Operating Systems 2017/2018 | Part VII – Storage Devices

Ricardo Rocha

Department of Computer Science

Faculty of Sciences

University of Porto

Slides based on the book

'Operating System Concepts, 9th Edition,

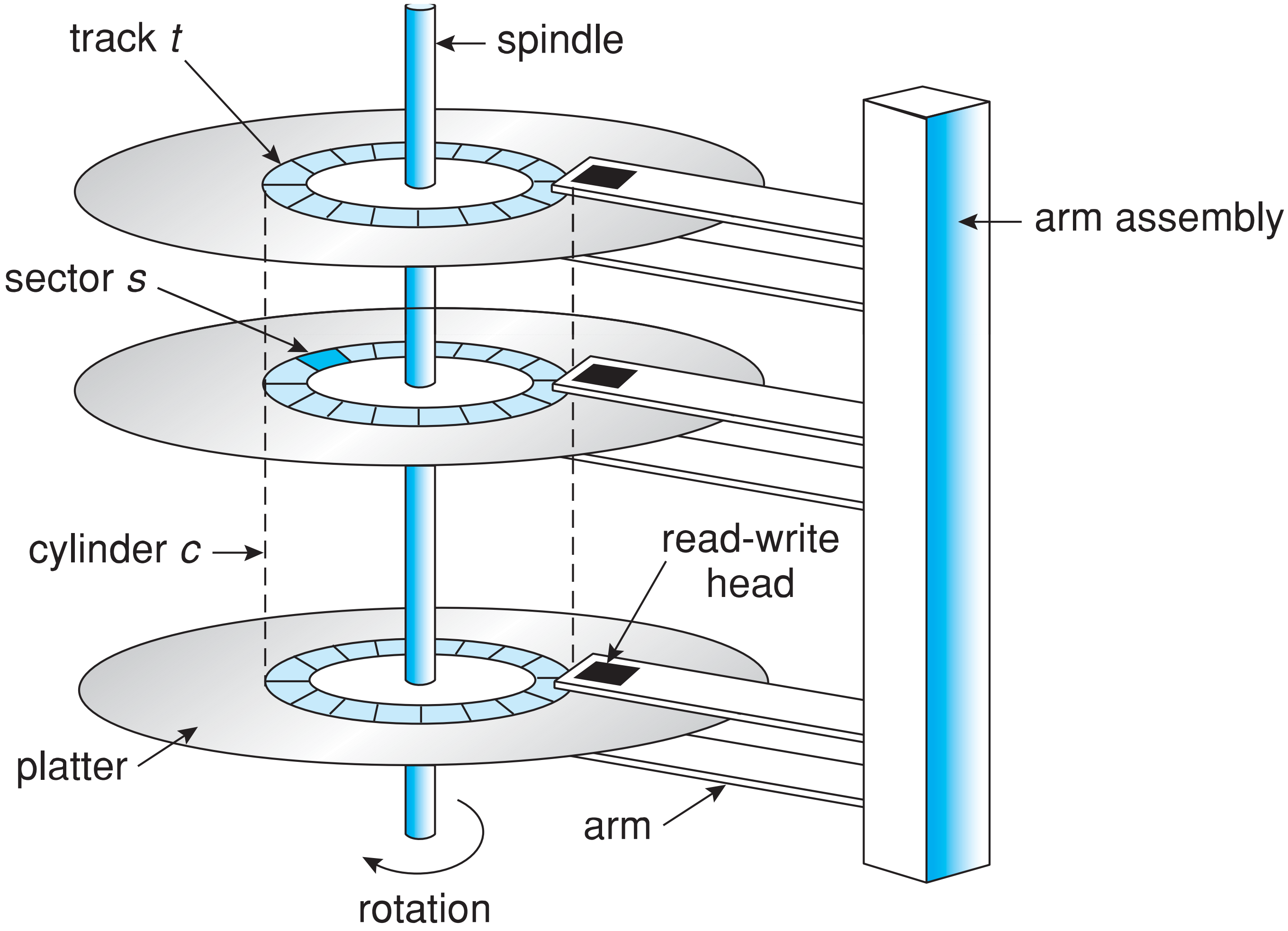
Abraham Silberschatz, Peter B. Galvin and Greg Gagne, Wiley'

Chapter 12

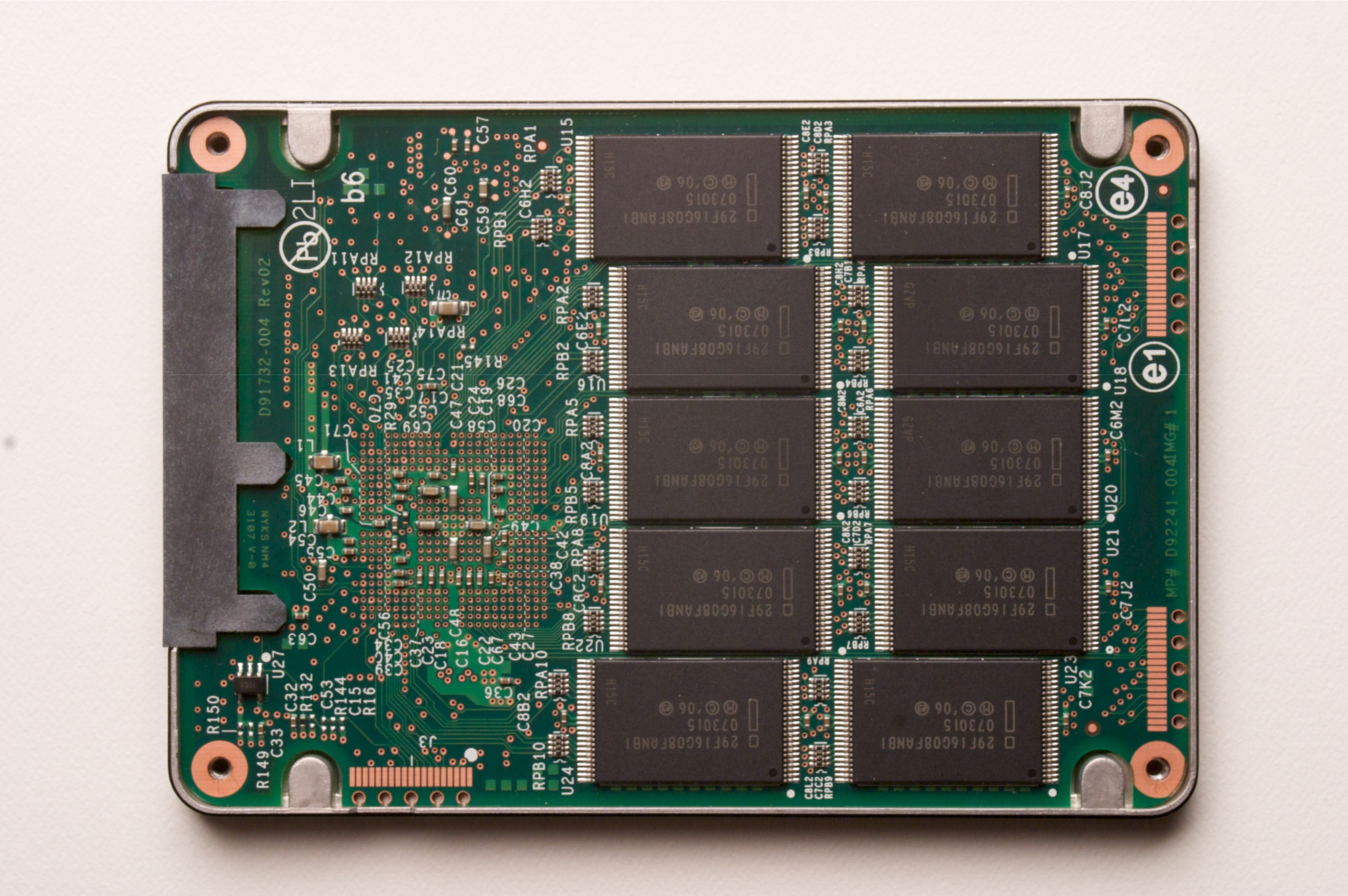
Hard Disk Drive (HDD)

HDD Overview

- Data is stored on HDDs by recording it magnetically on a set of **platters**
  - Common platter diameters range from 1.8 to 3.5 inches
  - A platter is logically divided into (thousands of) circular **tracks** consisting of (hundreds of) **sectors**
  - A set of tracks that are at one arm position makes up a **cylinder**



Solid State Disk (SSD)



SSD Overview

- An SSD is nonvolatile memory that is used like a hard drive
  - Many technology variations
- SSDs have the same characteristics as traditional HDDs but:
  - Can be more reliable because they have no moving parts
  - Can be faster because they have no seek or latency time
  - Can consume less power
- On the other hand, SSDs:
  - Have less capacity and are more expensive per megabyte
  - May have shorter life spans, so their use is somewhat limited (some systems, e.g. laptops, use them as a direct replacement for disk drives, while others use them as a new cache tier, moving data between HDDs, SSDs and memory to optimize performance)

Data Transfers

- A disk drive is attached to a computer via a set of wires called an **I/O bus**
  - Busses vary, including EIDE, ATA, SATA, USB, SCSI, Firewire, …
- Data transfers are carried out by special hardware called **controllers**
  - The **host controller** is the controller at the computer end of the bus
  - The **disk controller** is the controller built into each disk drive
- To perform a disk I/O operation:
  - The operating system starts by placing a command into the host controller
  - The host controller then sends the command to the disk controller
  - The disk controller operates the HDD hardware to carry out the command
  - Data transfer at the HDD happens between the disk surface and a built-in cache in the disk controller
  - Data transfer to the host occurs between the cache and the host controller

Latency Times

- Any I/O operation is a five-stage process:
  - **Queuing time** is the time for the device driver to process the desired request
  - **Controller time** is the time for the disk controller to carry out the desired operation
  - **Seek time** is the time to move the disk arm to the desired cylinder
  - **Rotational latency** is the time for the desired sector to rotate under the disk head
  - **Transfer time** is the time to transfer the desired data between the drive and the computer
- **Disk latency** is the sum of all times above and corresponds to the total amount of time between a request and the result be available

Typical Numbers

- Queueing time depends on device driver
- Controller time depends on controller hardware
- Average seek time typically 3–12 milliseconds
- Average rotational latency depends on RPM (rotations per minute) speed
  - 15,000 RPM = 250 RPS = 4ms per rotation = 2ms average rotational latency
  - 10,000 RPM = 167 RPS = 6ms per rotation = 3ms average rotational latency
- Transfer time typically 50-100 MB per second (depends on transfer size, RPM speed and bits density on track)

Performance Example

- Assumptions:
  - Ignoring queuing and controller times for now
  - Transfer rate of 100 MB/s, sector size of 1 KB
  - 15,000 RPM disk ⇒ 2ms average rotational latency, 3ms average seek time
- Read sector from random place on disk:
  - 3ms (seek) + 2ms (rotational latency) + 0.01ms (transfer) = 5.01ms
  - Effective transfer rate: approximately **200 KB per second**
- Read sector from random place in same cylinder:
  - 2ms (rotational latency) + 0.01ms (transfer) = 2.01ms
  - Effective transfer rate: approximately **500 KB per second (2.5 times better!)**

Performance Example

- Assumptions:
  - Ignoring queuing and controller times for now
  - Transfer rate of 100 MB/s, sector size of 1 KB
  - 15,000 RPM disk ⇒ 2ms average rotational latency, 3ms average seek time
- Read next sector in same track:
  - 0.01ms (transfer)
  - Effective transfer rate: **100 MB per second (200 times better !!!)**
- Key to use disks effectively is to **minimize seek and rotational delays**

Disk Scheduling

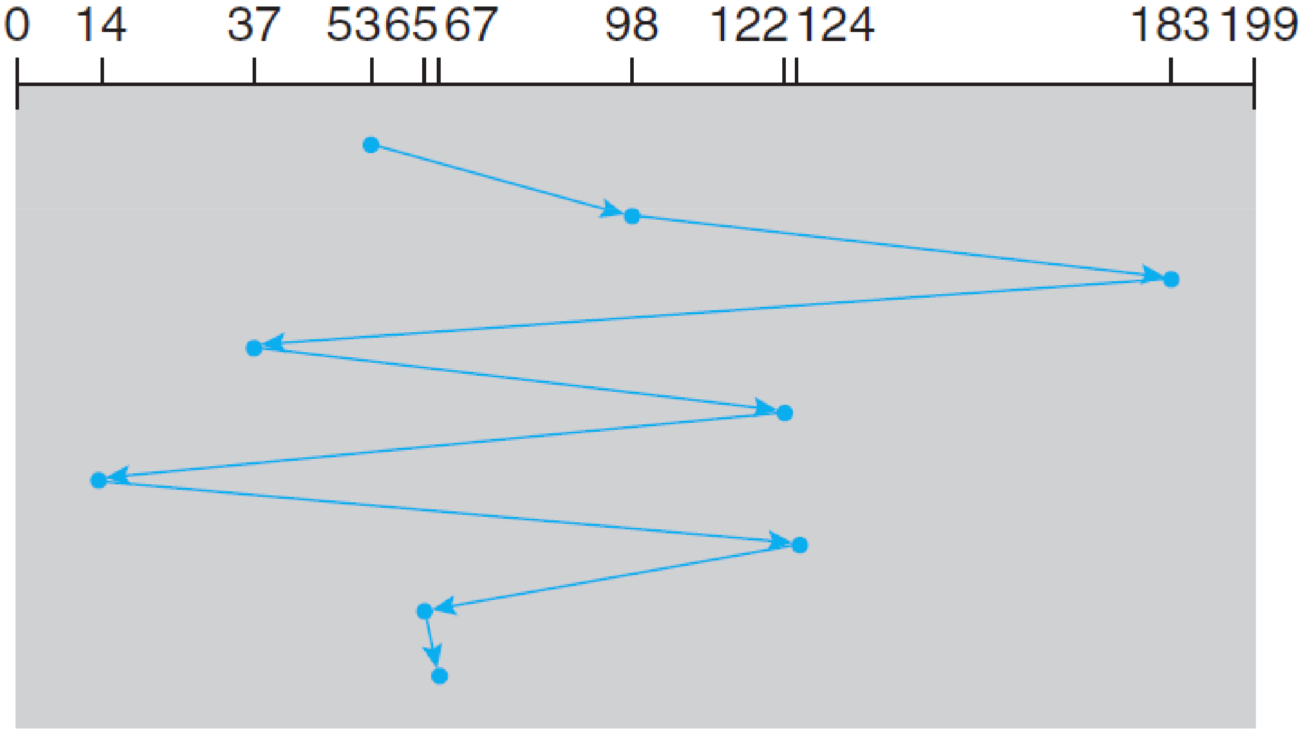
- An I/O request specifies several pieces of information:
  - The type of operation (input or output)
  - The disk and memory addresses for the transfer
  - The number of sectors to be transferred
- Operating system maintains a **queue of pending requests** per disk
  - When one request completes, OS chooses which request to service next
- How does the operating system schedules the servicing of disk requests?
  - Several optimization algorithms exist (most of them only consider the tracks being requested, ignoring the sectors)
  - Optimization algorithms only make sense when a queue exists, otherwise a pending request can be serviced immediately

First-Come First-Served (FCFS)

- FCFS services requests in the order they arrive

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

head starts at 53



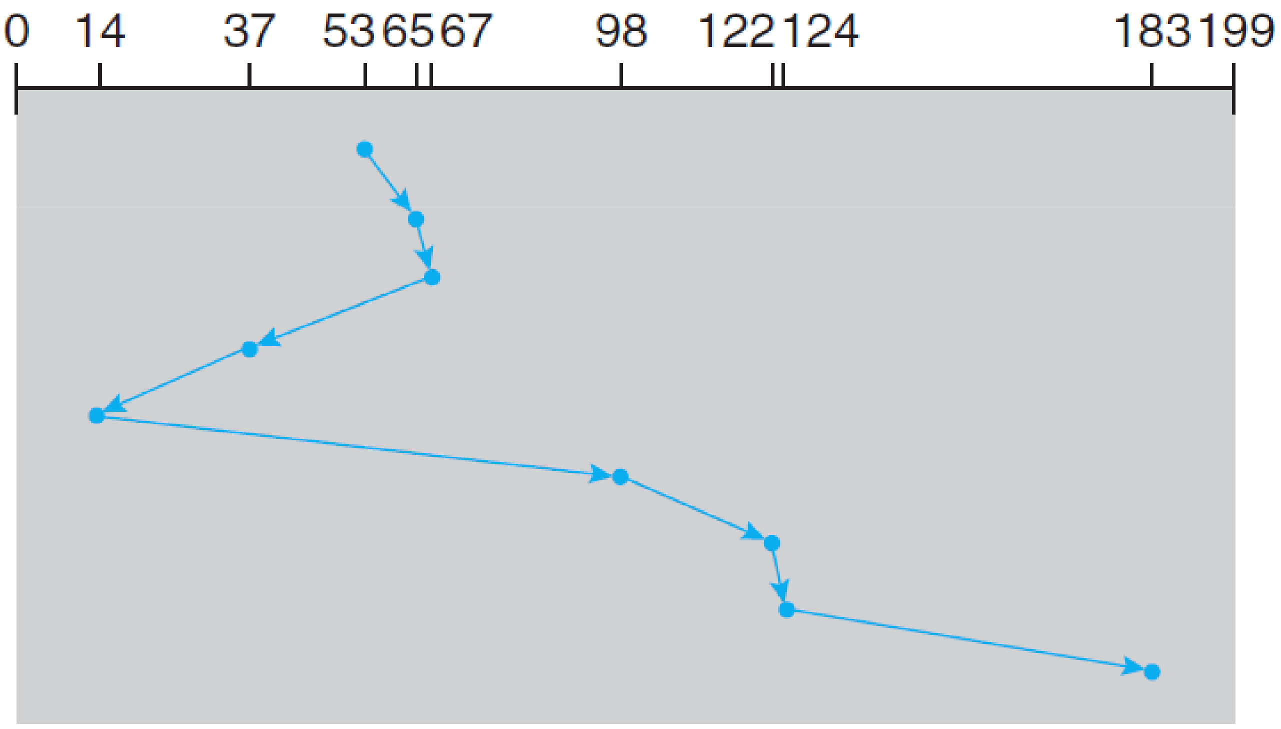
Total head movements: 640 cylinders

Shortest Seek Time First (SSTF)

- SSTF services the request closest to the current head position

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

head starts at 53



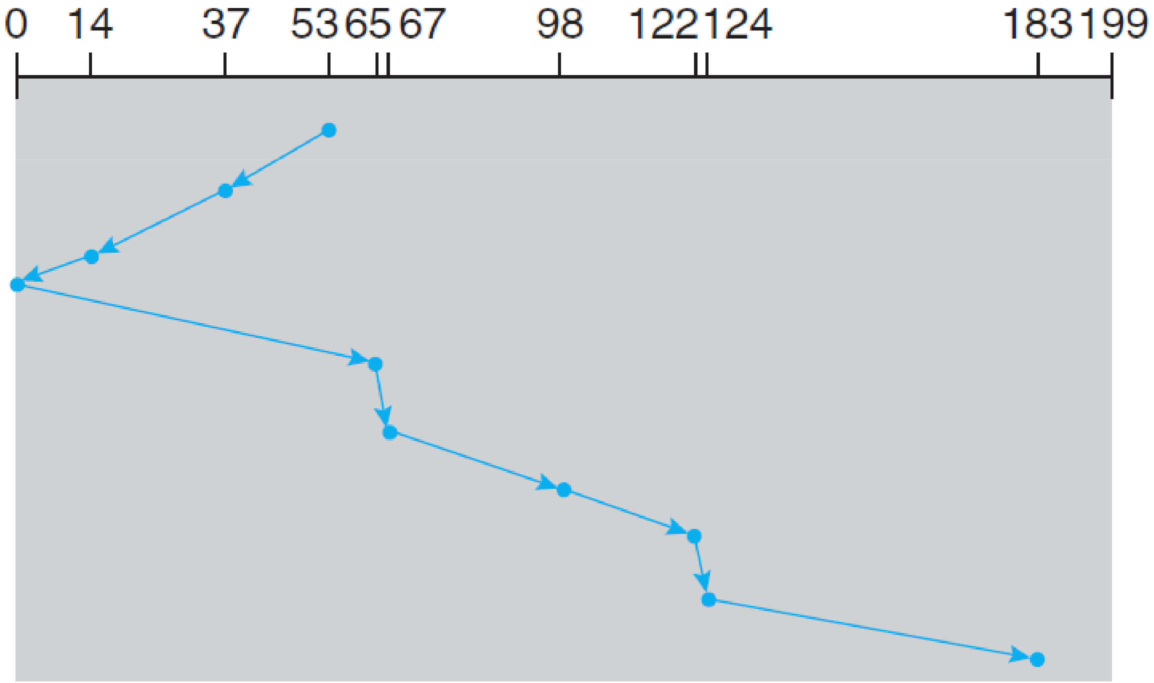
Total head movements: 236 cylinders

SCAN (or Elevator Algorithm)

- SCAN behaves like an elevator in a building, first servicing requests in one way and then reversing to service requests in the other way

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

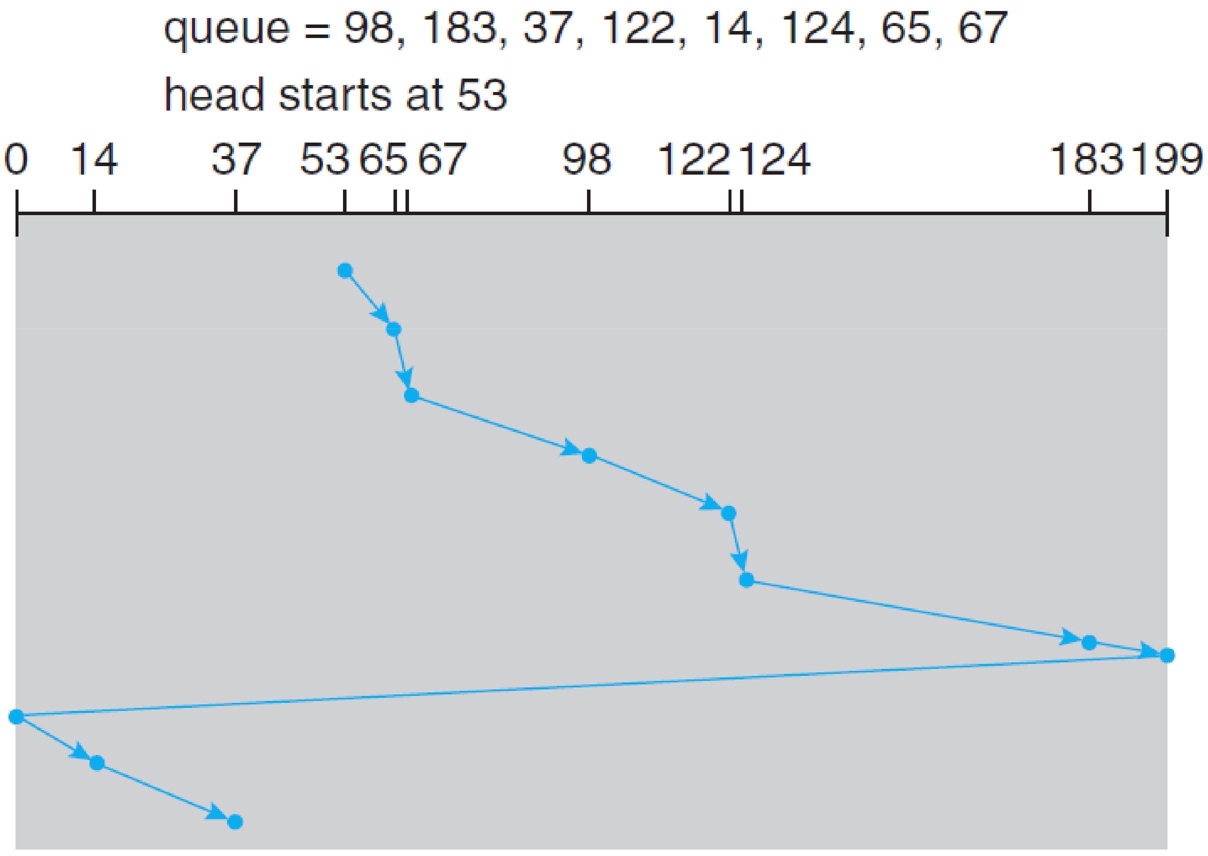
head starts at 53



Total head movements: 236 cylinders

Circular SCAN (C-SCAN)

- C-SCAN works like SCAN but only services requests in one direction



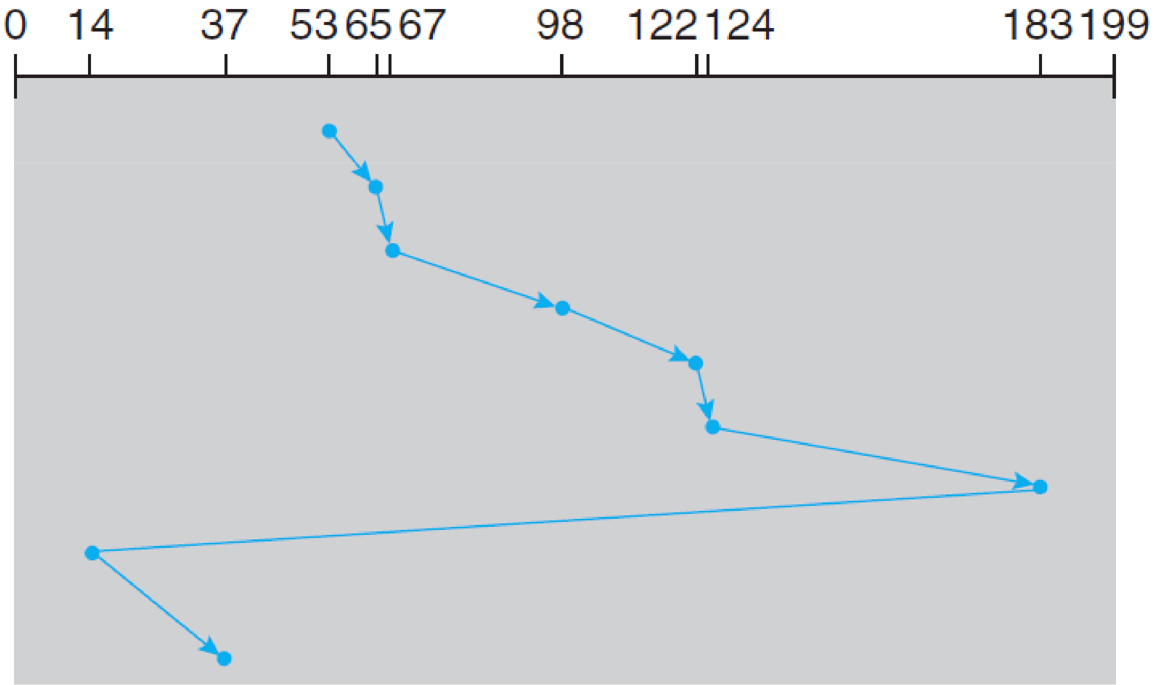
Total head movements: 382 cylinders

C-LOOK (and LOOK)

- Version of C-SCAN (and SCAN) that reverses direction after servicing the last request in each direction, thus avoiding going until the end of the disk

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

head starts at 53



Total head movements: 322 cylinders

Pros and Cons

- FCFS
  - (+) Fair among requesters
  - (–) Order of arrival may lead to very long seeks
- SSTF
  - (+) Reduce seeks
  - (–) May lead to starvation
- SCAN
  - (+) Low seeks and no starvation
  - (–) Favors middle tracks (when the head reverses direction, the higher number of pending requests – assuming a uniform distribution of requests – is at the other end of the disk and those requests will have to wait the longest)

Pros and Cons

- C-SCAN
  - (+) Provides a more uniform wait time than SCAN
  - (–) Longer seeks on the way back
- LOOK & C-LOOK
  - (+) Avoids useless seeks
  - (–) Same as corresponding SCAN version

Disk Scheduling – Discussion

- Given so many algorithms, how do we choose the best one?
  - SSTF is common and has a natural appeal, since it increases performance over FCFS
  - LOOK and C-LOOK perform better for systems that place a heavy load on the disk, since they avoid starvation
- However, performance depends on the number and types of requests
- Requests are also greatly influenced by the file allocation method
  - A program reading a contiguously allocated file will generate several requests that are close together on the disk, resulting in limited head movement
  - In contrast, a linked or indexed file may include blocks that are widely scattered on the disk, resulting in greater head movement

Disk Scheduling – Discussion

- All these algorithms focus on minimizing seek time but, for **modern disks**, the **rotational latency can be nearly as large as the average seek time**
  - For the OS it is difficult to schedule for improved rotational latency, because modern disks do not disclose the physical location of logical blocks
  - Disk manufacturers have been alleviating this problem by implementing disk scheduling algorithms in the controller hardware (OS just needs to send batch of requests to the controller, the controller then queues and schedules them to improve both the seek time and the rotational latency)
  - If I/O performance is the only consideration, the OS can turn over the responsibility of disk scheduling to the disk hardware
  - In practice, however, the OS has constraints on the service order for requests (for instance, demand paging may take priority over application I/O; writes are more urgent than reads if the cache is running out of free pages; …)

SSD Scheduling

- Disk scheduling algorithms focus primarily on minimizing the amount of head movements, but **SSDs do not contain moving disk heads!**
  - Most SSD schedulers **simply use FCFS** order

- However, the observed behavior of SSDs indicates that the time required to service reads is uniform but that the time to service writes, because of the properties of flash memory, is not uniform
  - Some SSD schedulers exploit this characteristic by **merging adjacent write requests**, servicing read requests still in FCFS order