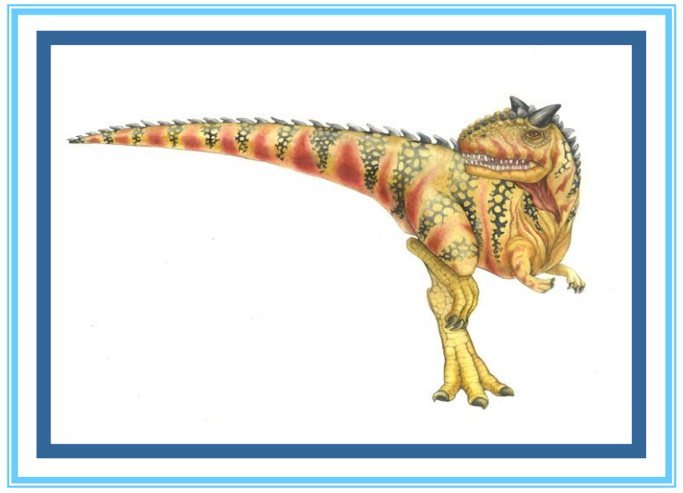
Chapter 12: Mass-Storage Systems

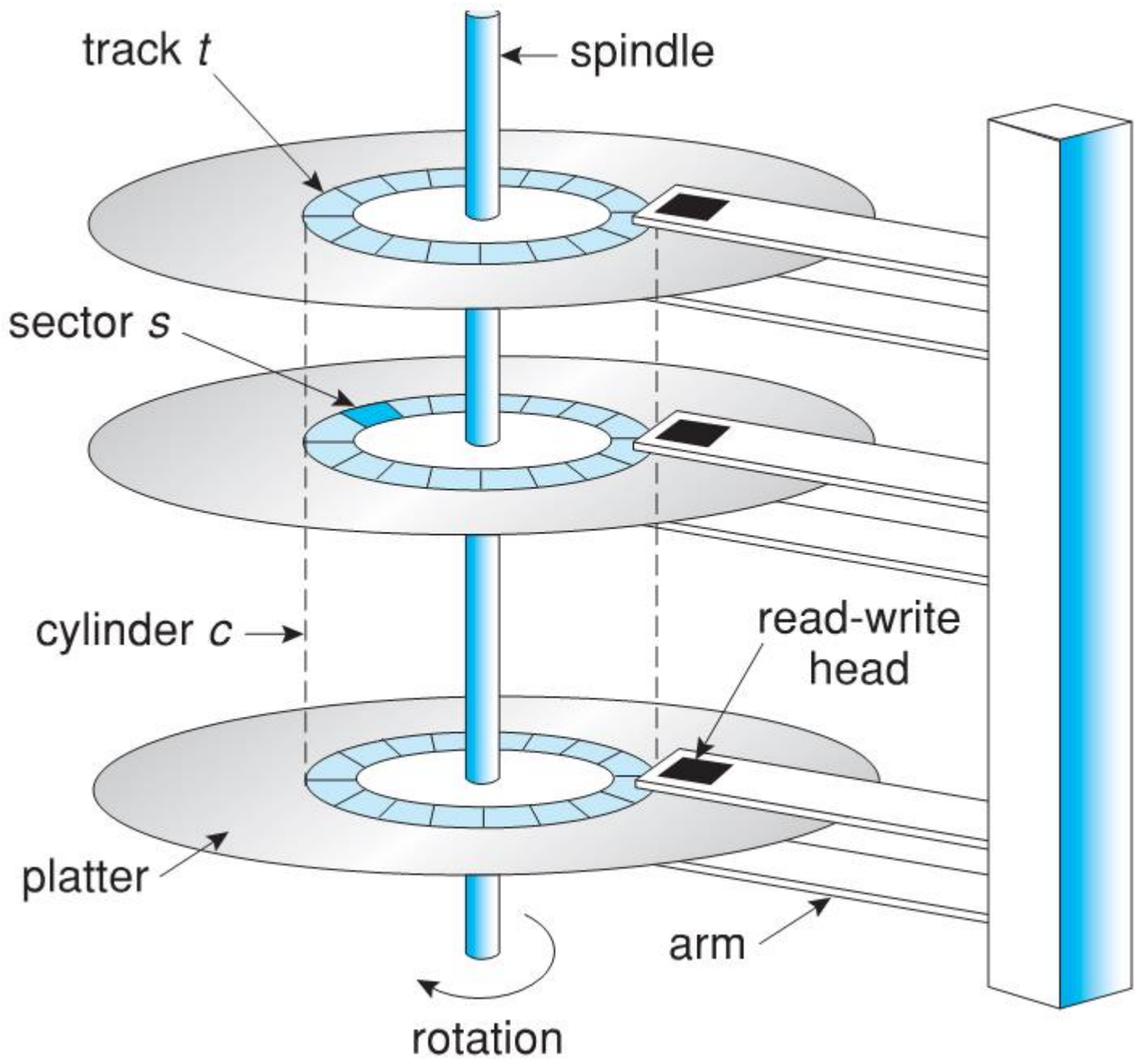


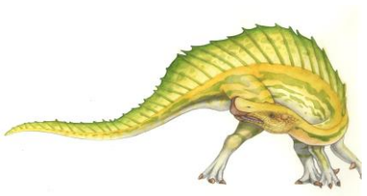
Chapter 12: Mass-Storage Systems

- Magnetic disk
- Disk Management
- Disk Scheduling
- Swap-Space Management
- RAID Structure

Magnetic Disks

- Moving-head Disk Mechanism





Disk Structure

- **Low-level formatting** — Dividing a disk into sectors that disk controller can r/w
  - A sector consists of: header, data (512bytes), error correction code (**ECC**)
  - Usually 512 bytes of data but can be selectable
  - Sector 0 is the first sector of the first track on the outermost cylinder. Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then the rest of the cylinders from outermost to innermost
- **Logical formatting** (making a file system): store initial file-system data structure on disk.
- Disks are addressed as large 1D arrays of **blocks** (the smallest unit of transfer).
- To increase efficiency most file systems group blocks into **clusters**
  - Disk I/O done in blocks
  - File I/O done in clusters (so more sequential-access)

Performance of Magnetic Disks

- **Magnetic disks** provide bulk of secondary storage of modern computers
  - **seek time:** time to move disk arm to desired cylinder
  - **rotational latency:** time for desired sector to rotate under the disk head
    - **Positioning time**: seek time + rotational latency
  - **Transfer rate** is rate at which data flow between drive and computer
- Disks can be removable
- Drive attached to computer via **I/O bus**
  - Busses vary, including
    - **EIDE**, **ATA**, **SATA**, **USB**, **Fiber Channel**, **SCSI, SAS, Firewire**
  - **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array

Performance of Magnetic Disks

- Performance
  - Seek time: from 3ms to 12ms
    - average seek time measured
  - Latency based on spindle speed
    - 1/(RPM / 60)
    - average latency = ½ latency

4200rpm: 1/(4200/60)= 0.01428s
➔ Avg latency: 7.14ms

  - Transfer Rate
    - theoretical – 6 Gb/sec, real – 1Gb/sec

- **Average I/O time**

  = controller overhead

  + average seek time

  + average latency

  + (amount of data / transfer rate)

- Example: to transfer a 4KB block on a 7200 RPM disk with a 5ms average seek time, 1Gb/sec transfer rate with a 0.1ms controller overhead
  - 0.1ms + 5ms +1/2×1/ (7200/60) sec + 4KB / 1Gb/sec

  = 0.1ms + 5ms + 0.00417sec + 0.000124 sec = 9.39ms

Chapter 12: Mass-Storage Systems

- Magnetic disk
- Disk Management
- Disk Scheduling
- RAID Structure

Disk Scheduling

- OS is responsible for using hardware efficiently
  - Minimize seek time (≈ seek distance)
  - Maximized disk **bandwidth:** total number of bytes transferred, divided by the time between the first request and the completion of the last transfer
- To improve the seek time and disk bandwidth ➔ by managing the order where disk I/O requests are serviced.
- I/O request includes: input or output mode, disk address, memory address, number of sectors to transfer
- Idle disk can immediately work on I/O request, busy disk means work must queue
  - Optimization algorithms only make sense when a queue exists
  - OS maintains queue of requests
- Several algorithms exist to schedule the servicing of disk I/O requests
  - We illustrate scheduling algorithms with a queue below, head pointer 53

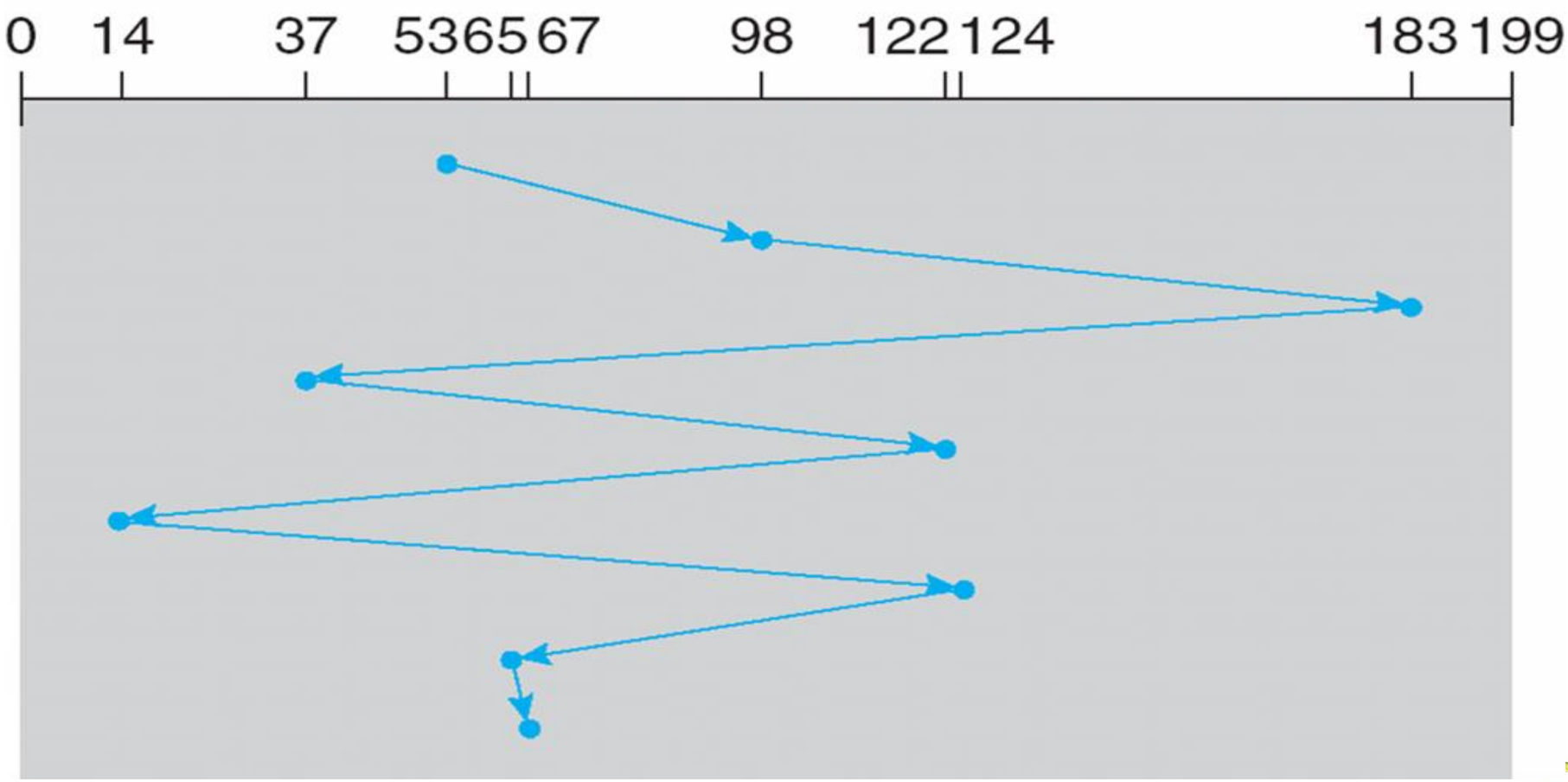
  98, 183, 37, 122, 14, 124, 65, 67

FCFS

Illustration shows total head movement of 640 cylinders:

53→98→183→37→122→14→124→65

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

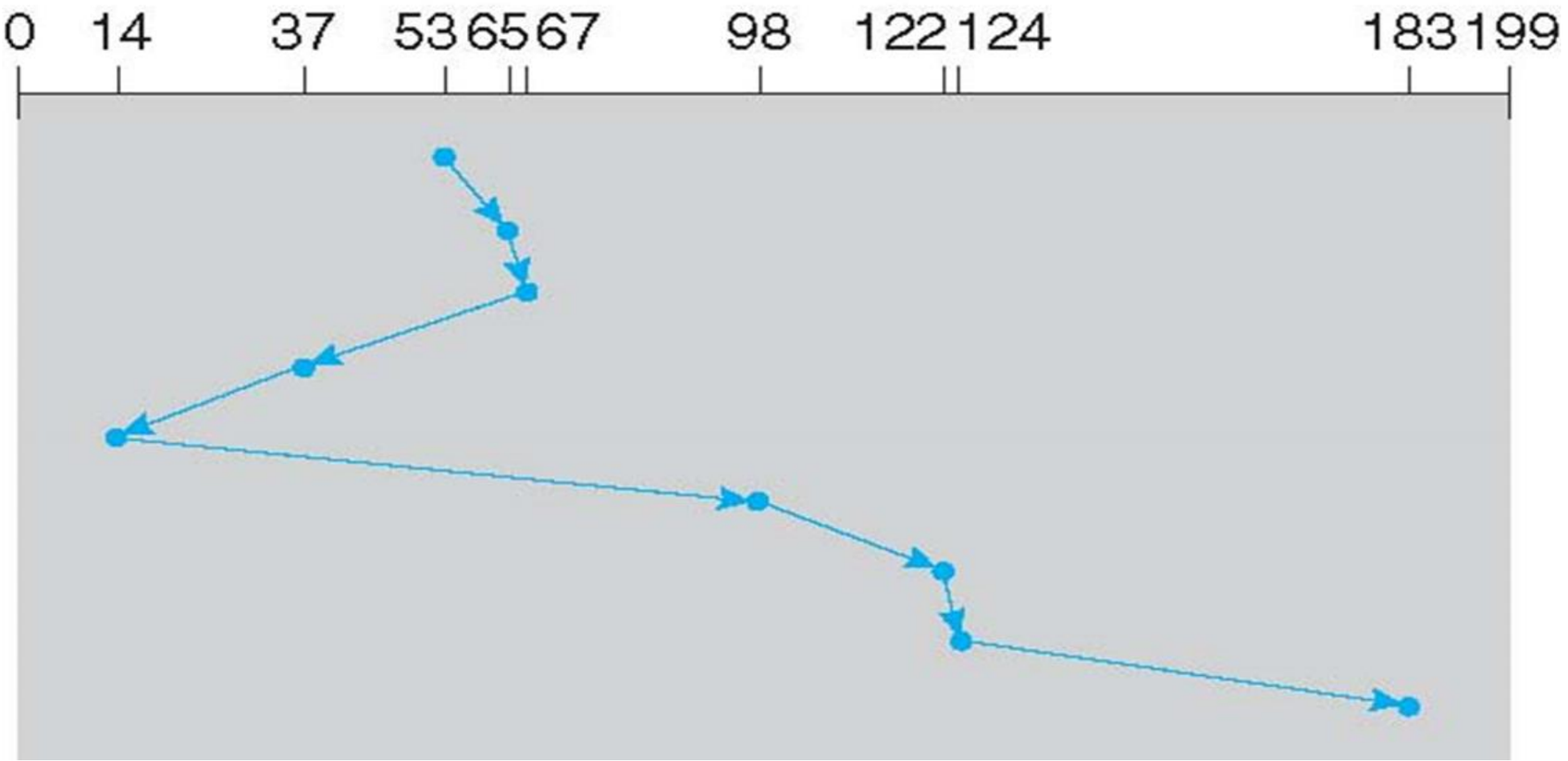


SSTF (Shortest Seek Time First)

- SSTF selects the request with the minimum seek time from the current head position
- SSTF is a form of SJF scheduling; may cause starvation of some requests
- Illustration shows total head movement of 236 cylinders:

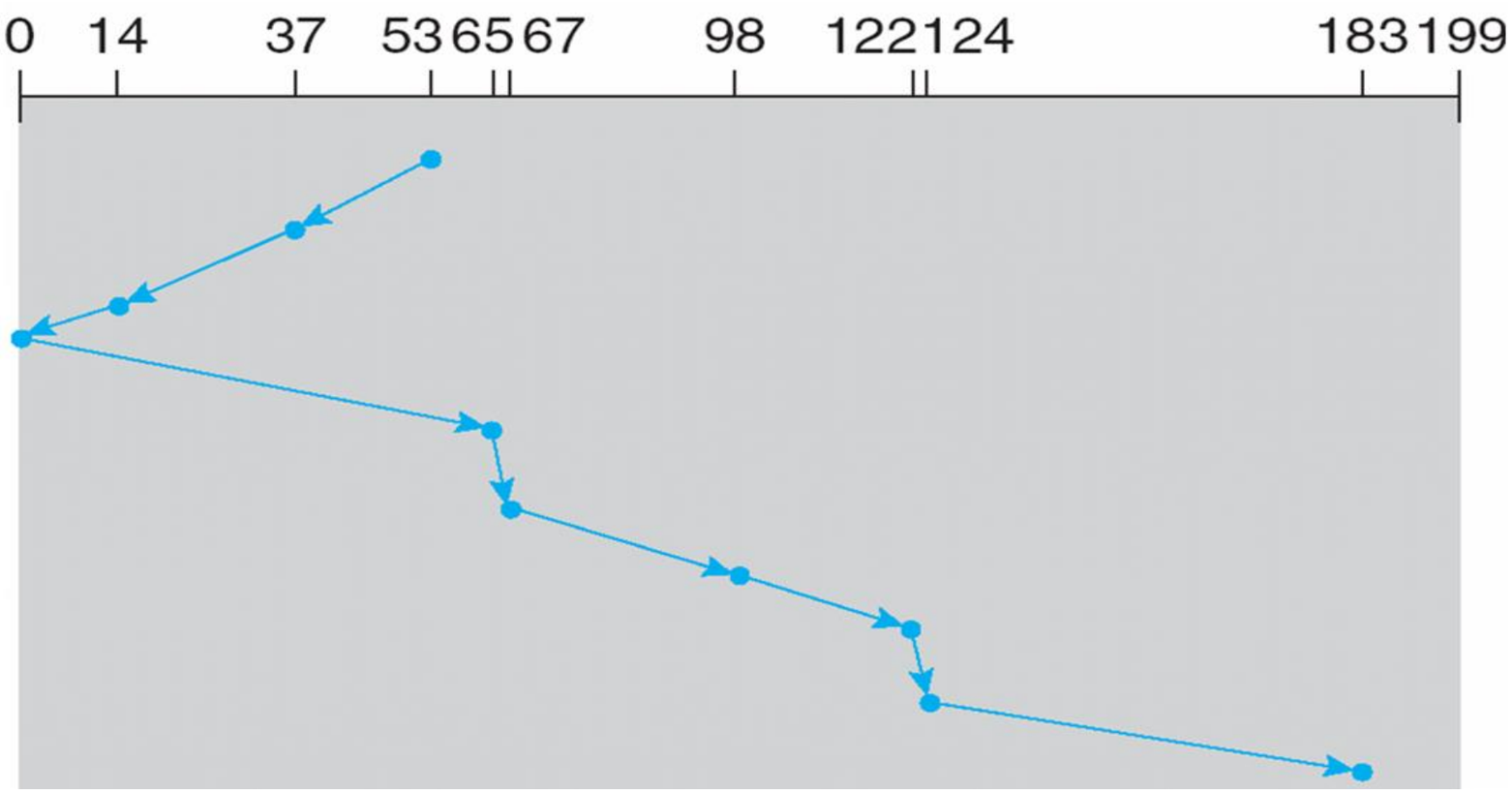
  53→65→67→37→14→98→122→124→183

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



SCAN

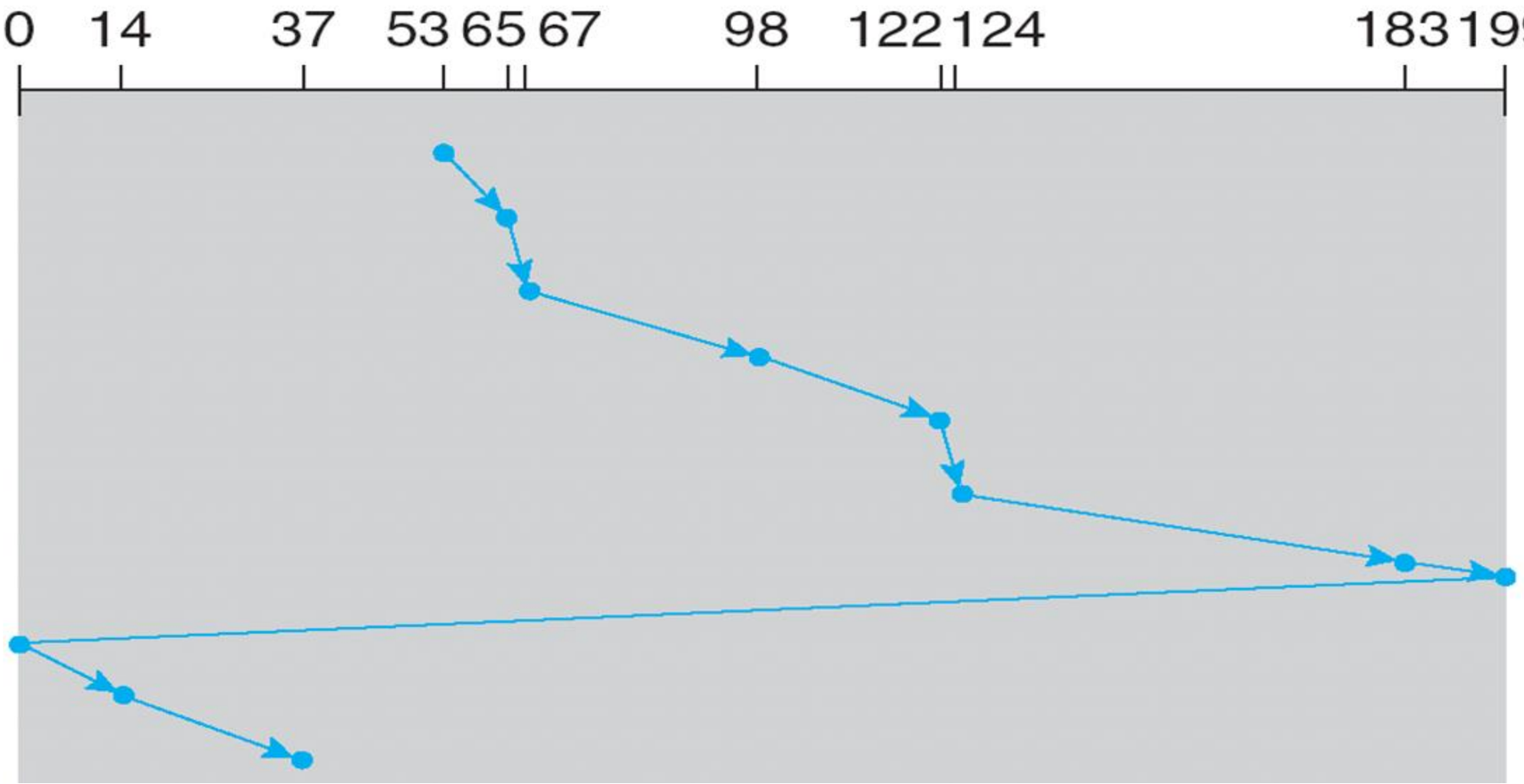
- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- **SCAN algorithm:** sometimes called the **elevator algorithm**
- total head movement of 236 cylinders (assume disk arm is moving toward 0)
  - 53→37→14→ **0** →65→67→98→122→124→183



SCAN favors the requests near to both the innermost and the outermost tracks

C-SCAN

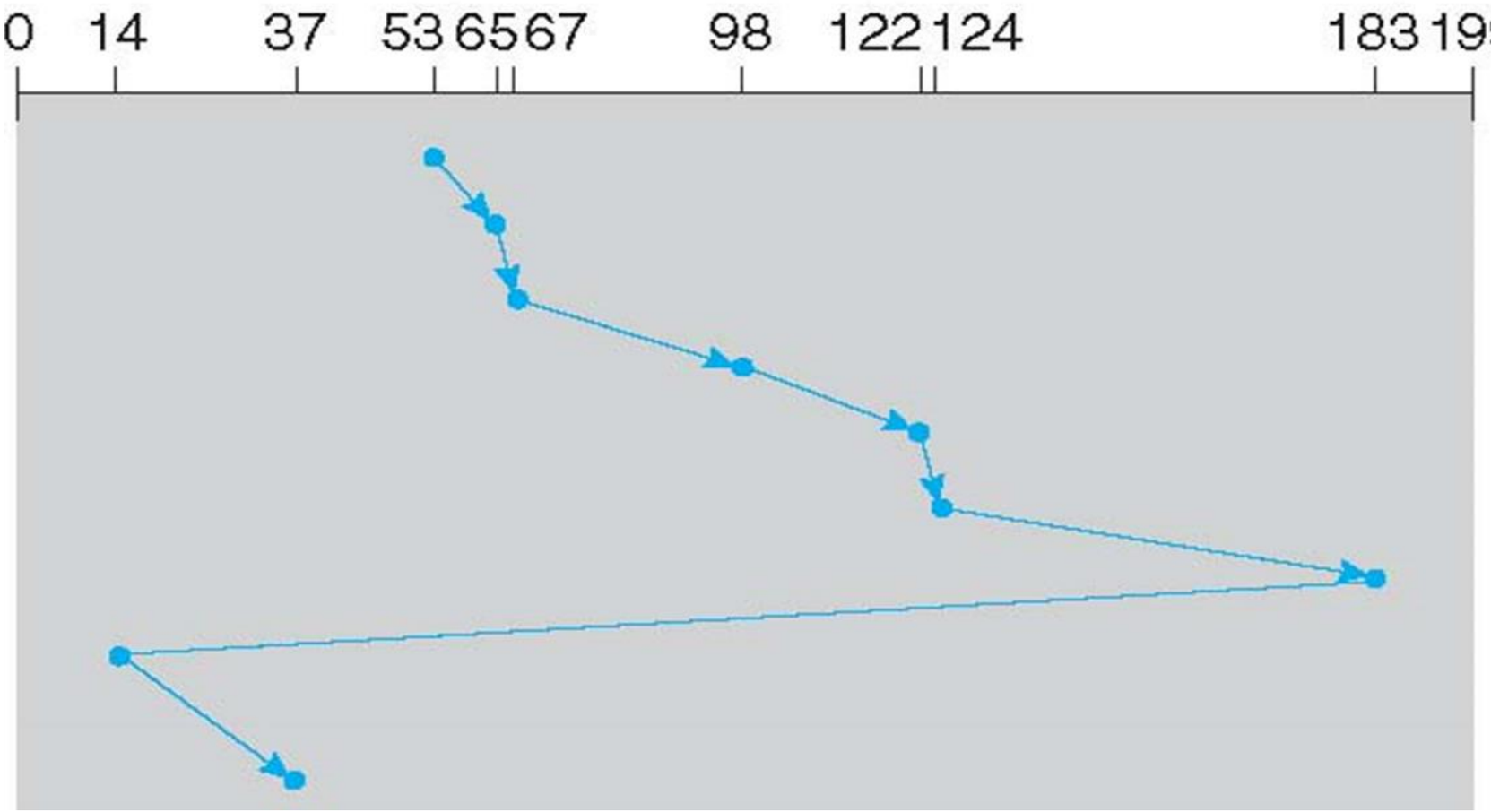
- Provides a more uniform wait time than SCAN
- The head moves from one end of the disk to the other, servicing requests as it goes
  - When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip
- Total number of cylinders?



LOOK & C-LOOK

- Arm only goes as far as the last request in each direction, then reverses direction immediately, without going all the way to the end of the disk
  - SCAN that follows this pattern is called LOOK:
  - C-SCAN that follows this pattern is called C-LOOK
- Total number of cylinders? LOOK: 208

C-LOOK



Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on disk
  - Less starvation
- Performance depends on the number and types of requests
- Requests for disk service can be influenced by the file-allocation method and metadata layout
  - Continuously allocated file will generate requests that are close together on disk
  - Linked or indexed file may include blocks that are widely scattered on the disk
  - Put the directory entry on middle cylinder or caching the directory in memory can improve the performance
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary
- What about rotational latency? Difficult for OS to calculate
- I/O performance may not be the only consideration. OS may have other constraints on the order of request servicing. (e.g., flush file metadata block to disk first)

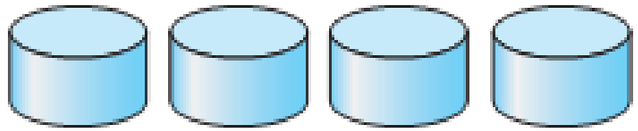
Chapter 12: Mass-Storage Systems

- Magnetic disk
- Disk Management
- Disk Scheduling
- RAID Structure

RAID structure

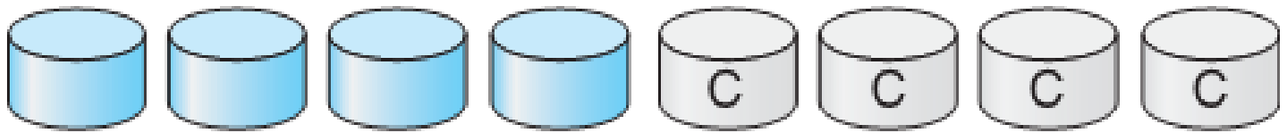
- RAID – redundant array of inexpensive disks
  - multiple disk drives provides reliability via **redundancy**
- Disk **striping** uses a group of disks as one storage unit to improve transfer rate
  - bit-level striping: reduce response time of both large and small accesses
  - block-level striping (**RAID0**): reduce response time of large accesses, increase the throughput of multiple small access by load balancing
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data
  - **Mirroring** or **shadowing** (**RAID 1**) keeps duplicate of each disk
  - **Block interleaved parity** (**RAID 4, 5, 6**) uses much less redundancy
- RAID within a storage array can still fail if the array fails, so automatic **replication** of the data between arrays is common
- Frequently, a small number of **hot-spare** disks are left unallocated, automatically replacing a failed disk and having data rebuilt onto them

RAID Levels

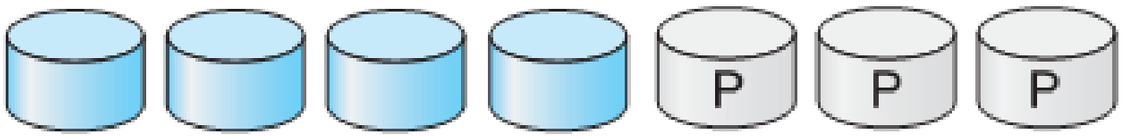


(a) RAID 0: non-redundant striping.

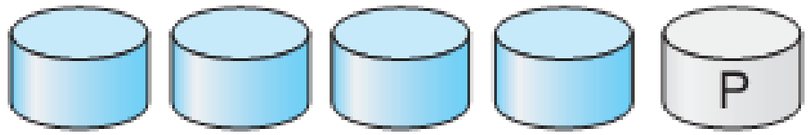
(block-interleaved)



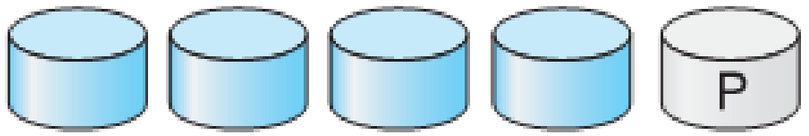
(b) RAID 1: mirrored disks.



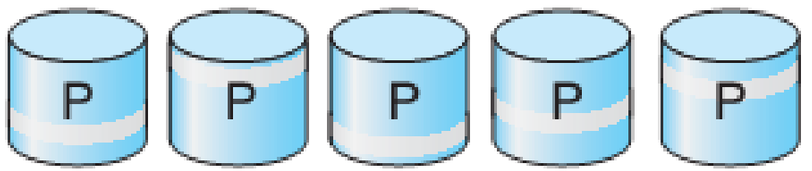
(c) RAID 2: memory-style error-correcting codes.



(d) RAID 3: bit-interleaved parity.



(e) RAID 4: block-interleaved parity.



(f) RAID 5: block-interleaved distributed parity.

C: a second copy of the data
P: parity (error-correcting data)

RAID 0, 1, 2

- **RAID 0: Non-redundancy striping**
  - Striping at block level
  - Without any redundancy
- **RAID 1: Mirroring**
  - N + N disks, replicate data
  - Write data to both data disk and mirror disk
  - On disk failure, read from mirror
  - Simple, widely used.
- **RAID 2: Bit-interleaved Hamming-code parity**
  - N + E disks (e.g., 4 + 3)
  - Split data at bit level across N disks
  - Generate E-bit ECC
  - Too complex, not used in practice

RAID 3, 4

- **RAID 3: bit-Interleaved Parity (N + 1 disks)**
  - Data striped across N disks at bit level
  - Parity disk: Redundant disk stores parity
  - Read access: Read all disks (high transfer rates, but no multiple I/Os in parallel)
  - Write access: Generate new parity and update all disks
  - On failure: Use parity to reconstruct missing data
  - Not widely used
- **RAID 4: block-Interleaved Parity (N + 1 disks)**
  - Data striped across N disks at block level
  - Parity disk: Redundant disk stores parity for a group of blocks
  - Read access: Read only the required blocks (allow multiple small reads in parallel)
  - Write access: read-modify-write (small independent writes cannot be done in parallel)
  - On failure: Use parity to reconstruct missing data
  - Not widely used

RAID 5

- **RAID 5: Distributed Parity (N + 1 disks)**
  - Distributed Parity: Like RAID 4, but parity blocks distributed across disks
    - Avoids parity disk being a bottleneck
  - Read access:
    - Read only the required blocks
    - allow multiple small reads in parallel
  - Write access:
    - read-modify-write
    - some small independent writes can be done in parallel
  - Widely used

RAID1+0 vs RAID0+1

- RAID 1+0
  - Every two disks form a RAID1 (mirroring) to increase reliability, and then use striping on these RAID1s to form a RAID0 to increase the throughput.
- RAID 0+1
  - Divide disks into two sets (each set consists of equal numbers of disks). In each set, striping is applied to increase the throughput, and then mirroring is applied across these two sets for reliability.
- Given four disks a, b, c, d