I/O management and disk scheduling

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

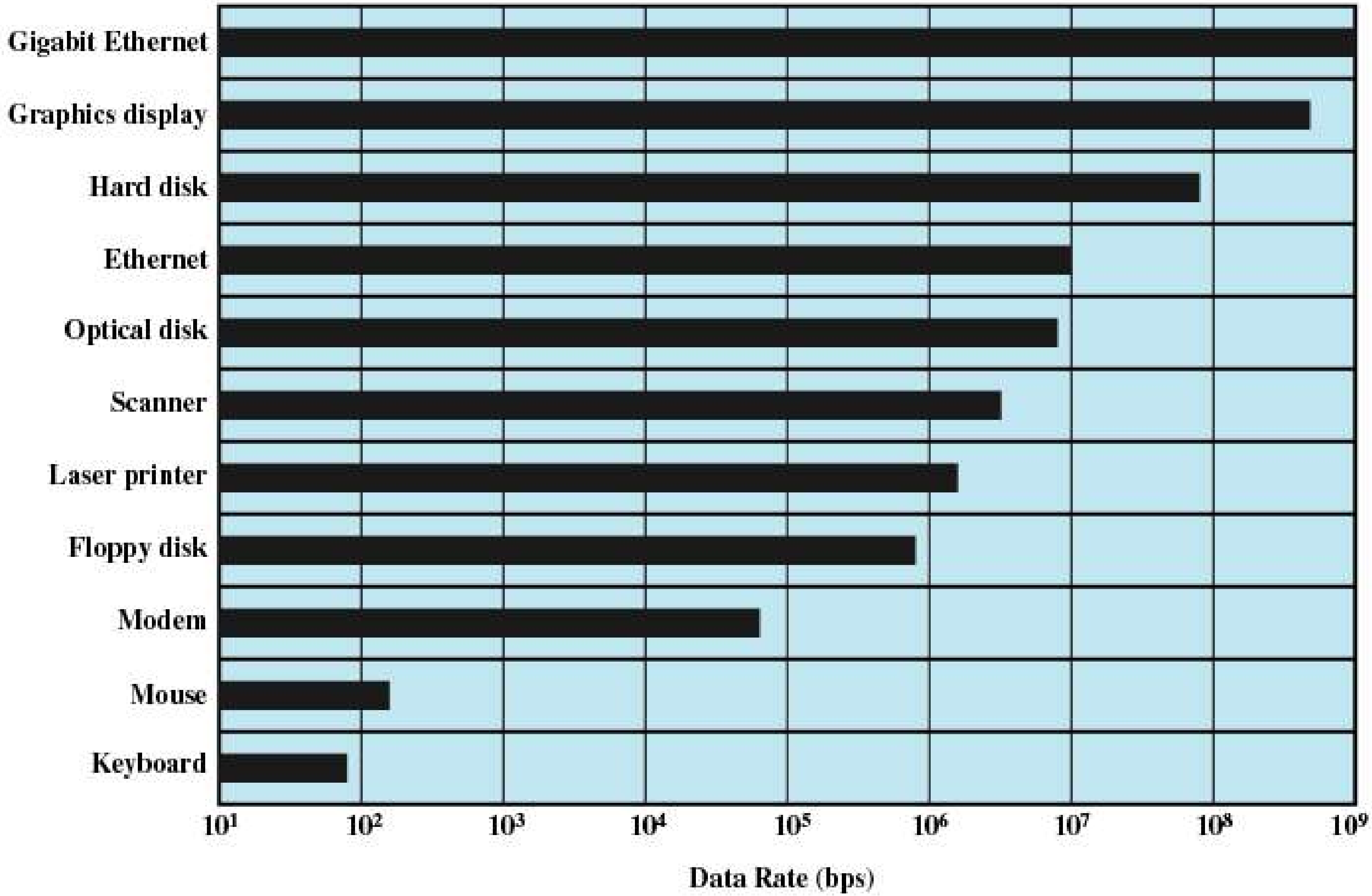
- Machine readable
  - Used to communicate with electronic equipment
  - Disk and tape drives
  - Sensors
  - Controllers
  - Actuators

Categories of I/O Devices

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

Differences in I/O Devices

- May be differences of several orders of magnitude between the data transfer 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
- 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
    - missed tcp segments may be retransmitted
    - disk errors are usually unrecoverable

blocks and streams

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

Performing I/O

Techniques

- Programmed
  - busy-waiting
- Interrupt-driven
- DMA

Architectures

- direct control of device
- controller + programmed I/O
- controller + interrupt-driven
- controller + DMA
- I/O processor

Operating System Design Issues

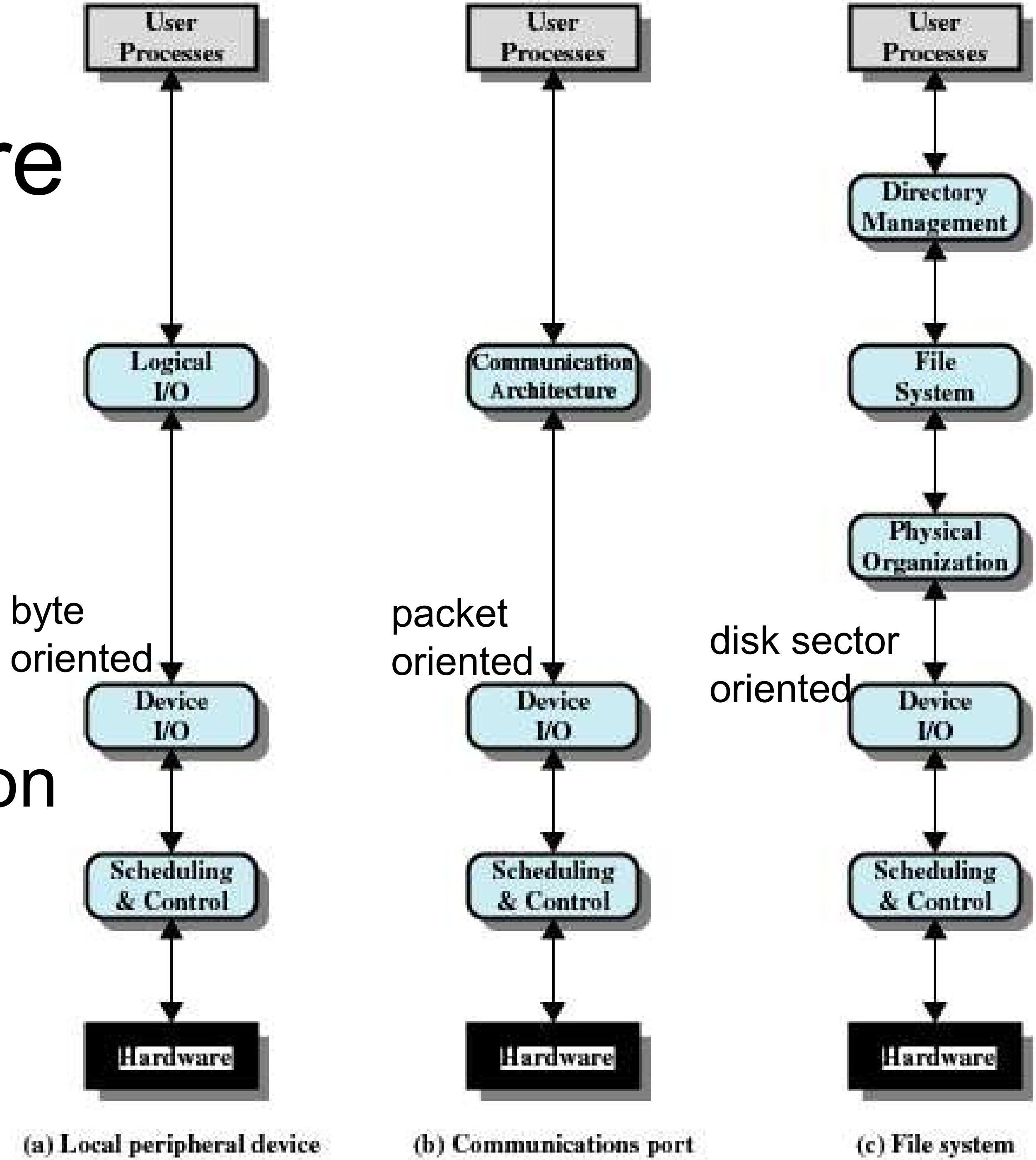
- Efficiency
  - Most I/O devices extremely slow compared to main memory
  - Use of multiprogramming allows for some processes to be waiting on I/O while another process executes
  - I/O cannot keep up with processor speed
  - Swapping is used to bring in additional Ready processes
    - ... and this requires further I/O operations

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
    - upper levels use abstract primitives: read, write, open, close, lock, unlock

I/O architecture models

- logical I/O
  - e.g. open, read, write, close
- communication architecture
  - e.g. TCP/IP



I/O Buffering

- Reasons for buffering
  - write: processes must wait for I/O to complete before proceeding
  - read: optimization is possible
    - read haead
  - without buffering pages destination of I/O must remain in main memory during I/O
    - no full swap out is possible!

Single Buffer

- Operating system assigns a buffer in system memory for an I/O request
- Block-oriented
  - block input transfers are directed to the buffer
  - buffer content is moved to user space when requested by the process
  - Another block is moved into the buffer
    - Read ahead

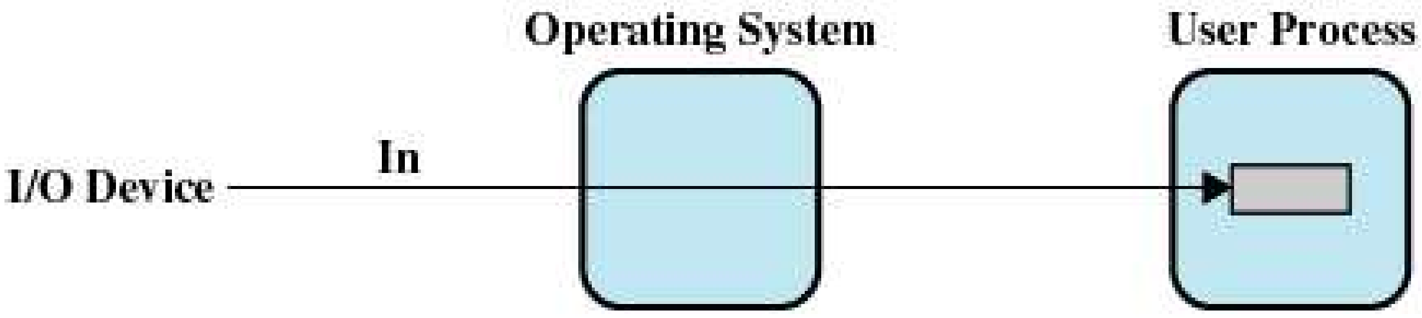
Single Buffer

- 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

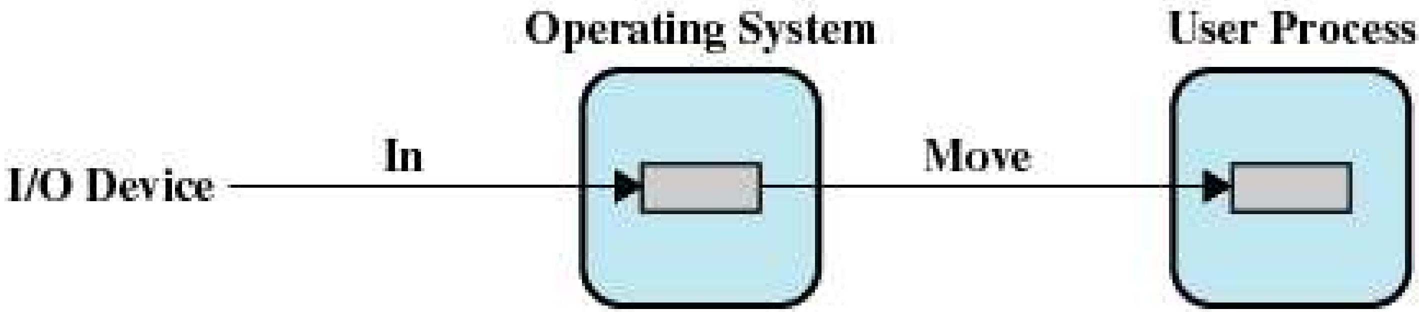
Single Buffer

- Stream-oriented
  - streams can be segmented in fixed or variable size chunks
  - For terminals: one 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



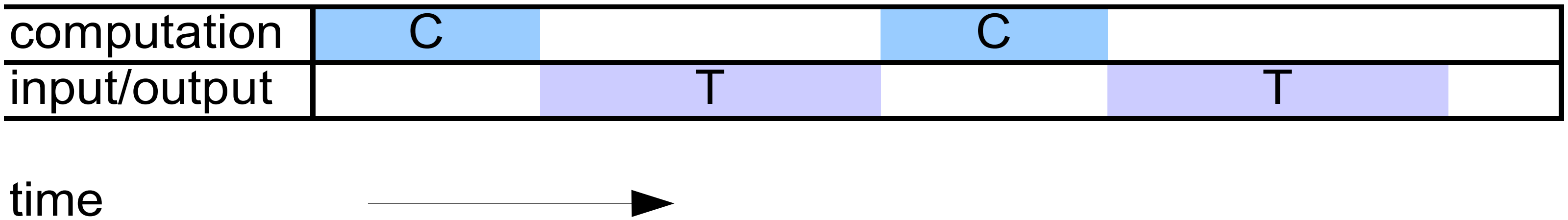
(a) No buffering



(b) Single buffering

no buffer

- T: input (output) time for one block
- C: computation time after input (input)



no buffer

Total time for each block:

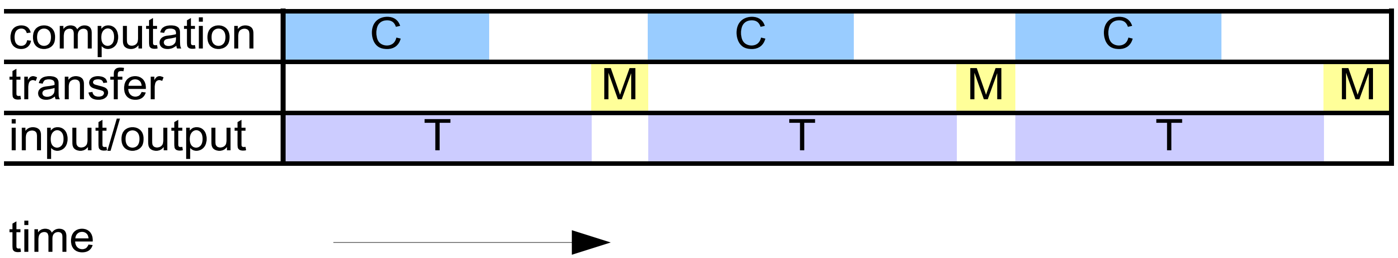
- T+C
  - input: we suppose process cannot read while processing the previous block
  - output: we suppose process cannot compute while is waiting for output to finish.

question: are these assumptions always valid?

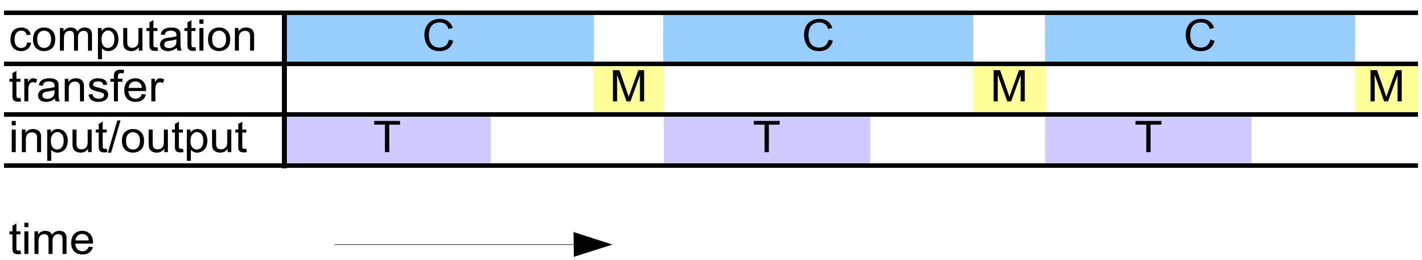
single buffer (input)

- we suppose the operating system can read ahead
  - if T>C the process waits for the buffer to be filled
  - if T<C the OS waits for the buffer to be free

- T>C



- T<C

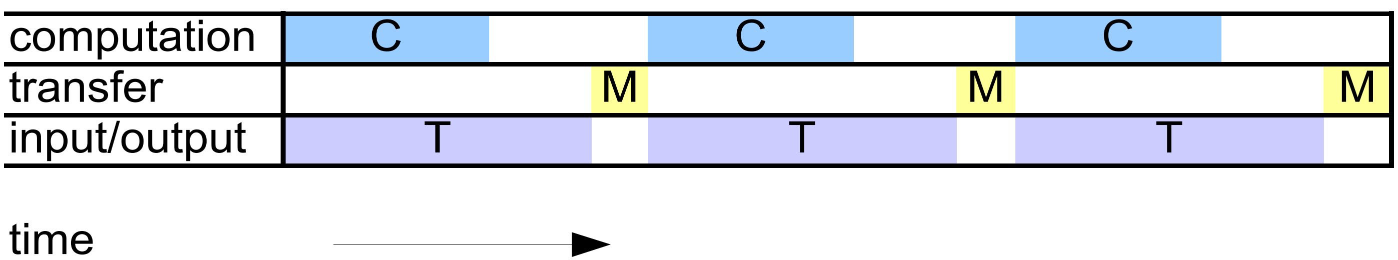


- max[T,C]+M
  - M: time to move data from system buffer to user space

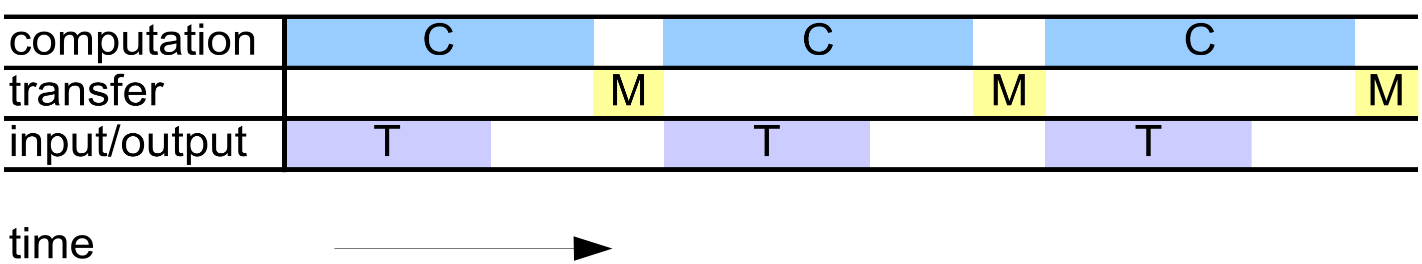
single buffer (output)

- we suppose the application does not need to wait until actual end of output operation
  - if T>C the process waits for buffer to be free
  - if T<C the OS waits for the buffer to be filled

- T>C



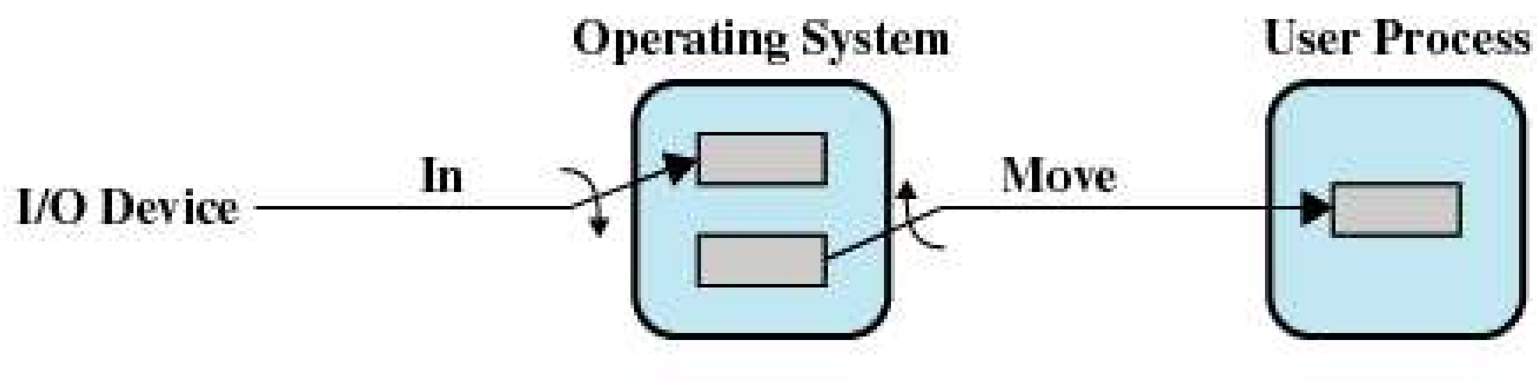
- T<C



- max[T,C]+M
  - M: time to move data from user space to system buffer

Double Buffer

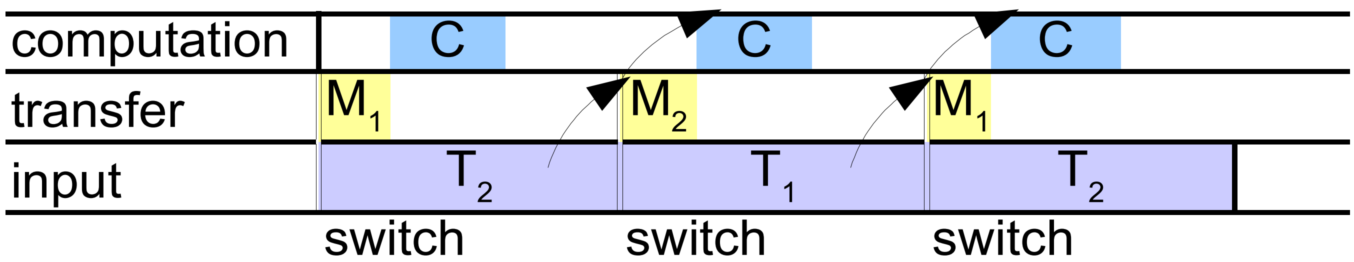
- Use two system buffers instead of one
- A process can transfer data to or from one buffer while the operating system empties or fills the other buffer



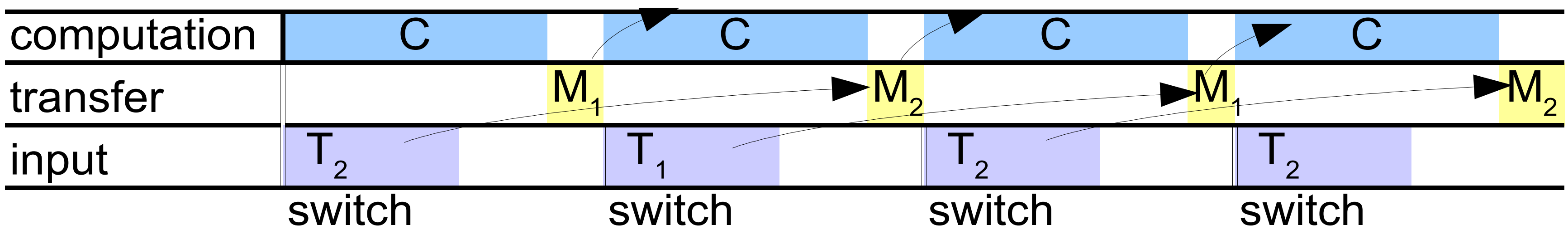
(c) Double buffering

double buffer (input)

- max[C+M, T]
- T>C, input

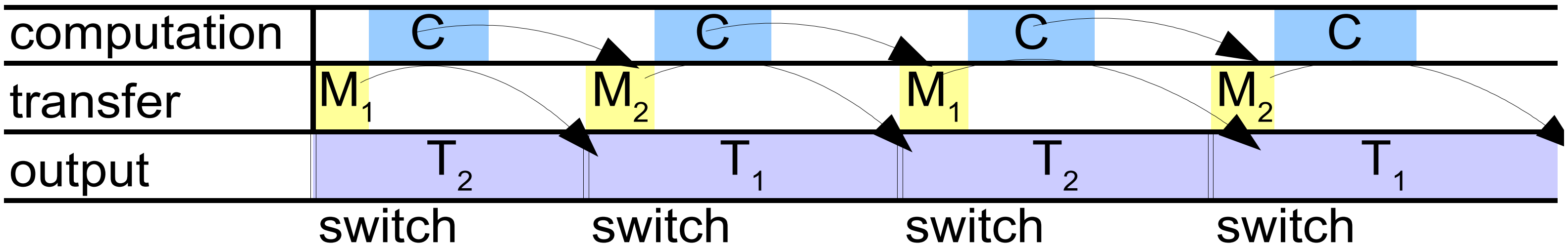


- T<C, input

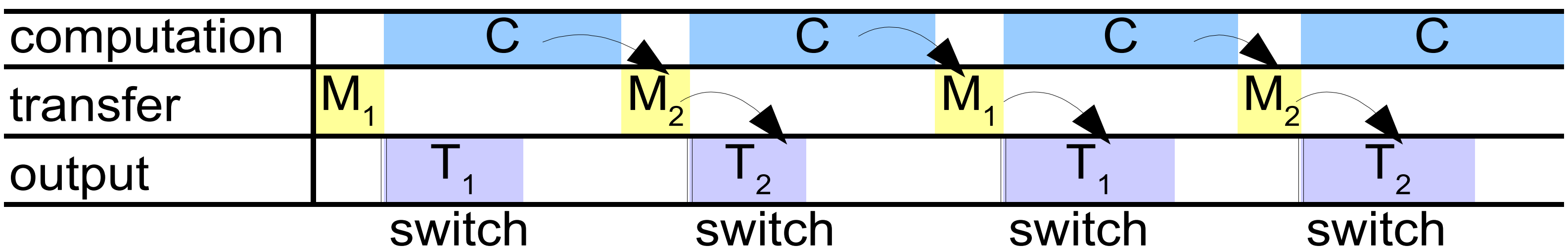


double buffer (output)

- max[C+M, T]
- T>C, output

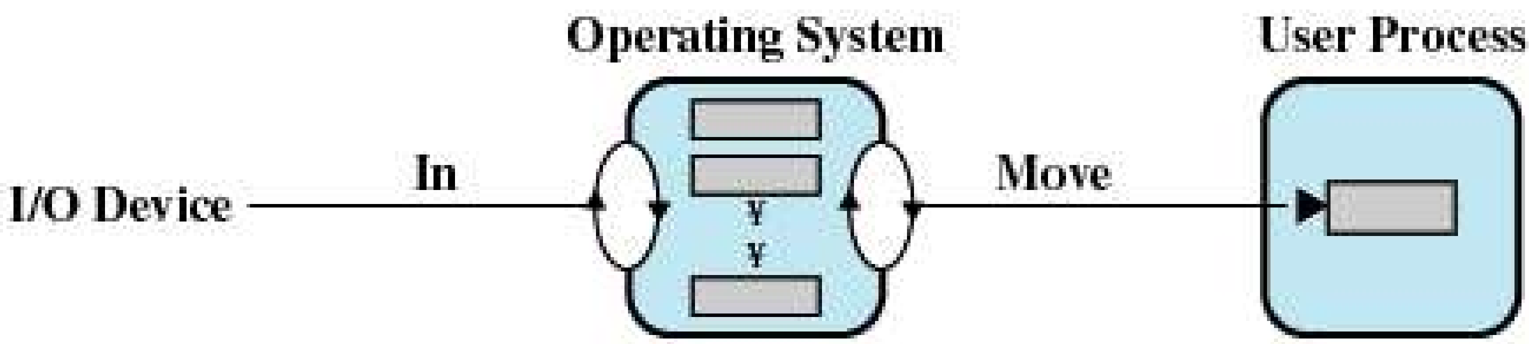


- T<C, output



Circular Buffer

- Buffers also smooth speed differences
- More than two buffers may be used



(d) Circular buffering

disk scheduling

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

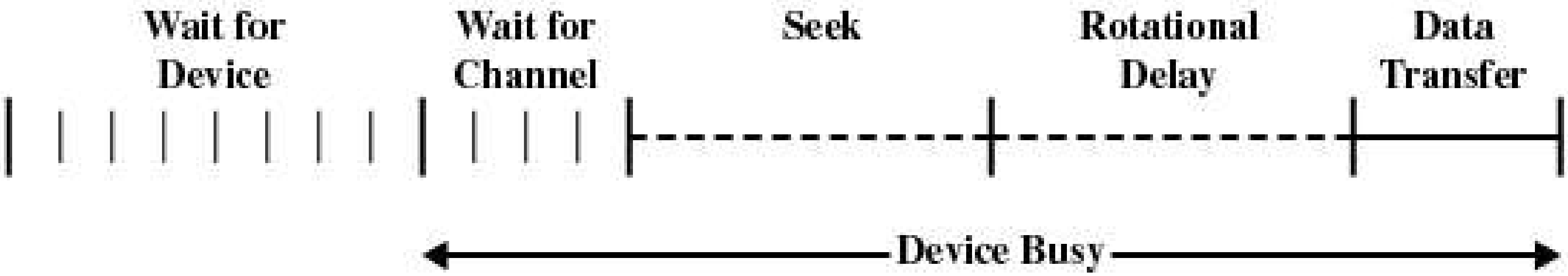


Figure 11.6 Timing of a Disk I/O Transfer

Disk Performance Parameters

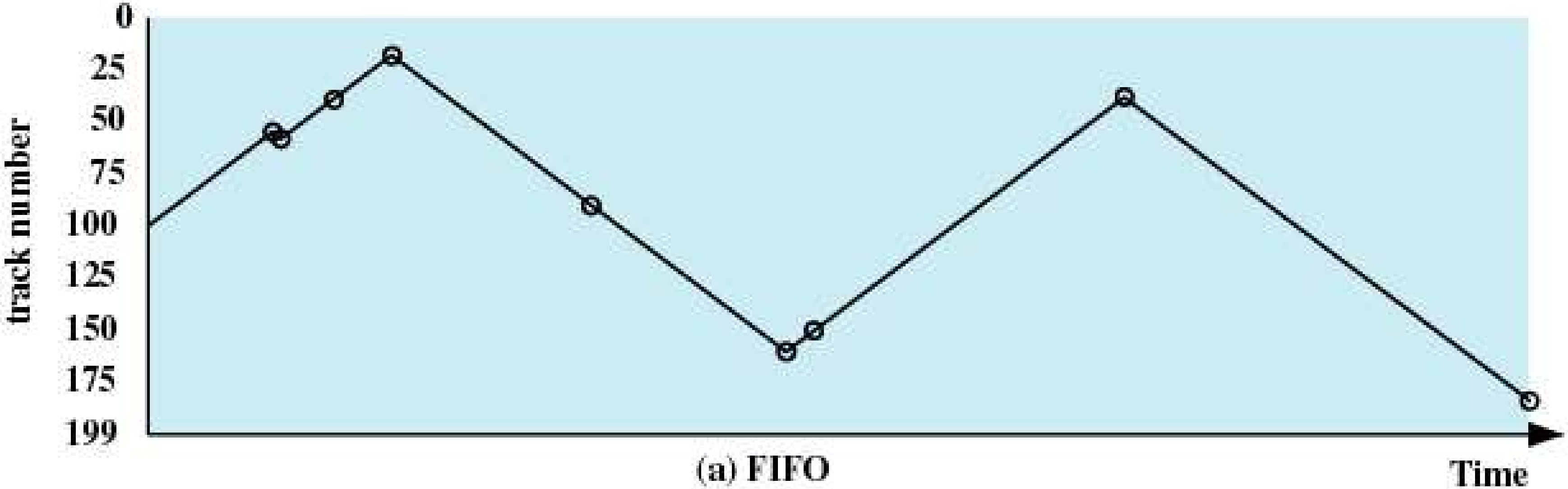
- Access time
  - Sum of seek time and rotational delay
  - The time it takes to get in position to read or write
- Data transfer occurs as the sector moves under the head

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 poor performance

Disk Scheduling Policies

- First-in, first-out (FIFO)
  - Process request sequentially
  - Fair to all processes
  - if there are many processes it performs like random scheduling



(a) FIFO

Disk Scheduling Policies

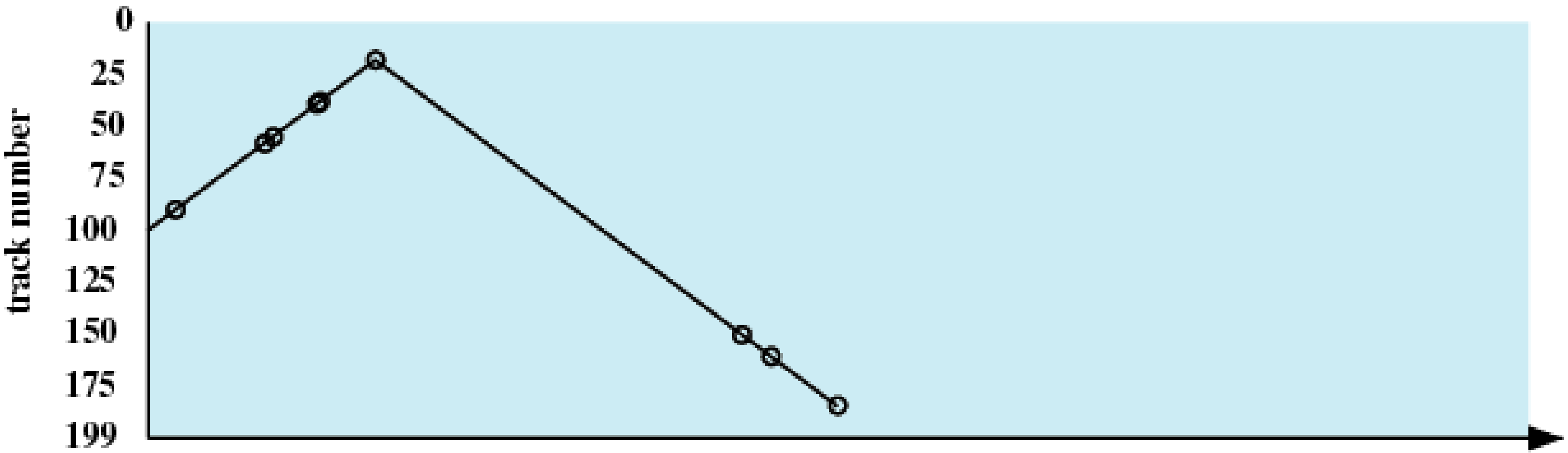
- Priority
  - Goal is not to optimize disk use but to meet other objectives
  - Short batch jobs may have higher priority
  - Provide good interactive response time

Disk Scheduling Policies

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

Disk Scheduling Policies

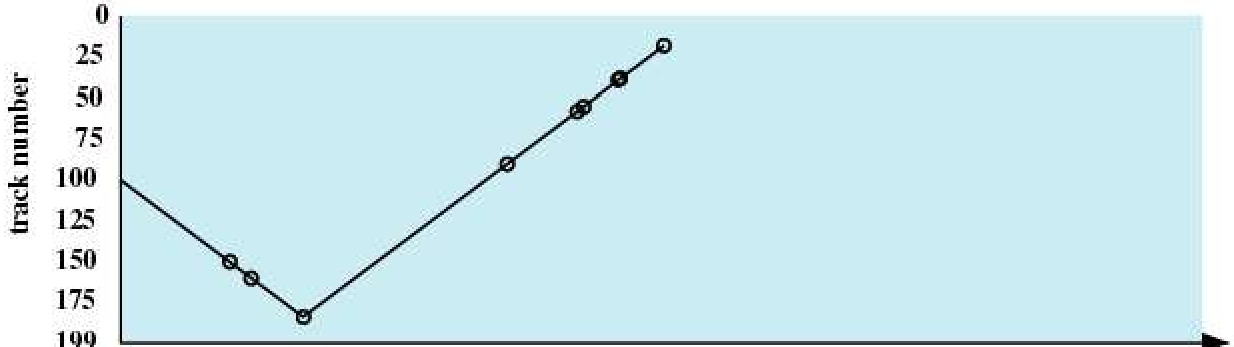
- Shortest Service Time First
  - Select the disk I/O request that requires the least movement of the disk arm from its current position
  - Always choose the minimum Seek time
  - Possibility of starvation



(b) SSTF

Disk Scheduling Policies

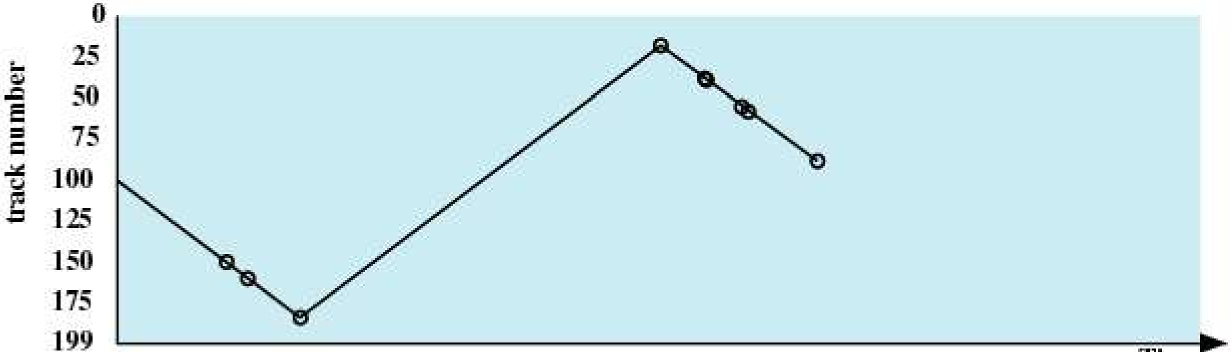
- SCAN (LOOK)
  - no starvation
  - Arm moves in one direction only, satisfying all outstanding requests until it reaches the last track in that direction
  - Direction is reversed



(c) SCAN

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



(d) C-SCAN

Disk Scheduling Policies

- N-step-SCAN
  - Segments the disk request queue into subqueues of length N
  - Subqueues are processed 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 requests

Disk Scheduling Algorithms

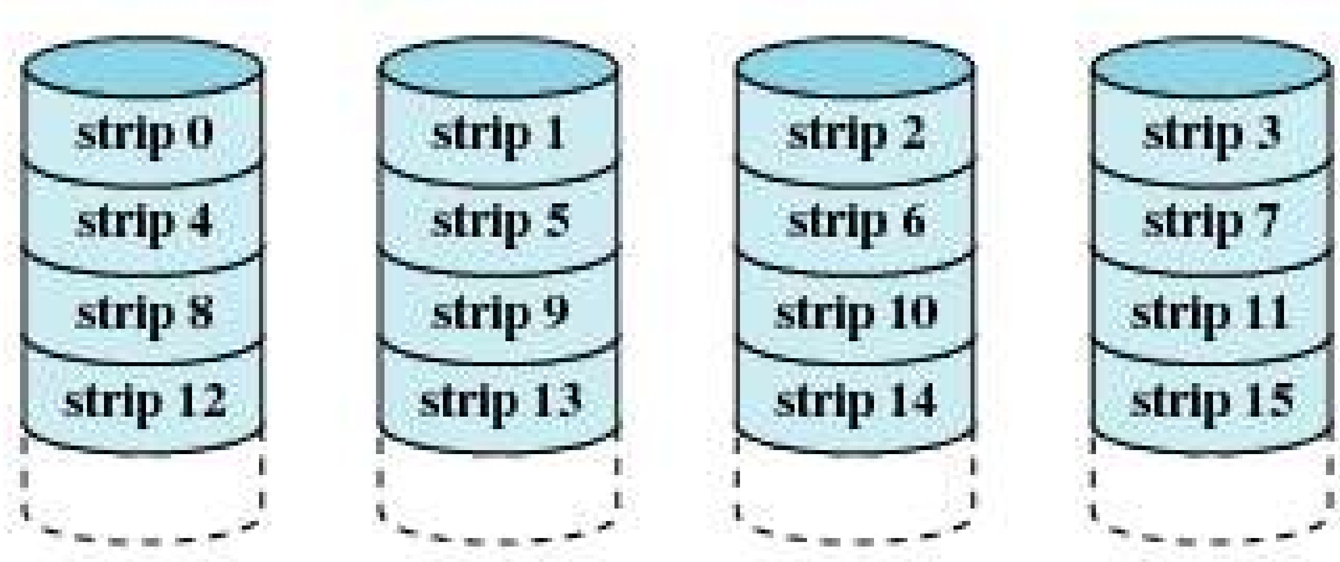
Table 11.2 Comparison of Disk Scheduling Algorithms

(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

RAID

- Redundant Array of Independent Disks
- Set of physical disk drives viewed by the operating system as a single logical drive
- Data are distributed across the physical drives of an array
- Redundant disk capacity is used to store parity information

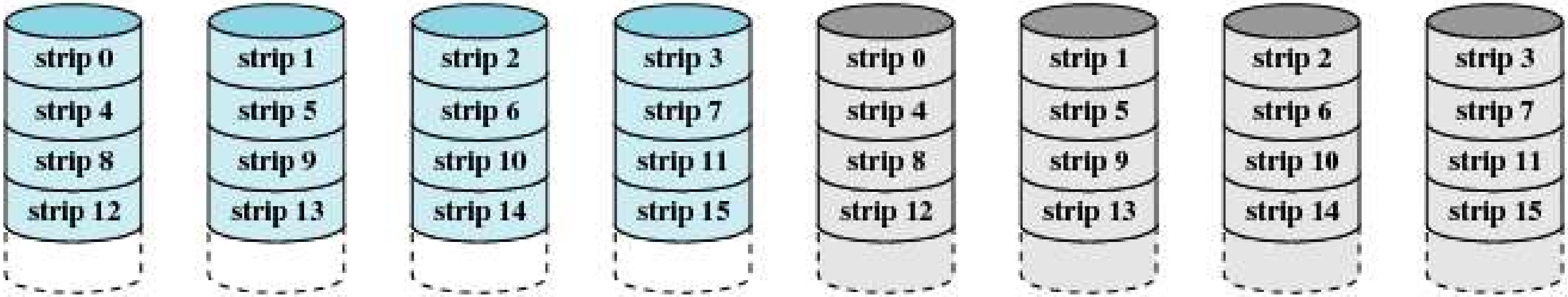
RAID 0 (non-redundant, striping)



(a) RAID 0 (non-redundant)

- availability: lower than single disk
- large I/O: very good
- high request rate: very good

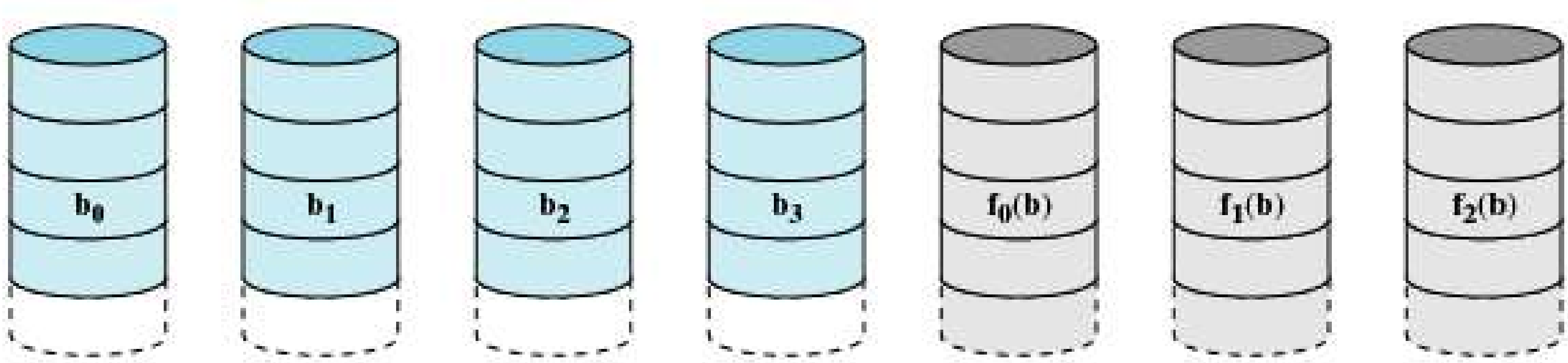
RAID 1 (mirrored)



(b) RAID 1 (mirrored)

- availability: high
- large I/O: very good for read
- high request rate: very good for read

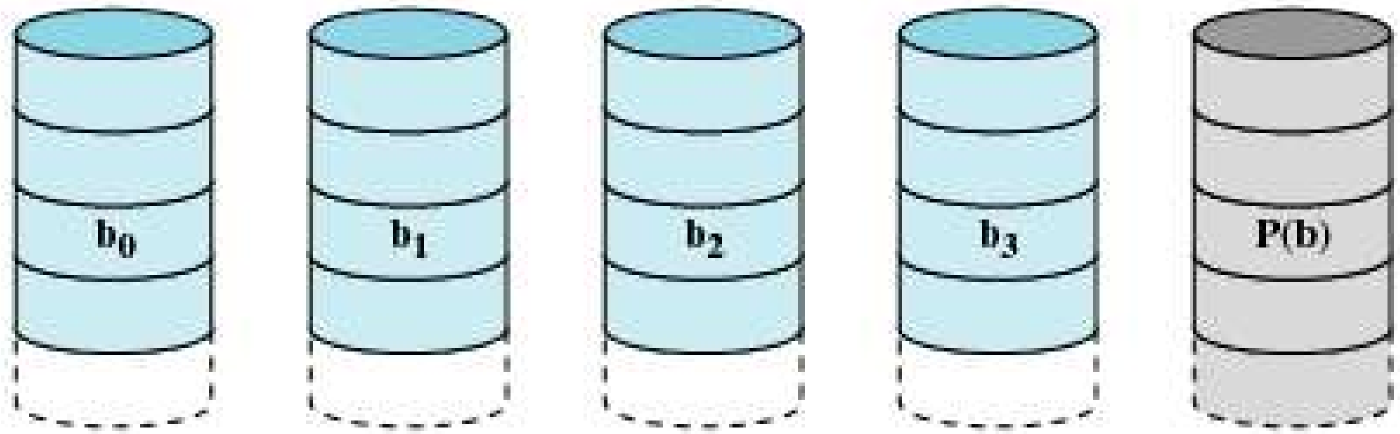
RAID 2 (redundancy through Hamming code)



(c) RAID 2 (redundancy through Hamming code)

- disks should be syncronized
- availability: high also for high bit error rate
- large I/O: best!!!
- high request rate: about twice single disk
- expensive!

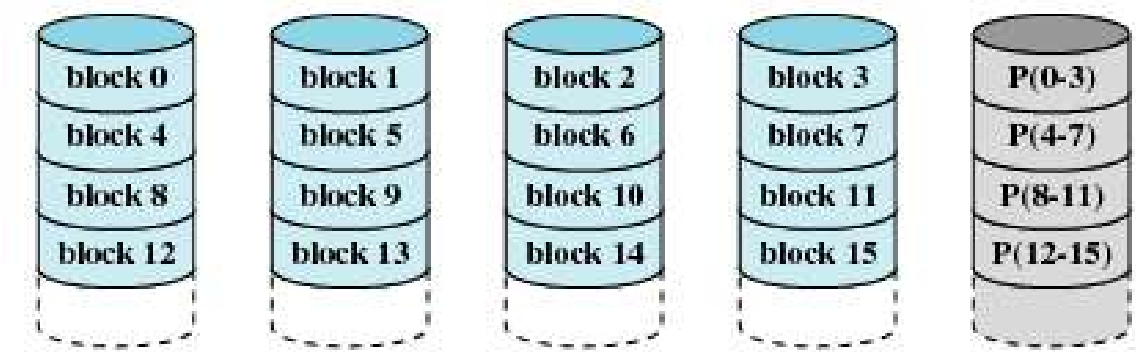
RAID 3 (bit-interleaved parity)



(d) RAID 3 (bit-interleaved parity)

- disks should be syncronized
- availability: high
- large I/O: best!!!
- high request rate: about twice single disk

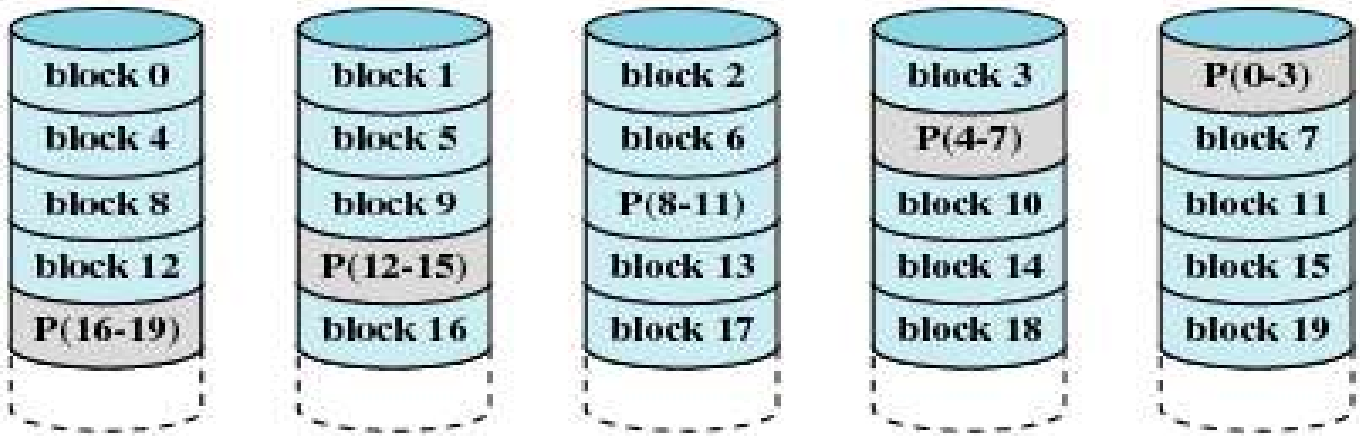
RAID 4 (block-level parity)



(e) RAID 4 (block-level parity)

- disks are independent
- availability: high
- **P is a bottlenek for write**
- large I/O: good, very bad for write
- high request rate: very good for read, very bad for write

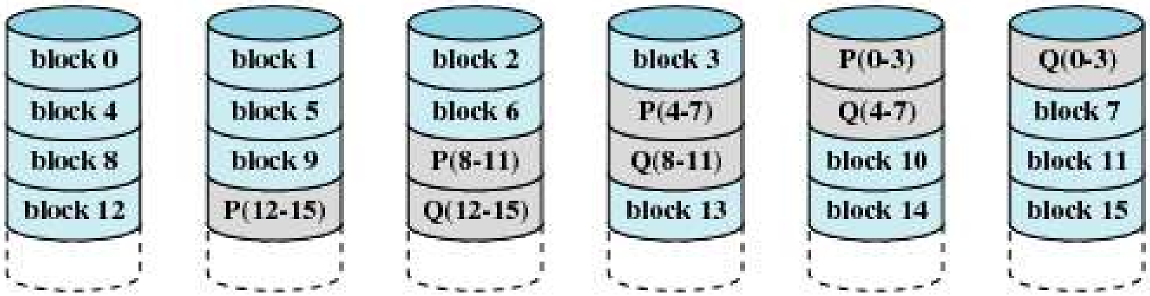
RAID 5 (block-level distributed parity)



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

- disks are independent
- availability: high
- large I/O: very good, bad for write (no bottlenek)
- high request rate: very good for read, bad for write

RAID 6 (dual redundancy)



(g) RAID 6 (dual redundancy)

- availability: highest
  - two disks may fail without data loss
- large I/O: very good for read, bad for write
- high request rate: very good for read, very bad for write