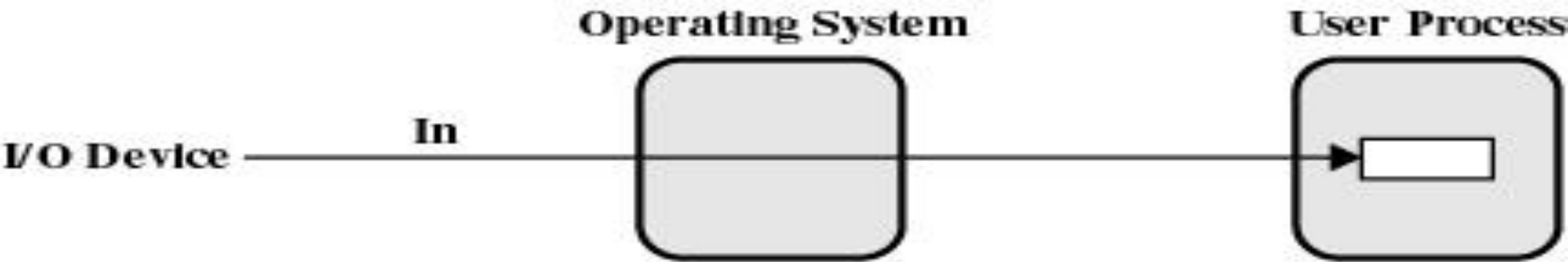
I/O Buffering

- ▶ **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

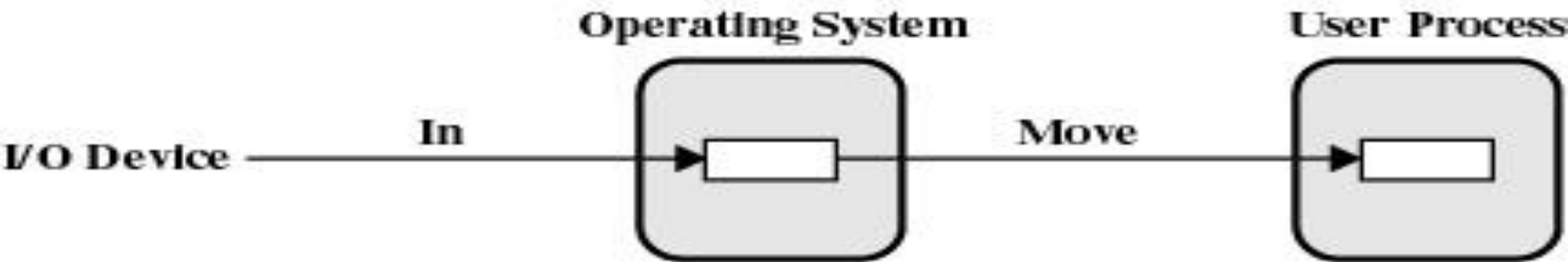
Single Buffer

- ▶ 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



(a) No buffering



(b) Single 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
- Swapping can occur since input is taking place in system memory, not user memory
- Operating system keeps track of assignment of system buffers to user processes

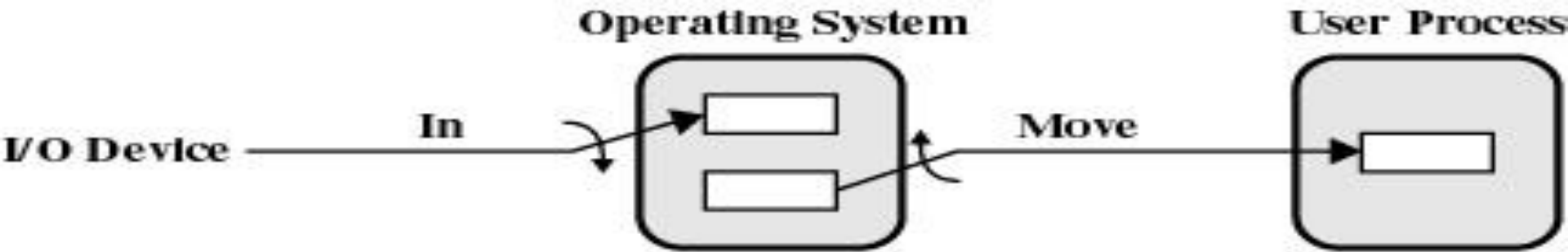
Single Buffer

- ▶ 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

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

I/O Buffering



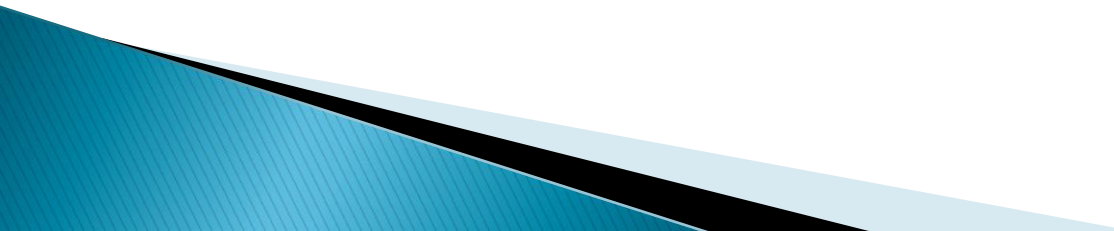
(c) Double buffering



(d) Circular buffering

Figure 11.6 I/O Buffering Schemes (input)

Circular Buffer

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

Disk Scheduling

Disk Performance Parameters

- ▶ Disk I/O operation depends on the:
 - Computer system
 - Operating System
 - Nature of the I/O channel
 - Disk controller hardware

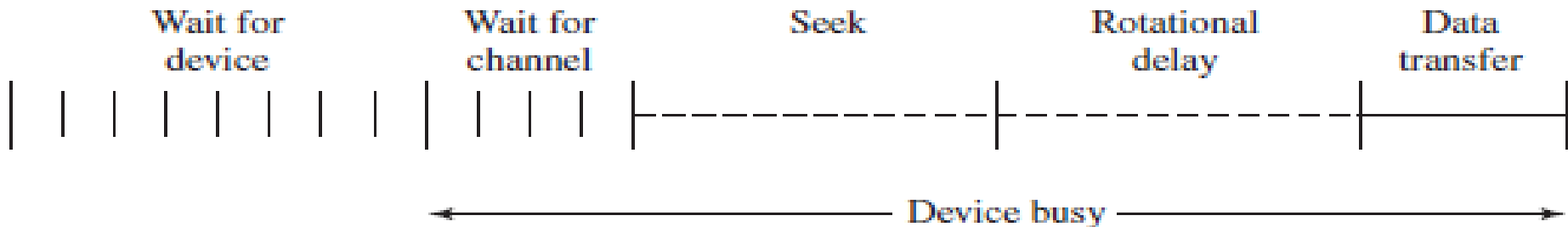


Figure 11.6 Timing of a Disk I/O Transfer

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 to position the head at the track
- ▶ Rotational delay/latency: time to position the head at the beginning of sector
- ▶ Access Time = Seek Time + Rotational delay/latency
- ▶ Transfer Time: time required to transfer of data

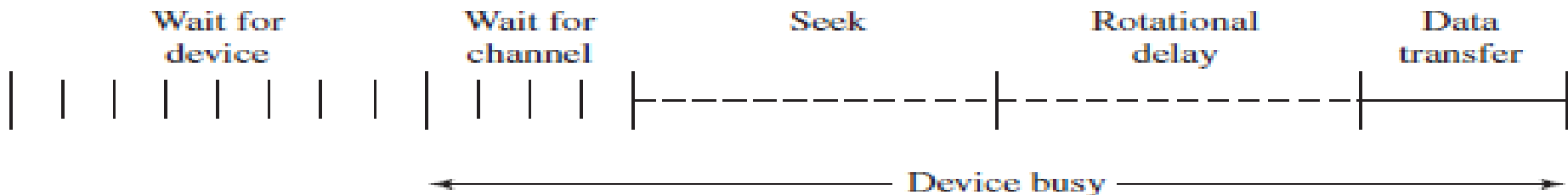


Figure 11.6 Timing of a Disk I/O Transfer

A Timing Comparison

▶ Assume:

- Average seek time = 4ms
- Average rotational time = 4ms
- 512 bytes per sector with 500 sectors per track
- Read File of 2500 sectors for a total of 1.28MBs
- File is *sequentially organized* in 5 tracks

▶ To read the first track:

- Average seek = 4ms
- Rotational delay = 4ms
- Read 500 sectors = 8ms
- Total = 16ms

▶ Total time = $16 + (4 \times 12) = 64\text{ms} = 0.064\text{seconds}$

A Timing Comparision

▶ Assume:

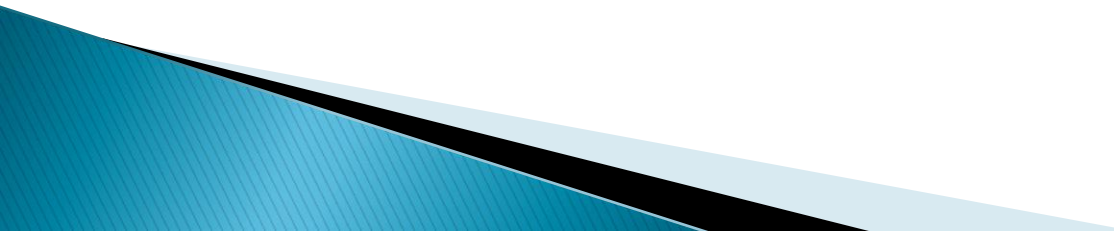
- Average seek time = 4ms
- Average rotational time = 4ms
- 512 bytes per sector with 500 sectors per track
- Read File of 2500 sectors for a total of 1.28MBs
- File is *randomly distributed* in different tracks

▶ For each sector:

- Average seek = 4ms
- Rotational delay = 4ms
- Read 1 sector = 0.016ms
- Total = 8.016ms

▶ Total time = $2500 \times 8.016 = 20,040\text{ms} = 20.04\text{seconds}$

Disk Performance Parameters

- ▶ 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 Policies

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

Disk Scheduling Policies

▶ Priority

- Goal is not to optimize disk use but to meet other objectives
- Short batch jobs may have higher priority
- Provide good interactive response time

Disk Scheduling Policies

- ▶ Last-in, first-out
 - Good for transaction processing systems
 - The device is given to the most recent user in order to have little arm movement
 - Possibility of starvation since a job may never regain the head of the line

Disk Scheduling Policies

- ▶ Shortest Service Time First
 - Select the disk I/O request that requires the least movement of the disk arm from its current position
 - Always choose the minimum seek time

Disk Scheduling Policies

▶ SCAN

- Arm moves in one direction only, satisfying all outstanding requests until it reaches the last track in that direction
- Direction is reversed

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

Disk Scheduling Policies

▶ N-step-SCAN

- Segments the disk request queue into subqueues of length N
- Subqueues are process one at a time, using SCAN
- New requests added to other queue when queue is processed

▶ FSCAN

- Two queues
- One queue is empty for new request

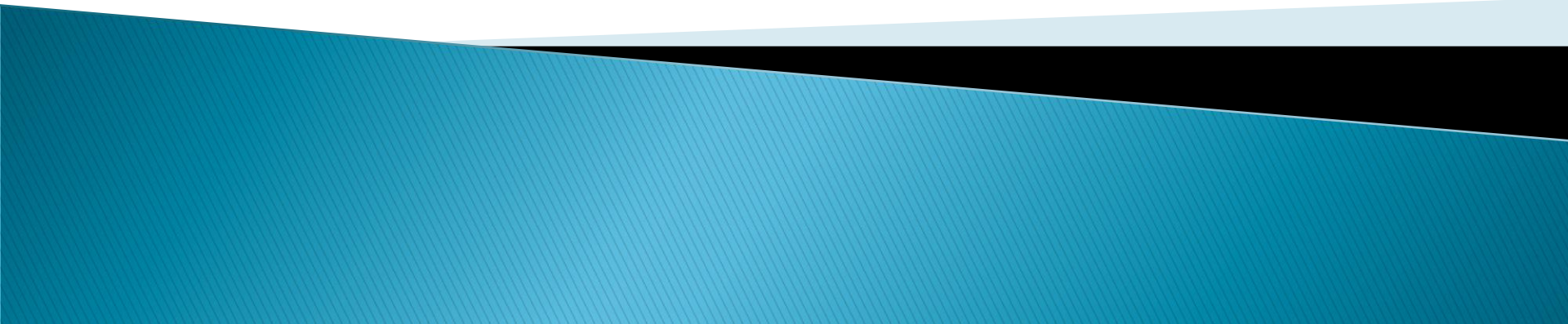
Table 11.2 Comparison of Disk Scheduling Algorithms

(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed	Next track accessed	Number of tracks traversed
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	Average seek length	27.5	Average seek length	27.8	Average seek length	35.8

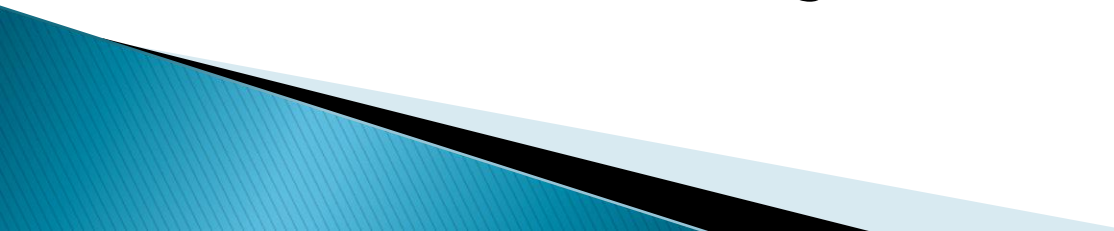
Table 11.3 Disk Scheduling Algorithms

Name	Description	Remarks
Selection according to requestor		
RSS	Random scheduling	For analysis and simulation
FIFO	First in first out	Fairest of them all
PRI	Priority by process	Control outside of disk queue management
LIFO	Last in first out	Maximize locality and resource utilization
Selection according to requested item		
SSTF	Shortest service time first	High utilization, small queues
SCAN	Back and forth over disk	Better service distribution
C-SCAN	One way with fast return	Lower service variability
N-step-SCAN	SCAN of N records at a time	Service guarantee
FSCAN	N-step-SCAN with N = queue size at beginning of SCAN cycle	Load sensitive

Redundant Array of Independent Disks (RAID)

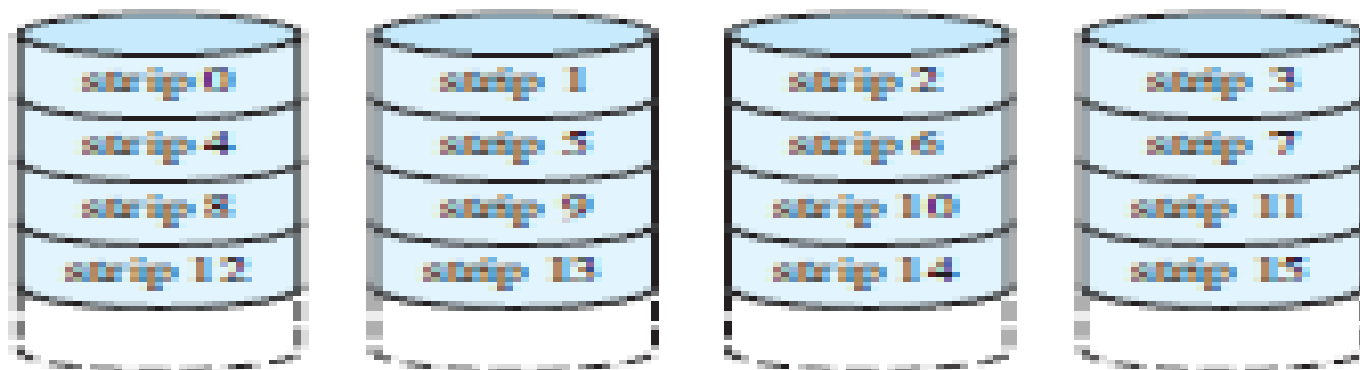


RAID

- ▶ Redundant Array of Independent Disks (RAID)
 - ▶ idea: replace large-capacity disks with multiple smaller-capacity drives to improve the I/O performance
 - ▶ RAID is a set of physical disk drives viewed by the OS as a single logical drive
 - ▶ data are distributed across physical drives in a way that enables simultaneous access to data from multiple drives
 - ▶ redundant disk capacity is used to compensate the increase in the probability of failure due to multiple drives
 - ▶ size RAID levels (design architectures)
- 

RAID Level 0

- ▶ does not include redundancy
- ▶ data is striped across the available disks
 - disk is divided into strips
 - strips are mapped round-robin to consecutive disks
 - a set of consecutive strips that map exactly one strip to each array member is called stripe



(a) RAID 0 (non-redundant)

RAID Level 1

- ▶ redundancy achieved by duplicating all the data
- ▶ each logical disk is mapped to two separate physical disks so that every disk has a mirror disk that contains the same data
 - a read can be serviced by either of the two disks that contains the requested data (improved performance over RAID 0 if reads dominate)
 - a write request must be done on both disks but can be done in parallel
 - recovery is simple but cost is high



(b) RAID 1 (mirrored)

RAID Levels 2 and 3

- ▶ parallel access: all disks participate in every I/O request
- ▶ small strips (byte or word size)
- ▶ RAID 2: error correcting code (Hamming) is calculated across corresponding bits on each data disk and stored on $\log(\text{data})$ parity disks; necessary only if error rate is high
- ▶ RAID 3: a single redundant disk which keeps the parity bit
- ▶ in the event of failure, data can be reconstructed but only one request at the time can be satisfied



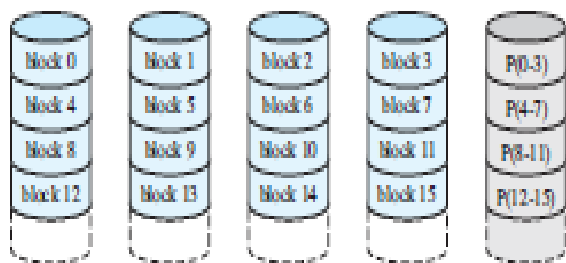
(c) RAID 2 (redundancy through Hamming code)



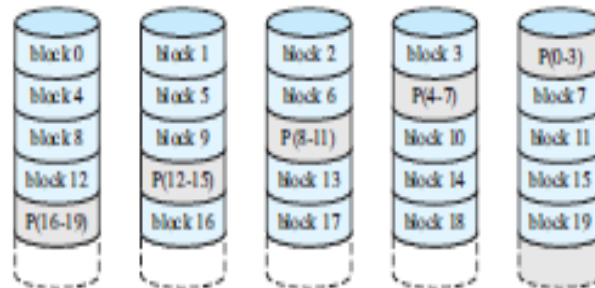
(d) RAID 3 (bit-interleaved parity)

RAID Levels 4, 5 and 6

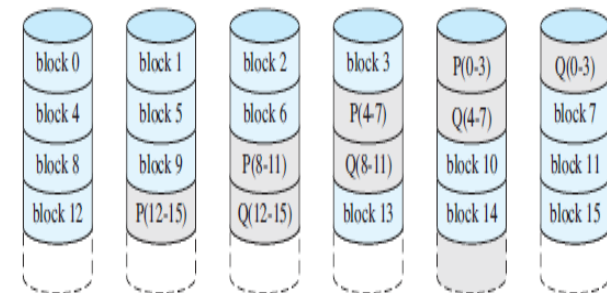
- ▶ independent access: each disk operates independently, multiple I/O request can be satisfied in parallel
- ▶ large strips (block size)
- ▶ RAID 4: parity strips are stored in separate disk
- ▶ RAID 5 & 6: parity strips are distributed across all disks
- ▶ RAID 6: backup disk is used for parity strips for more reliability



(e) RAID 4 (Block-level parity)



(f) RAID 5 (Block-level distributed parity)



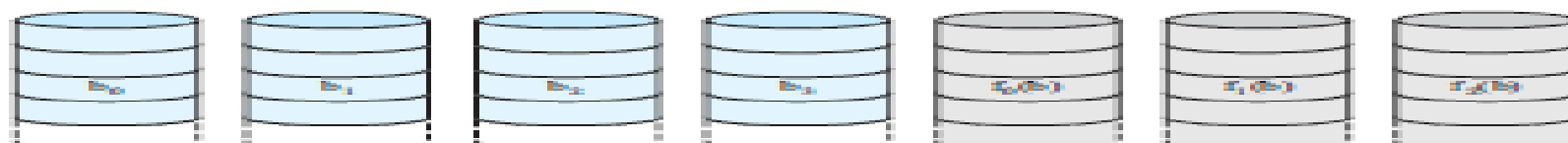
(g) RAID 6 (dual redundancy)



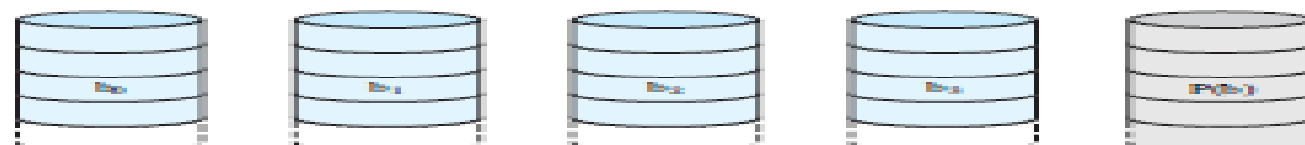
(a) RAID 0 (stripe diskset)



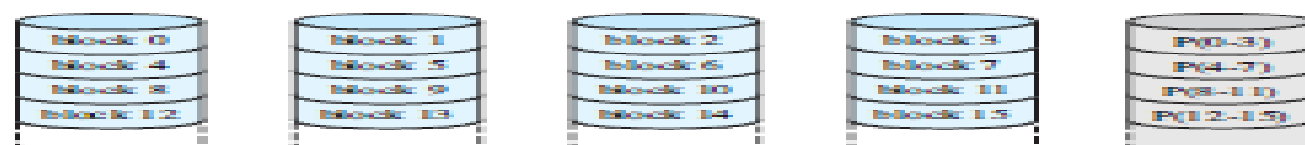
(b) RAID 1 (mirrored)



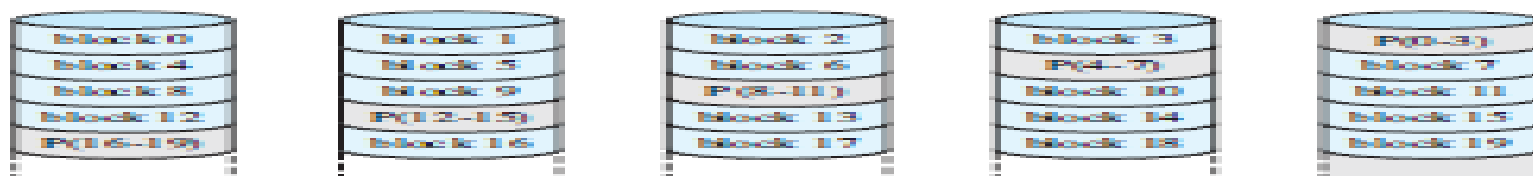
(c) RAID 2 (redundancy through Hamming code)



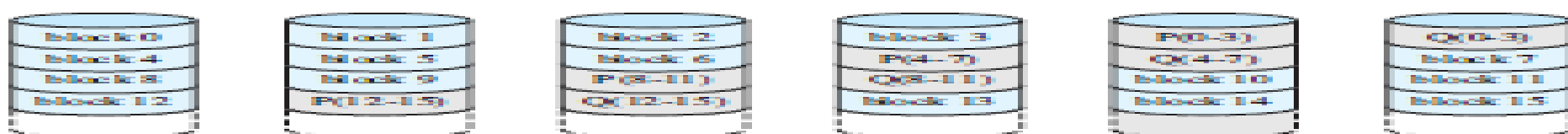
(d) RAID 3 (bit-interleaved parity)



(e) RAID 4 (block-level parity)



(f) RAID 5 (block-level distributed parity)



(g) RAID 6 (dual redundancy)