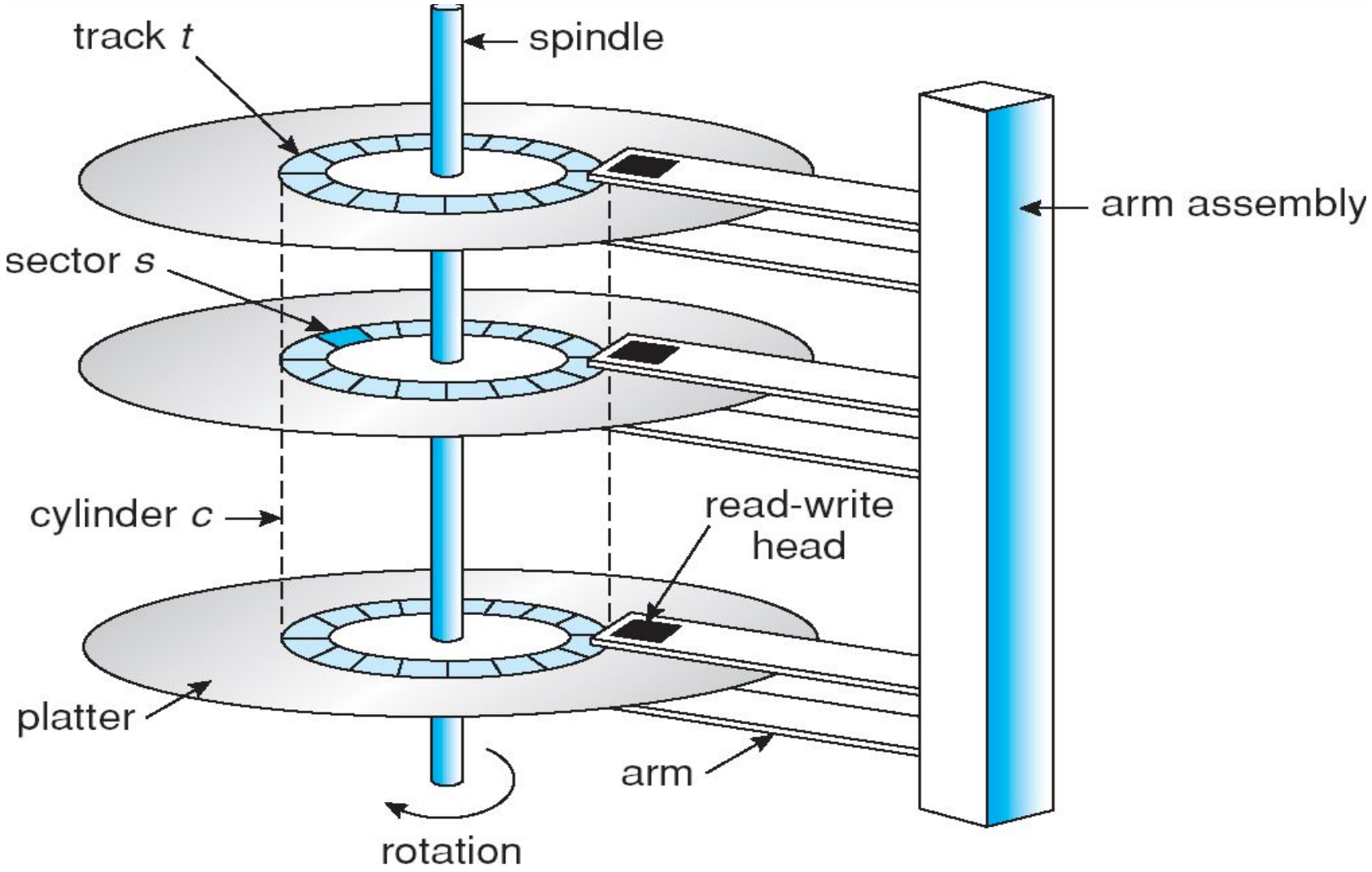
Operating Systems

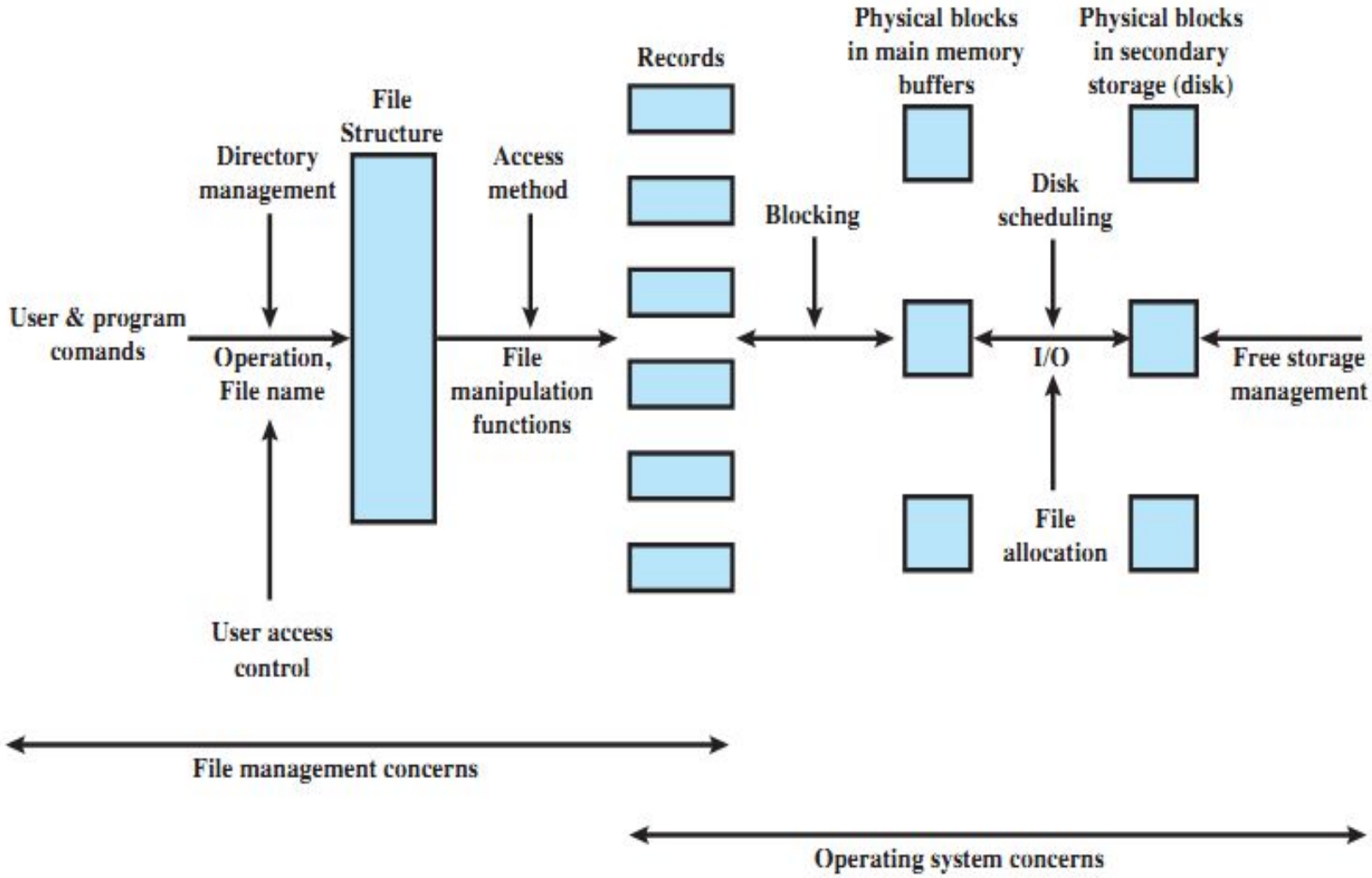
Disk Scheduling

A. Frank - P. Weisberg

Moving-head Disk Mechanism



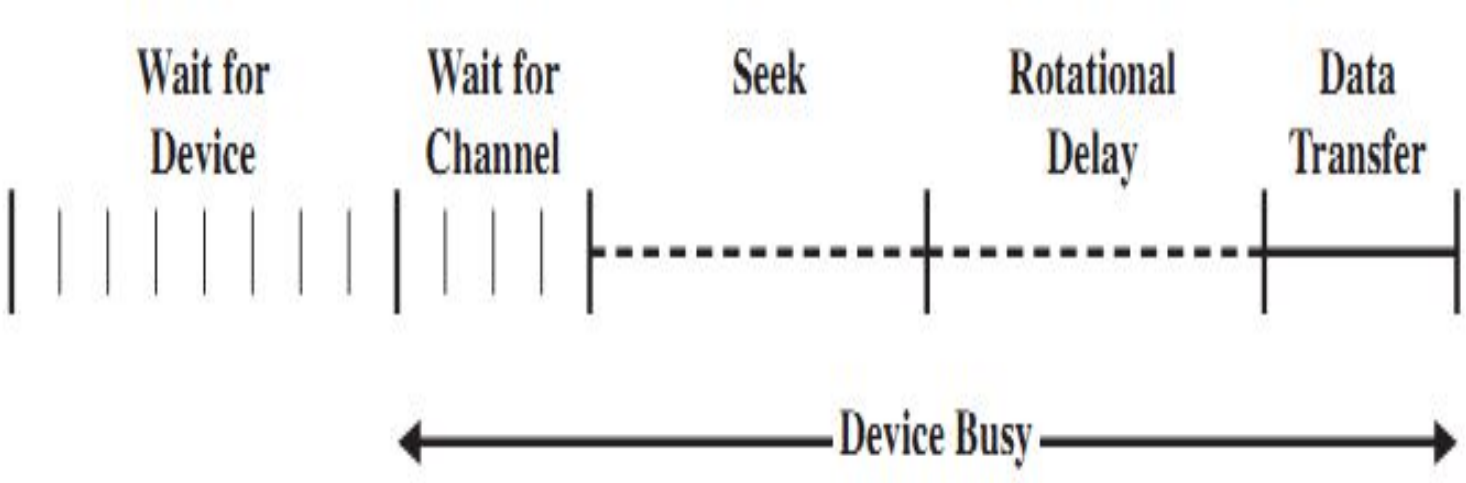
Elements of File Management



(Disk Scheduling (1

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth.
- Access time has two major components:
  - Seek time is the time for the disk are to move the heads to the cylinder containing the desired sector.
  - Rotational latency is the additional time waiting for the disk to rotate the desired sector to the disk head.
- Minimize seek time ≈ seek distance.
- Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of last transfer.

Components of Disk I/O Transfer



(Disk Scheduling (2

- There are many sources of disk I/O request:
  - OS
  - System processes
  - Users processes
- I/O request includes input/output mode, disk address, memory address, number of sectors to transfer.
- OS maintains queue of requests, per disk or device.
- Idle disk can immediately work on I/O request, busy disk means work must queue:
  - Optimization algorithms only make sense when a queue exists.

Disk Scheduling Algorithms

- Note that drive controllers have small buffers and can manage a queue of I/O requests (of varying “depth”).
- Several algorithms exist to schedule the servicing of disk I/O requests.
- The analysis is true for one or many platters.
- We illustrate them with a I/O request queue (cylinders are between 0-199):

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

head starts at 53

First Come First Serve (FCFS) Example

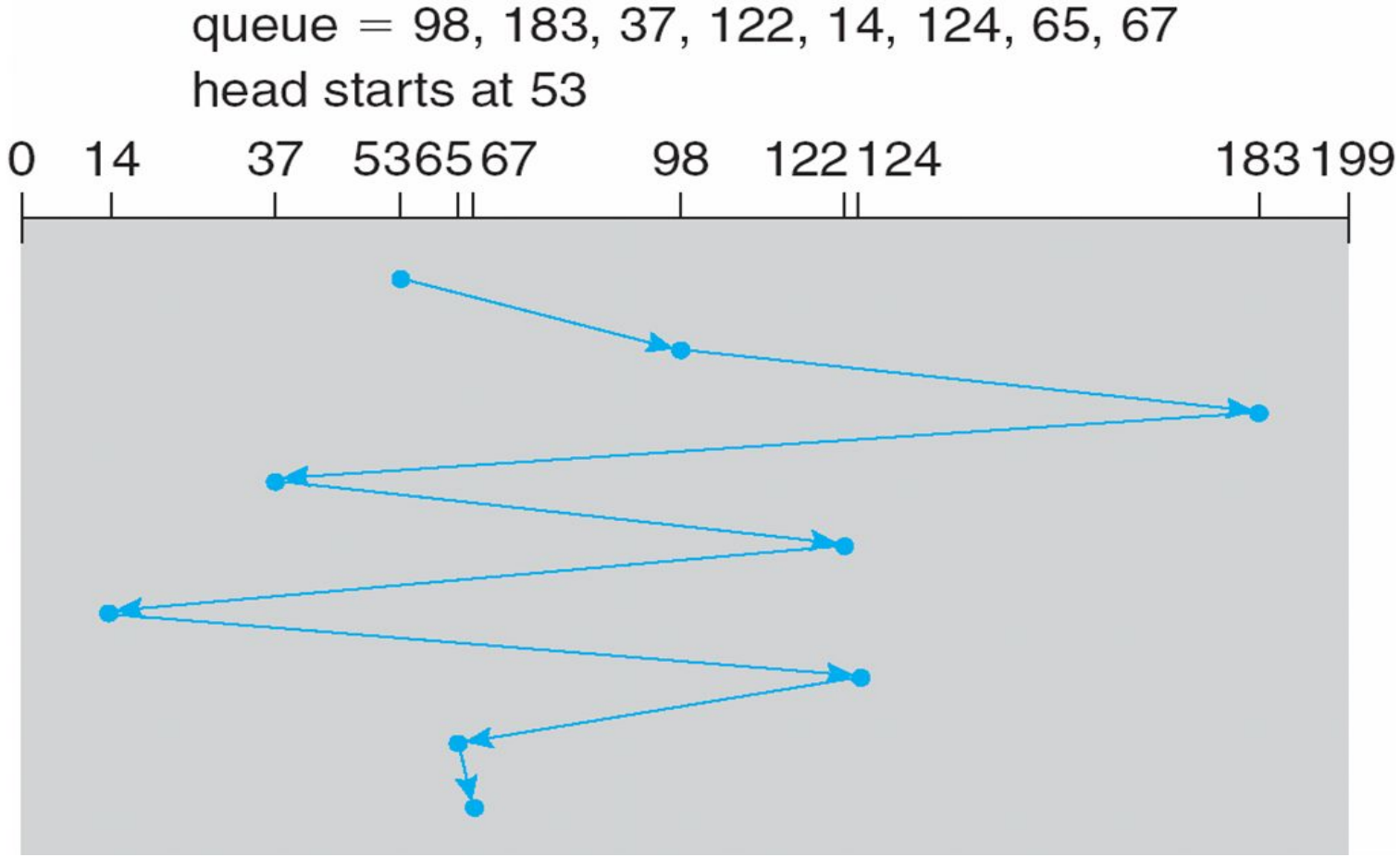


Illustration shows total head movement of 640 cylinders.

(First Come First Serve (FCFS

- Handle I/O requests sequentially.
- Fair to all processes.
- Approaches random scheduling in performance if there are many processes/requests.
- Suffers from global zigzag effect.

Shortest Seek Time First (SSTF) Example

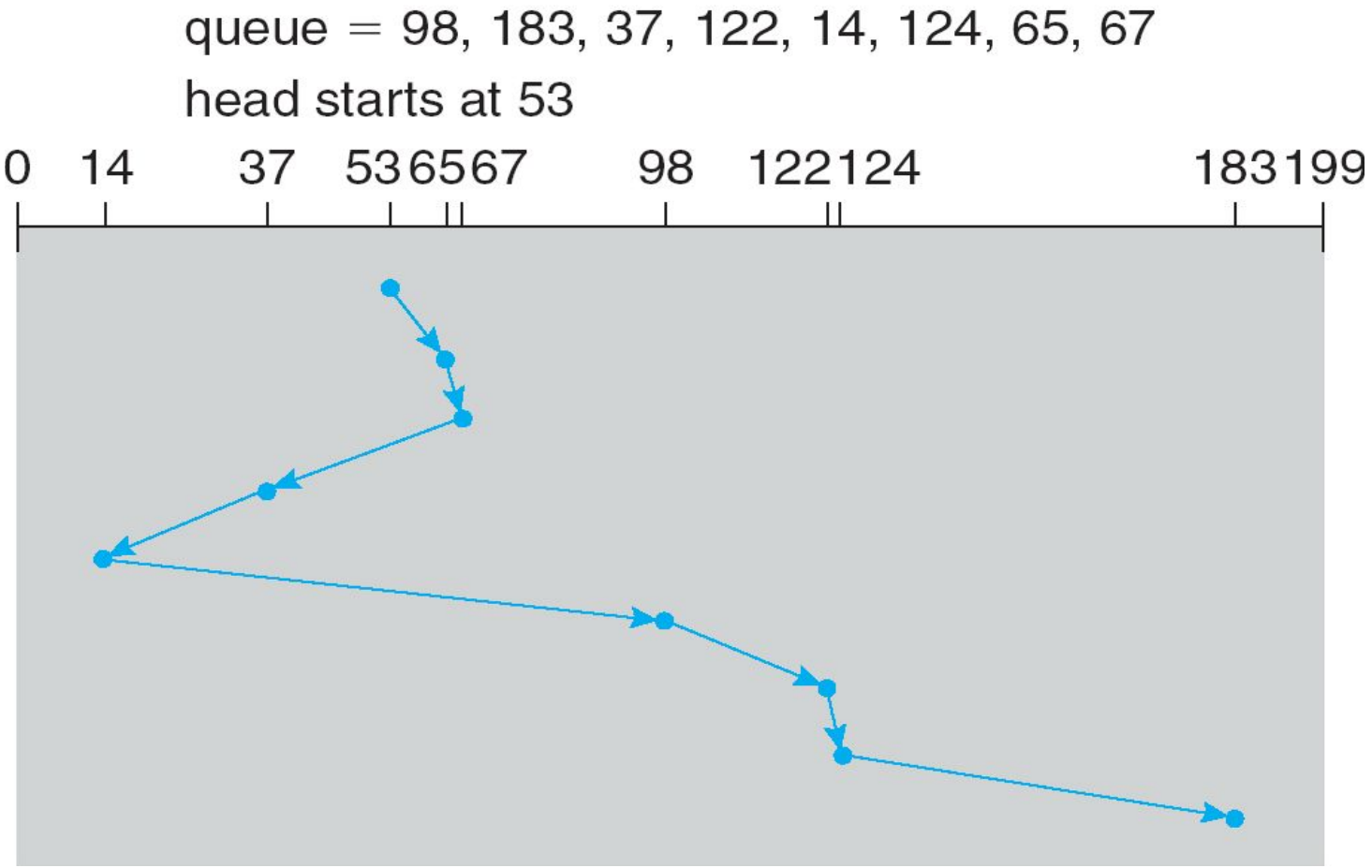


Illustration shows total head movement of 236 cylinders.

(Shortest Seek Time First (SSTF

- Selects the request with the minimum seek time from the current head position.
- Also called Shortest Seek Distance First (SSDF) – It's easier to compute distances.
- It's biased in favor of the middle cylinders requests.
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.

Elevator Algorithms

- Algorithms based on the common elevator principle.
- Four combinations of Elevator algorithms:

.Service in both directions or in only one direction –

.Go until last cylinder or until last I/O request –

Direction / Go until	Go until the last cylinder	Go until the last request
Service both directions	**Scan**	**Look**
Service in only one direction	**C-Scan**	**C-Look**

Scan Example

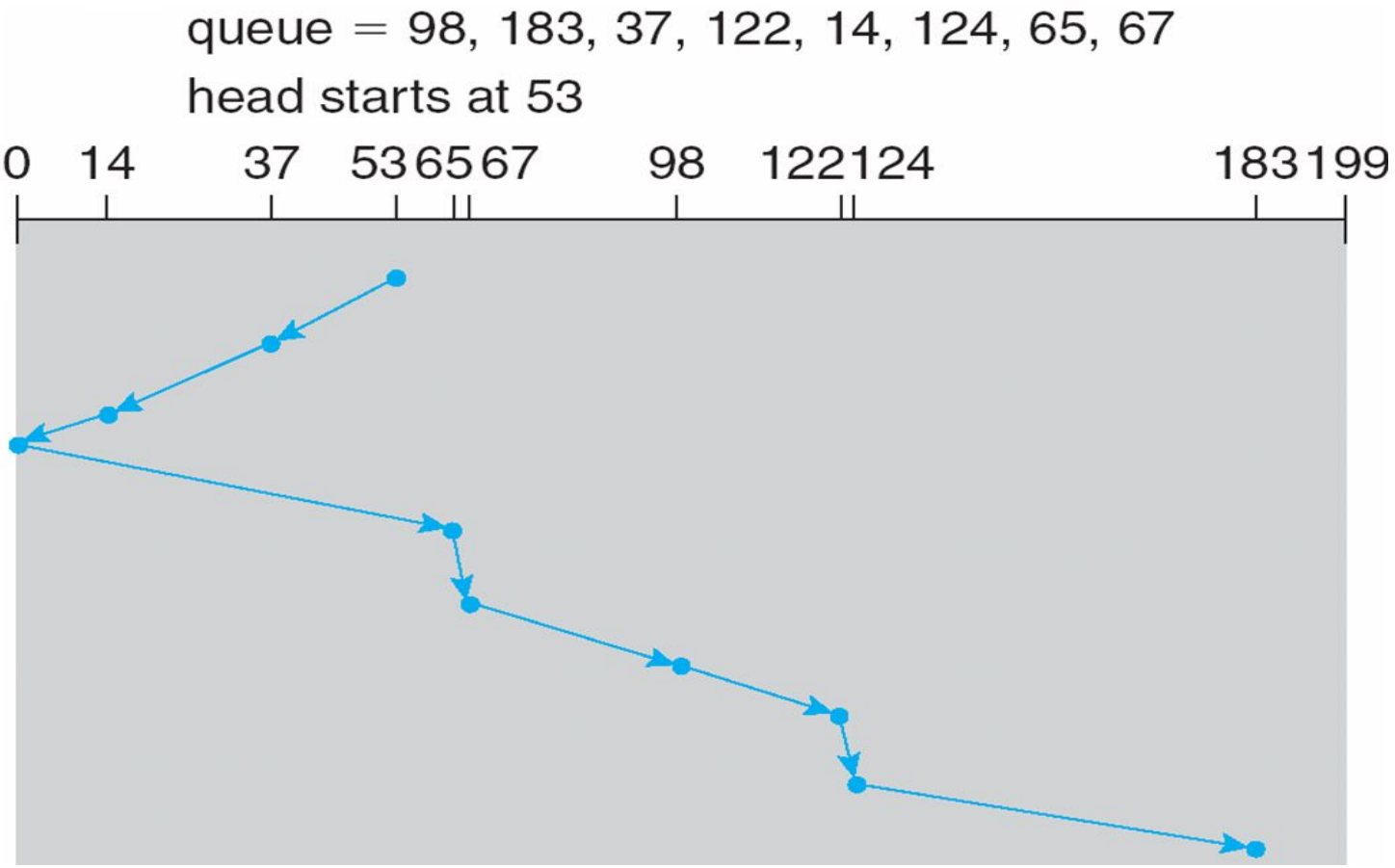


Illustration shows total head movement of 208 cylinders.

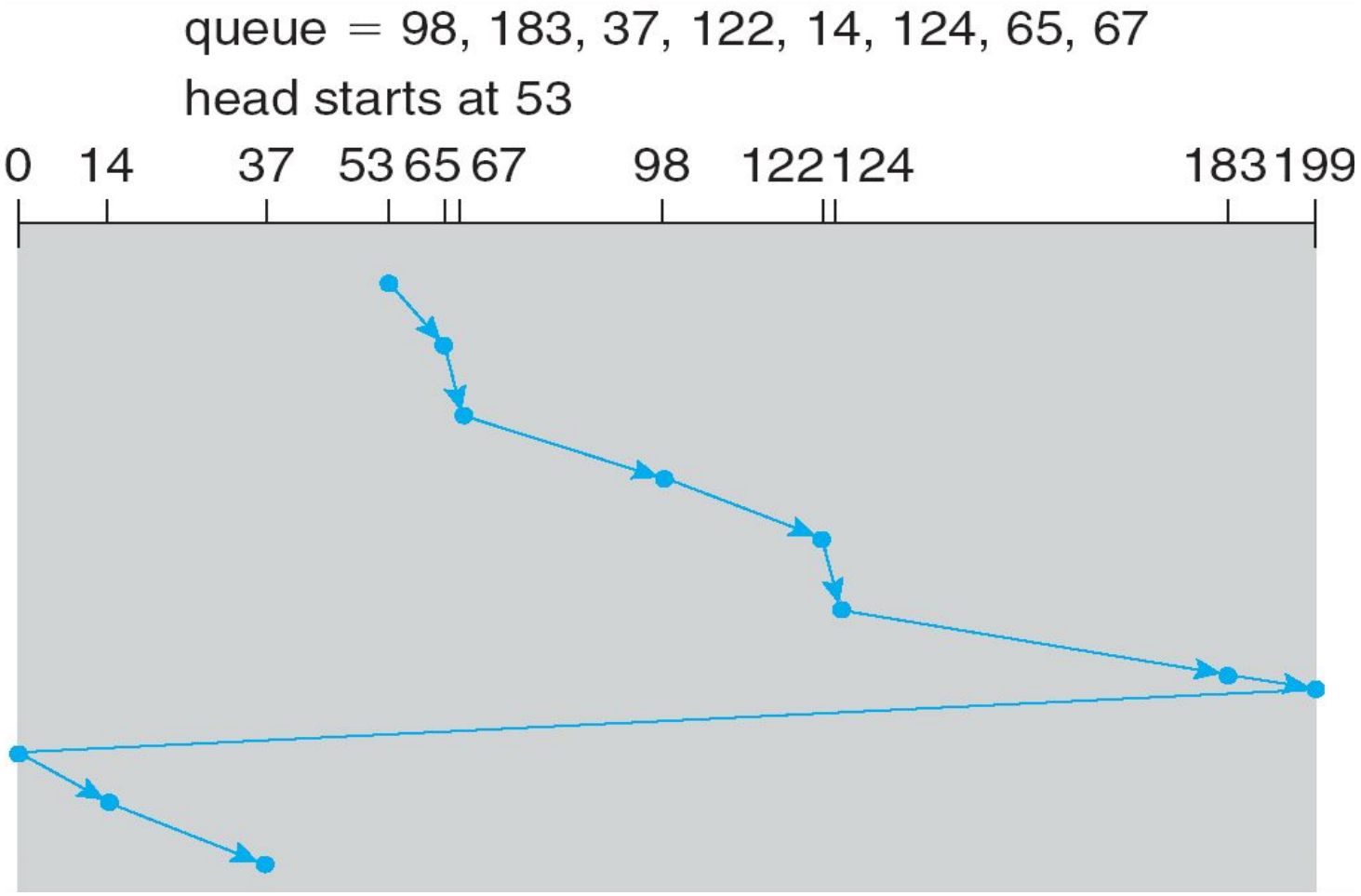
Scan

- 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.
- It moves in both directions until both ends.
- Tends to stay more at the ends so more fair to the extreme cylinder requests.

Look

- The disk arm starts at the first I/O request on the disk, and moves toward the last I/O request on the other end, servicing requests until it gets to the other extreme I/O request on the disk, where the head movement is reversed and servicing continues.
- It moves in both directions until both last I/O requests; more inclined to serve the middle cylinder requests.

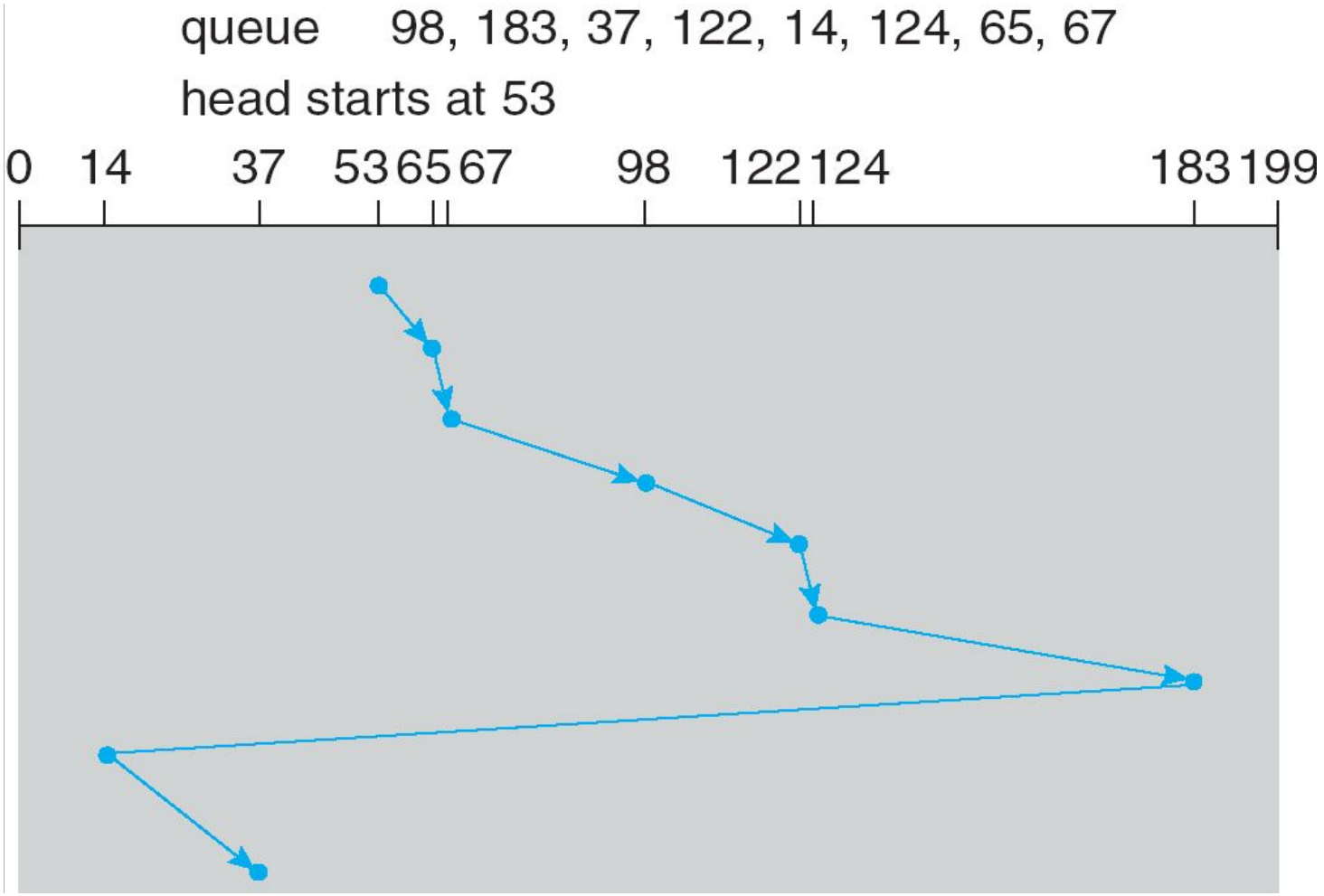
C-Scan Example



C-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.
- Provides a more uniform wait time than SCAN; it treats all cylinders in the same manner.

C-Look Example



C-Look

- 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.
- In general, Circular versions are more fair but pay with a larger total seek time.
- Scan versions have a larger total seek time than the corresponding Look versions.

Another Example

(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) LOOK (starting at track 100, in the direction of increasing track number)		(d) C-LOOK (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

Linux Scheduler

- Merging of successive requests
  - Front merge: if a new request immediately precedes an existing request
  - Back merge: if a new request immediately succeeds an existing request
- Inserting
  - Scheduling

Linux Scheduler

- Merging of successive requests
  - Front merge: if a new request immediately precedes an existing request
  - Back merge: if a new request immediately succeeds an existing request
- Inserting
  - Scheduling
  - Primarily uses elevator algorithms

Linux Scheduler

- Merging of successive requests
  - Front merge: if a new request immediately precedes an existing request
  - Back merge: if a new request immediately succeeds an existing request
- Inserting
  - Scheduling
  - Primarily uses elevator algorithms

(Linus Elevator (Used in v2.4

- Suppose you get a request R
- Try to merge a request
- If merge fails:
  - Is there any request that is older than T?
    - If yes, then add R to the end of queue
    - If no, then insert at a position so that it is in the “right” sequence
    - If right sequence not found, then add R to the end of queue

(Linus Elevator (Used in v2.4

- Advantages
  - Relatively simple
  - Usually provides decent throughput
- Disadvantages
  - Age checking is very random
  - Starvation is possible

(Deadline Scheduler (Used in v2.6

- One feature:
  - Reads are usually more urgent than writes
  - Why?
- Traditional elevator scheduling do not consider this aspect

(Deadline Scheduler (Used in v2.6

- One feature:
  - Reads are usually more urgent than writes
  - Why?
- Traditional elevator scheduling do not consider this aspect

(Deadline Scheduler (Used in v2.6

- All requests are assigned a deadline
  - Read requests: 500 ms
  - Write requests: 5 s
- Maintains three queues
  - For normal sector-wise operation
  - For FIFO read operation
  - For FIFO write operation
- If any FIFO queue's request expires, then handle all the requests of that queue

Disadvantage of Deadline Scheduler

- If there are frequent writes
  - Immediately after handling FIFO write request, can switch to FIFO read queue
  - Can lead to a major fall in throughput
- Anticipatory Scheduler
  - Waits for some few milliseconds after handling a read request
  - If a read/write request comes close to this sector during waiting period, handle it

Disadvantage of Deadline Scheduler

- If there are frequent writes
  - Immediately after handling FIFO write request, can switch to FIFO read queue
  - Can lead to a major fall in throughput
- Anticipatory Scheduler
  - Waits for some few milliseconds after handling a read request
  - If a read/write request comes close to this sector during waiting period, handle it

Completely Fair Scheduling

- In modern desktop systems, usually operations are not very disk intensive
- Maintain one queue for each process
- Within a queue, try to merge and insert operations
- Default for desktop systems

(Selecting a Disk-Scheduling Algorithm (2

- With low load on the disk, It's FCFS anyway.
- SSTF is common and has a natural appeal – good for medium disk load.
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk; Less starvation.
- Performance depends on number and types of requests.
- Requests for disk service can be influenced by the file-allocation method and metadata layout.
- Either SSTF or LOOK (as part of an Elevator package) is a reasonable choice for the default algorithm.