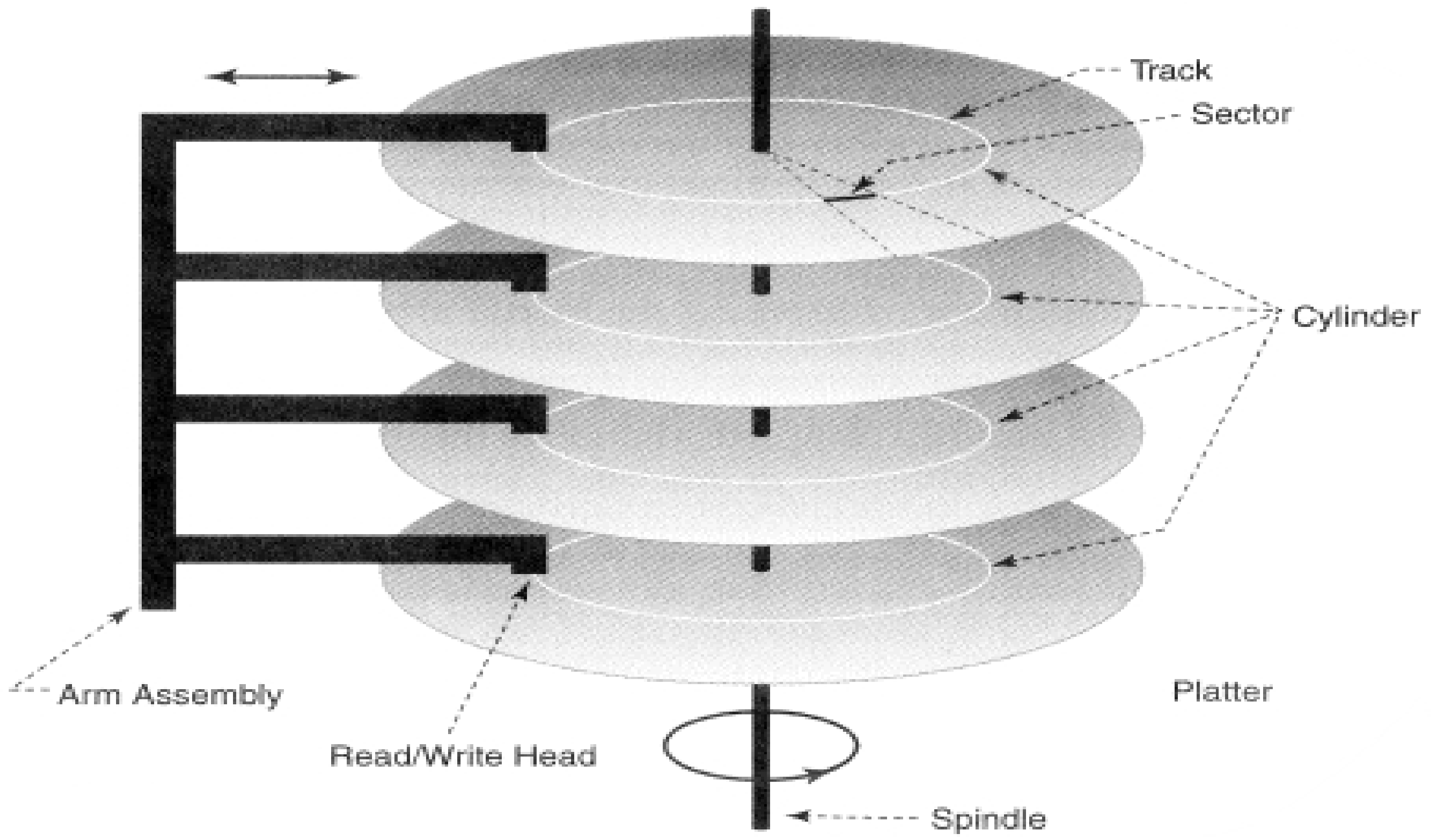
Disk Management

Physical Disk Structure



Disk Structure

- Disk drives are addressed as large 1-dimensional arrays of *logical blocks*, where the logical block is the smallest unit of transfer
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially
  - Sector 0 is the first sector of the first track (top platter) on the outermost cylinder
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost

Disk Access Time

- Two major components
  - *Seek time* is the time for the disk to move the heads to the cylinder containing the desired sector
    - Typically 5-10 milliseconds
  - *Rotational latency* is the additional time waiting for the disk to rotate the desired sector to the disk head
    - Typically, 2-4 milliseconds
- One minor component
  - Read/write time or transfer time – actual time to transfer a block, less than a millisecond

Disk Scheduling

- Should ensure a fast access time and disk bandwidth
- Fast access
  - Minimize total seek time of a group of requests
  - If requests are for different cylinders, average rotation latency has to be incurred for each anyway, so minimizing it is not the primary goal (though some scheduling possible if multiple requests for same cylinder is there)
- Seek time ≈ seek distance
- Main goal : reduce total seek distance for a group of requests
- Auxiliary goal: fairness in waiting times for the requests
- Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer

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

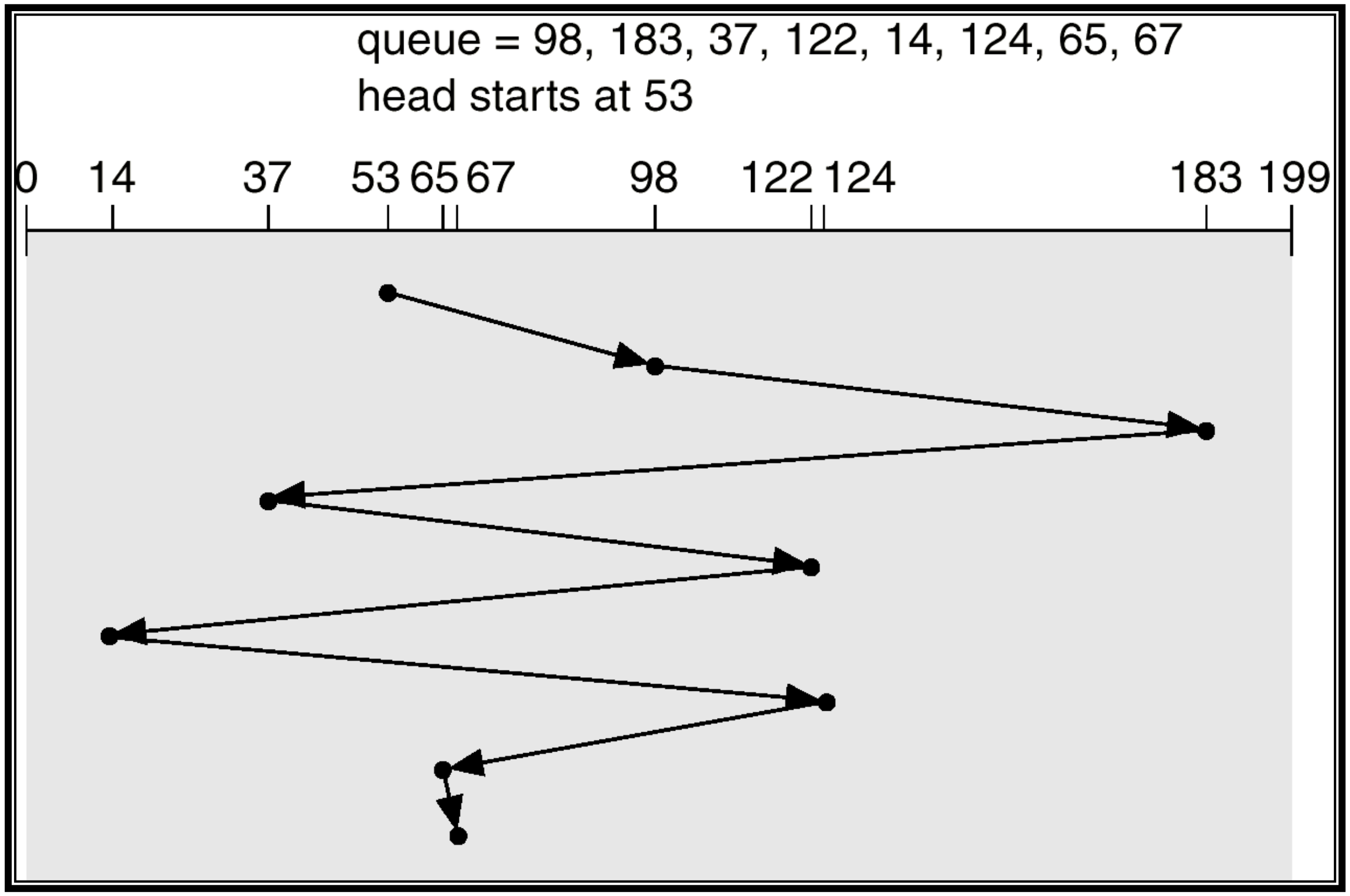
Head pointer 53

FCFS

- Service requests in the order they come
- Fair to all requests
- Can cause very large total seek time over all requests if the load is moderate to high

FCFS

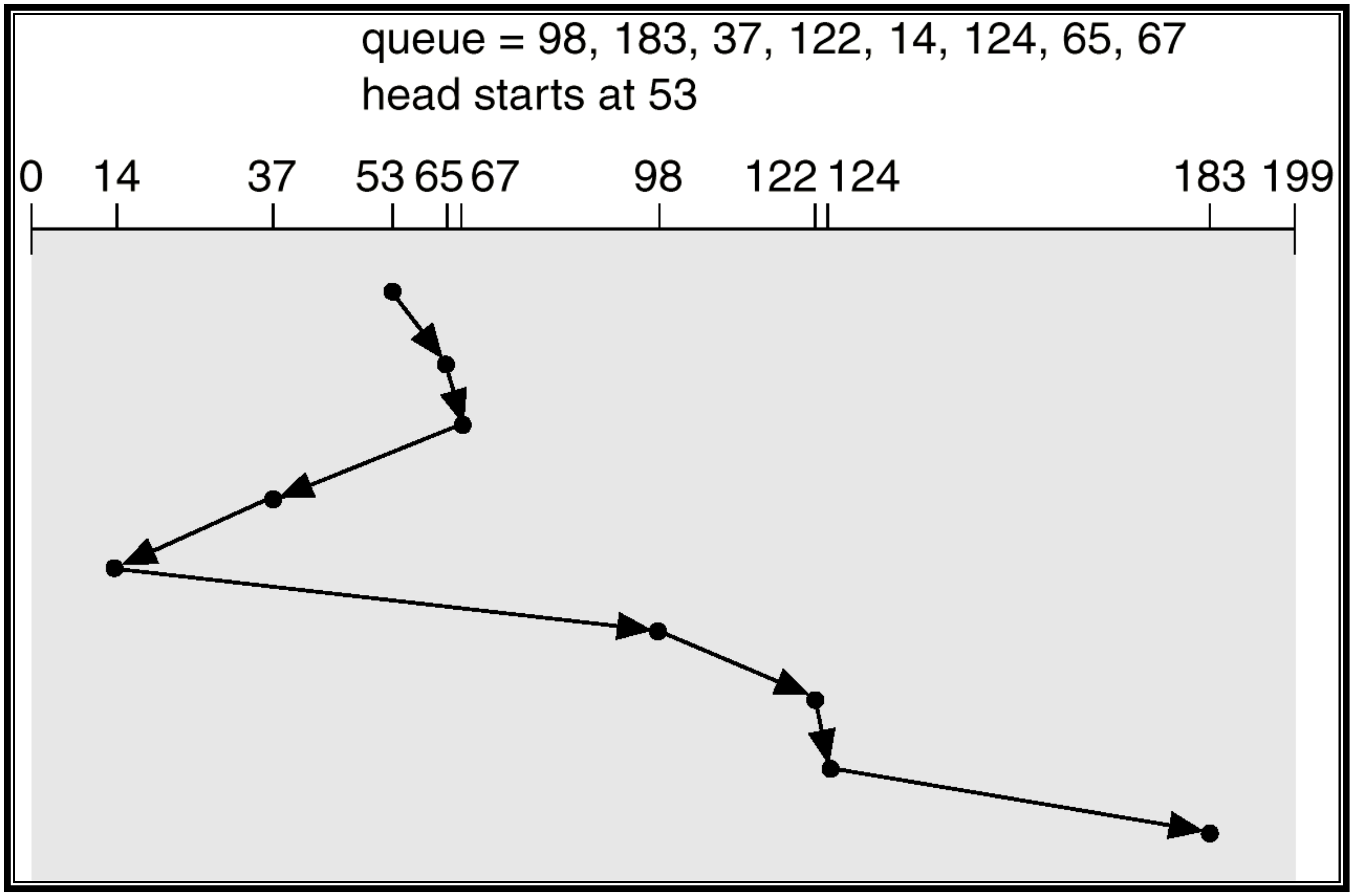
Illustration shows total head movement of 640 cylinders.



SSTF

- Selects the request with the minimum seek time from the current head position
- SSTF scheduling is a form of SJF scheduling
  - May cause starvation of some requests like SJF
  - But not optimal, unlike SJF
- Minimizes seek time, but not fair
- May work well if the load is not high

SSTF (Cont.)

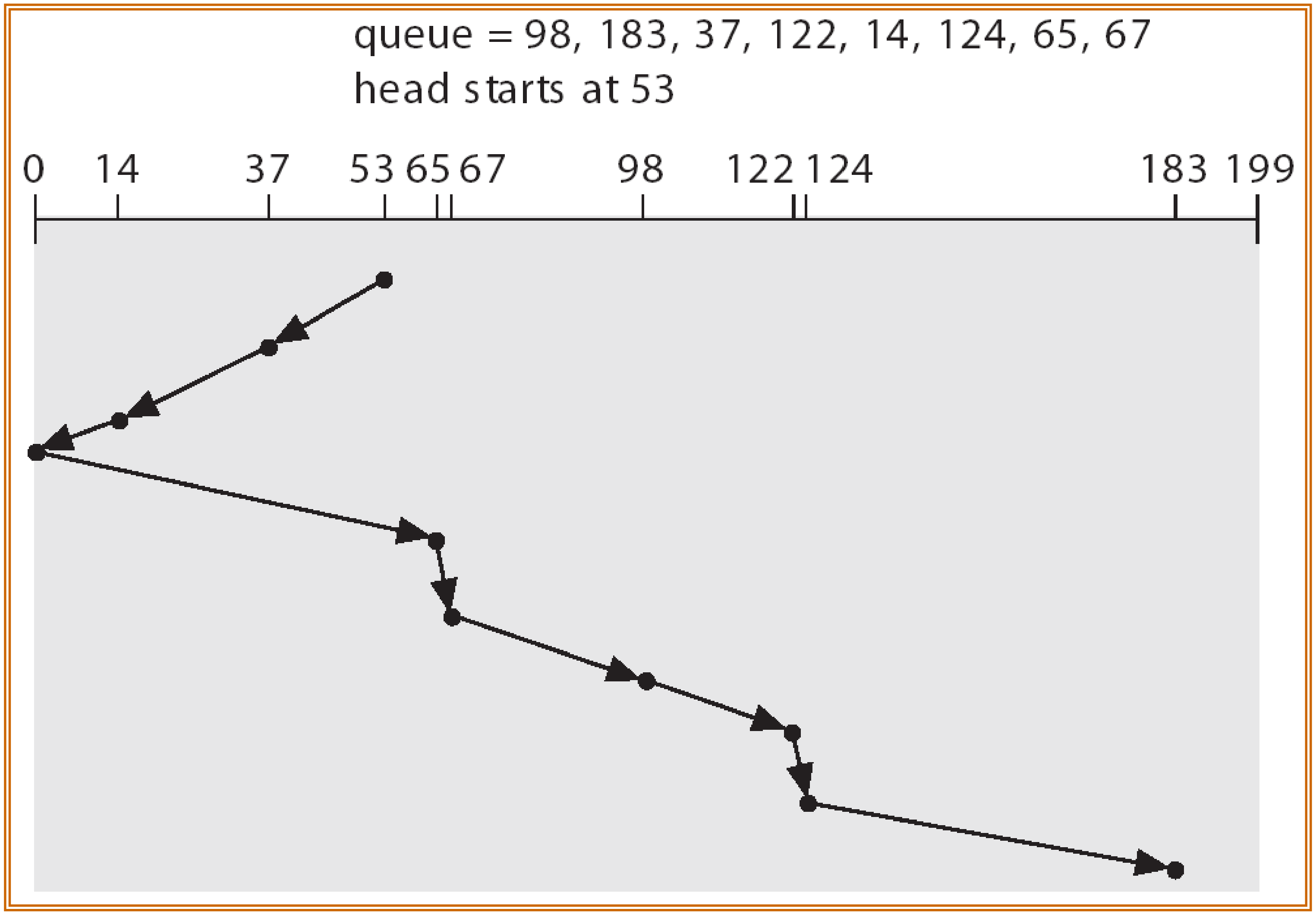


Total head movement = 236 cylinders

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*

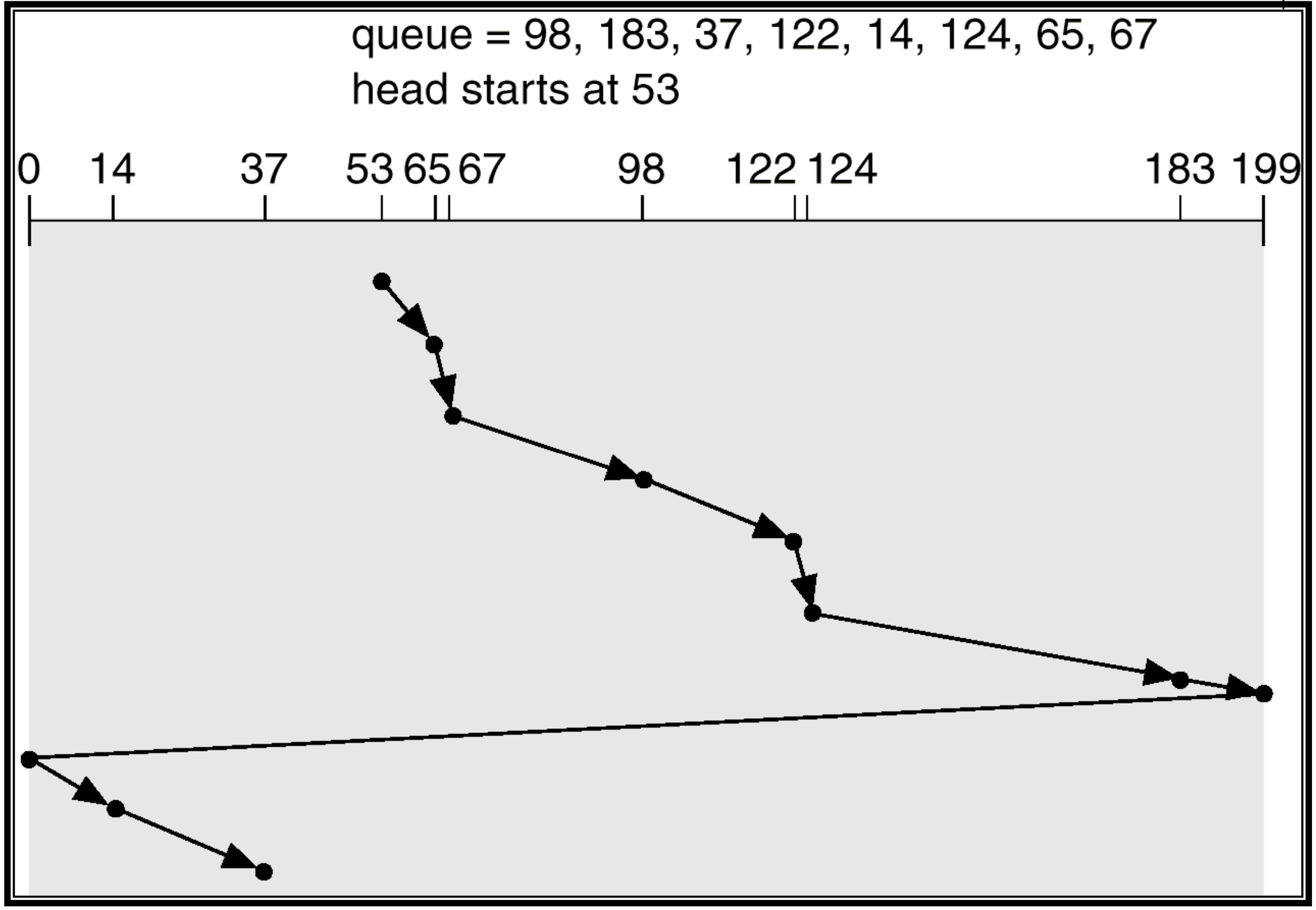
SCAN (Cont.)


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

- 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

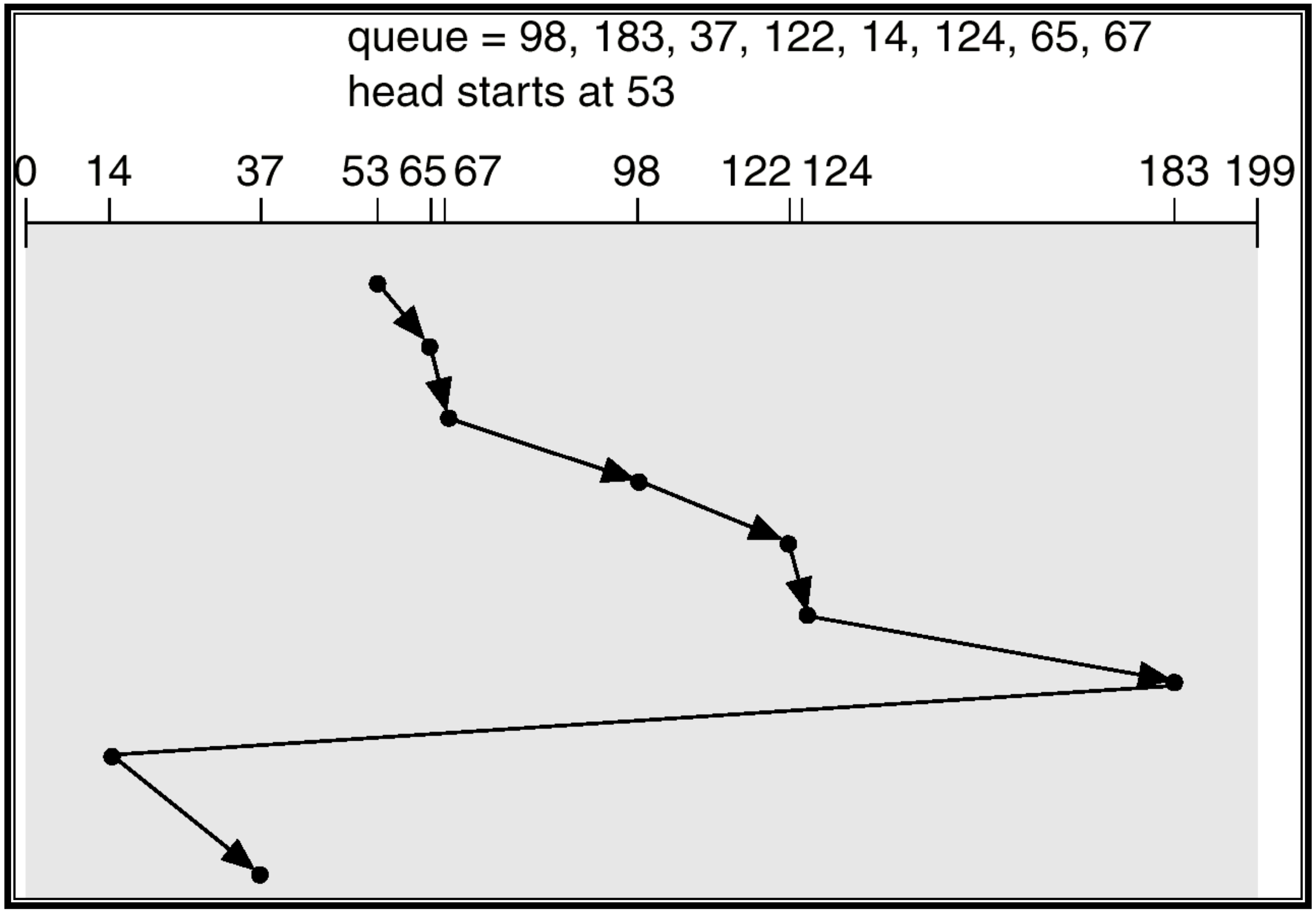
C-SCAN (Cont.)


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-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.

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

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 the disk
- Performance depends on the number and types of requests
- Requests for disk service can be influenced by the file-allocation method
- 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
- Either SSTF or C-LOOK is a reasonable choice for the default algorithm (depending on load)

Disk Management

- *Low-level formatting*, or *physical formatting* — Dividing a disk into sectors that the disk controller can read and write.
- To use a disk to hold files, the operating system still needs to record its own data structures on the disk
  - *Partition* the disk into one or more groups of cylinders
  - *Logical formatting* or "making a file system"
- Boot block initializes system
  - The bootstrap is stored in ROM
  - *Bootstrap loader* program
- Methods such as *sector sparing* used to handle bad blocks

Operating System Issues

- Major OS jobs are to manage physical devices and to present a virtual machine abstraction to applications
- For hard disks, the OS provides two abstraction:
  - Raw device – an array of data blocks.
  - File system – the OS queues and schedules the interleaved requests from several applications.

Application Interface

- Most OSs handle removable disks almost exactly like fixed disks — a new cartridge is formatted and an empty file system is generated on the disk.
- Tapes are presented as a raw storage medium, i.e., and application does not not open a file on the tape, it opens the whole tape drive as a raw device
- Usually the tape drive is reserved for the exclusive use of that application
- Since the OS does not provide file system services, the application must decide how to use the array of blocks
- Since every application makes up its own rules for how to organize a tape, a tape full of data can generally only be used by the program that created it