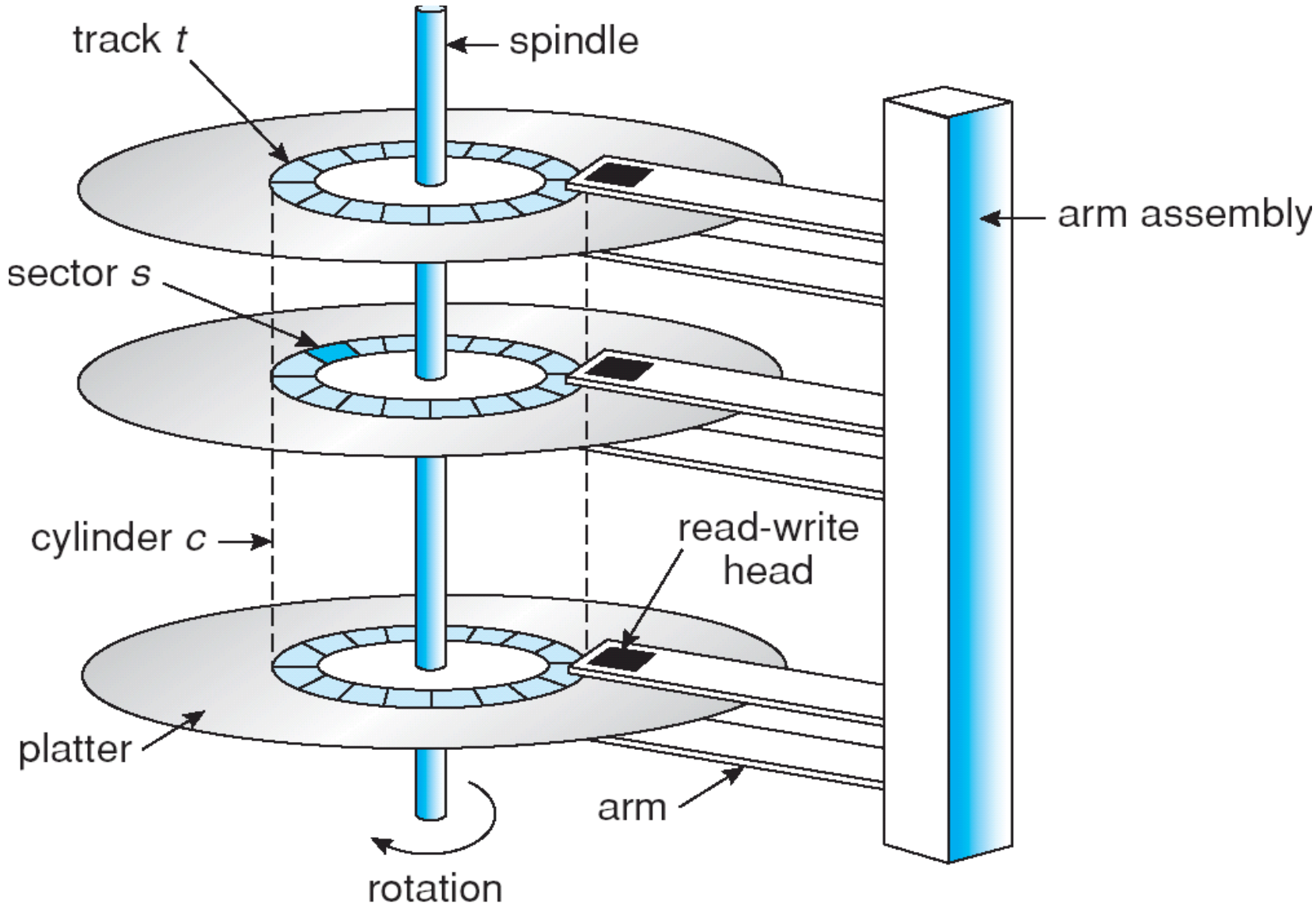
Secondary-Storage Structure

We discuss the physical structure of secondary storage device.

Magnetic Disks

Magnetic disks provide the bulk of secondary storage for modern computer systems.

The following diagram shows the internal structure of magnetic disk.


track t
spindle
arm assembly
sector s
cylinder c
read-write
head
platter
arm
rotation

A magnetic disk consists of a number of platters.

Each disk platter has a flat circular shape, like a CD.

Diameter of platter ranges from 1.8 to 5.25 inches.

The two surfaces of a platter are covered with a magnetic material.

Information is stored by recording it magnetically on the platters.

A read -write head "flies" just above each surface of every platter.

The heads are attached to a *disk arm* that moves all the heads as a unit.

The surface of a platter is divided into circular *tracks.*

Each track is subdivided into hundreds of sectors.

The set of tracks that are at one arm position makes up a *cylinder.*

There may be thousands of cylinders in a disk.

When the disk is in use, a drive motor spins it at high speed (60 to 200 times per second).

The time required to access data from the disk is called *access time*.

The *access time* consists of the time necessary to move the disk arm to the desired cylinder, called the *seek time* and the time necessary for the desired sector to rotate to the disk head, called the *rotational latency.*

Disks can transfer several megabytes of data per second, and they have *seek times* and *rotational latencies* of several milliseconds.

Because the disk head flies on an extremely thin cushion of air, there is a danger that the head will make contact with the disk surface and damage the magnetic surface.

This accident is called a *head crash.*

A *head crash* cannot be repaired; the entire disk must be replaced.

A disk drive is attached to a computer by a set of wires called an *I/O bus.*

Several kinds of buses are available, including *Enhanced Integrated Drive Electronics (EIDE)*, *Advanced Technology Attachment (ATA), Serial ATA (SATA), Universal Serial Bus (USB), Fiber Channel (FC)* and *Small Computer Systems Interface (SCSI)* buses.

The data transfers on a bus are carried out by special electronic processors called *controllers.*

The *host controller* is the controller at the computer end of the bus.

A *disk controller* is built into each disk drive.

To perform a disk I/O operation, the computer places a command into the *host controller*.

The *host controller* then sends the command via messages to the *disk controller*, and the disk controller operates the disk-drive hardware to carry out the command.

1) Average disk access time = Average Seek time + Average Rotational delay or latency + Transfer time

2) Average seek time = (k - 1) x t / 2

‘k’ is the number of tracks per surface
‘t’ is the time taken by head to move from one track to adjacent track

3) Average rotational latency = 1 / 2 x Time taken for full rotation

4) Capacity of a disk pack = Number of surfaces x Number of tracks per surface x Number of sectors per track x Storage capacity of one sector

5) Data transfer rate = Number of heads x Capacity of one track x Number of rotations in one second

6) Transfer time = Number of bytes to be transferred / Data transfer rate

7) Formatting overhead = Number of sectors x Overhead per sector

8) Formatted disk space = Total disk space or capacity – Formatting overhead

Consider a disk pack with the following specifications - 16 surfaces, 128 tracks per surface, 256 sectors per track and 512 bytes per sector.

Answer the following questions-

1. What is the capacity of disk pack?
2. If the disk is rotating at 3600 RPM, what is the data transfer rate?
3. If the disk system has rotational speed of 3000 RPM, what is the average access time with a seek time of 11.5 msec?
4. If the format overhead is 32 bytes per sector, what is the formatted disk space?

Capacity of disk pack = Number of surfaces x Number of tracks per surface x Number of sectors per track x Number of bytes per sector
= 16 x 128 x 256 x 512 bytes = 2^{28} bytes = 256 MB

Number of rotations in one second = (3600 / 60) rotations/sec = 60 rotations/sec

Data transfer rate = Number of heads x Capacity of one track x Number of rotations in one second
= 16 x (256 x 512 bytes) x 60 = 2^4 x 2^8 x 2^9 x 60 bytes/sec = 60 x 2^{21} bytes/sec = 120 MBps

Time taken for one full rotation = (60 / 3000) sec = (1 / 50) sec = 0.02 sec = 20 msec

Average rotational delay = 1/2 x Time taken for one full rotation
= 1/2 x 20 msec = 10 msec

Average access time = Average seek time + Average rotational delay + Other factors
= 11.5 msec + 10 msec + 0
= 21.5 msec

Formatting overhead = Total number of sectors x overhead per sector
= (16 x 128 x 256) x 32 bytes = 2^{19} x 2^{5} bytes = 2^{24} bytes = 16 MB

Formatted disk space = Total disk space – Formatting overhead
= 256 MB – 16 MB = 240 MB

What is the average access time for transferring 512 bytes of data with the following specifications-

- Average seek time = 5 msec
- Disk rotation = 6000 RPM
- Data rate = 40 KB/sec

Time taken for one full rotation = (60 / 6000) sec = (1 / 100) sec = 0.01 sec = 10 msec

Average rotational delay = 1/2 x Time taken for one full rotation = 1/2 x 10 msec = 5 msec

Transfer time = Number of bytes to be transferred / Data transfer rate = (512 bytes / 40 KB) sec = 0.0125 sec = 12.5 msec

Average access time = Average seek time + Average rotational delay + Transfer time
= 5 msec + 5 msec + 12.5 msec = 22.6 msec

A certain moving arm disk storage with one head has the following specifications-

- Number of tracks per surface = 200
- Disk rotation speed = 2400 RPM
- Track storage capacity = 62500 bits
- Average latency = P msec
- Data transfer rate = Q bits/sec

What is the value of P and Q?

Time taken for one full rotation = (60 / 2400) sec = (1 / 40) sec = 0.025 sec = 25 msec

Average latency or Average rotational latency = 1/2 x Time taken for one full rotation = 1/2 x 25 msec = 12.5 msec

Data transfer rate = Number of heads x Capacity of one track x Number of rotations in one second
= 1 x 62500 bits x (2400 / 60) = 2500000 bits/sec = 2.5×10^6 bits/sec

Thus, P = 12.5 and Q = 2.5×10^6

Consider a typical disk that rotates at 15000 RPM and has a transfer rate of 50×10^6 bytes/sec. If the average seek time of the disk is twice the average rotational delay. What is the average time (in milliseconds) to read or write a 512 byte sector of the disk?

Time taken for one full rotation = (60 / 15000) sec = 0.004 sec = 4 msec

Average rotational delay = 1/2 x Time taken for one full rotation = 1/2 x 4 msec = 2 msec

Average seek time = 2 x Average rotational delay = 2 x 2 msec = 4 msec

Transfer time = Number of bytes to be transferred / Data transfer rate = 512 bytes / (50×10^6 bytes/sec) = 10.24×10^{-6} sec = 0.01024 msec

Average time to read or write 512 bytes = Average seek time + Average rotational delay + Transfer time
= 4 msec + 2 msec + 0.01024 msec = 6.11 msec

A hard disk system has the following parameters-

- Number of tracks = 500
- Number of sectors per track = 100
- Number of bytes per sector = 500
- Time taken by the head to move from one track to another adjacent track = 1 msec
- Rotation speed = 600 RPM

What is the average time taken for transferring 250 bytes from the disk?

Average seek time = (500 -1) x 1 msec / 2 = 249.5 msec

Time taken for one full rotation = (60 / 600) sec = 0.1 sec = 100 msec

Average rotational delay = 1/2 x Time taken for one full rotation = 1/2 x 100 msec = 50 msec

Capacity of one track = Number of sectors per track x Number of bytes per sector = 100 x 500 bytes = 50000 bytes

Data transfer rate = Number of heads x Capacity of one track x Number of rotations in one second
= 1 x 50000 bytes x (600 / 60) = = 50000 x 10 bytes/sec = 5×10^5 bytes/sec

Transfer time = (250 bytes / 5×10^5 bytes) sec = 50×10^{-5} sec = 0.5 msec

Average time taken to transfer 250 bytes = Average seek time + Average rotational delay + Transfer time
= 249.5 msec + 50 msec + 0.5 msec = 300 msec

Mapping physical sectors to logical blocks

In the physical view of disk, the space in disk is divided into sectors. Sector is the smallest unit of transfer.

In the logical view of disk, the space in disk divided into blocks. Block is the smallest unit of transfer.

The size of a sector or block is usually 512 bytes.

The physical sectors of the disk are mapped to the logical blocks of the disk.

Sector 0 is the first *sector* of the first *track* on the outermost *cylinder*.

The mapping proceeds in order through that *track*, then through the rest of the *tracks* in that *cylinder*, and then through the rest of the *cylinders* from outermost to innermost.

So, the block address is indicated as

<c, t, s>

A hard disk has 63 sectors per track, 10 platters each with 2 recording surfaces and 1000 cylinders.
The address of a sector is given as a triple (c, h, s) where c is the cylinder number, h is the surface number and s is the sector number. Thus, the 0th sector is addressed as (0,0,0), the 1st sector as (0,0,1) and so on.

The address <400, 16, 29> corresponds to sector number-
1. 505035
2. 505036
3. 505037
4. 505038

400 x 20 x 63 + 16 x 63 + 29 = 504000 + 1008 + 29 = 505037

The address of 1039 sector is-
1. <0, 15, 31>
2. <0, 16, 30>
3. <0, 16, 31>
4. <0, 17, 31>

15 x 63 + 31 = 976
16 x 63 + 30 = 1038
16 x 63 + 31 = 1039
17 x 63 + 31 = 1102

Disk Scheduling

Operating system has to use the hardware efficiently.

To use the disk efficiently, the operating system has to improve the access time and bandwidth.

The 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.

The access time and bandwidth are improved by managing the order in which the disk I/O requests are serviced.

The disk I/O requests are placed in the disk queue.

To process or service these requests, several disk scheduling algorithms are used.

The different disk scheduling algorithms are:

1) First Come First Serve (FCFS) Scheduling

2) Shortest Seek Time First (SSTF) Scheduling

3) SCAN Scheduling

4) C- SCAN Scheduling

5) LOOK Scheduling

6) C-LOOK Scheduling

First Come First Serve (FCFS) Scheduling

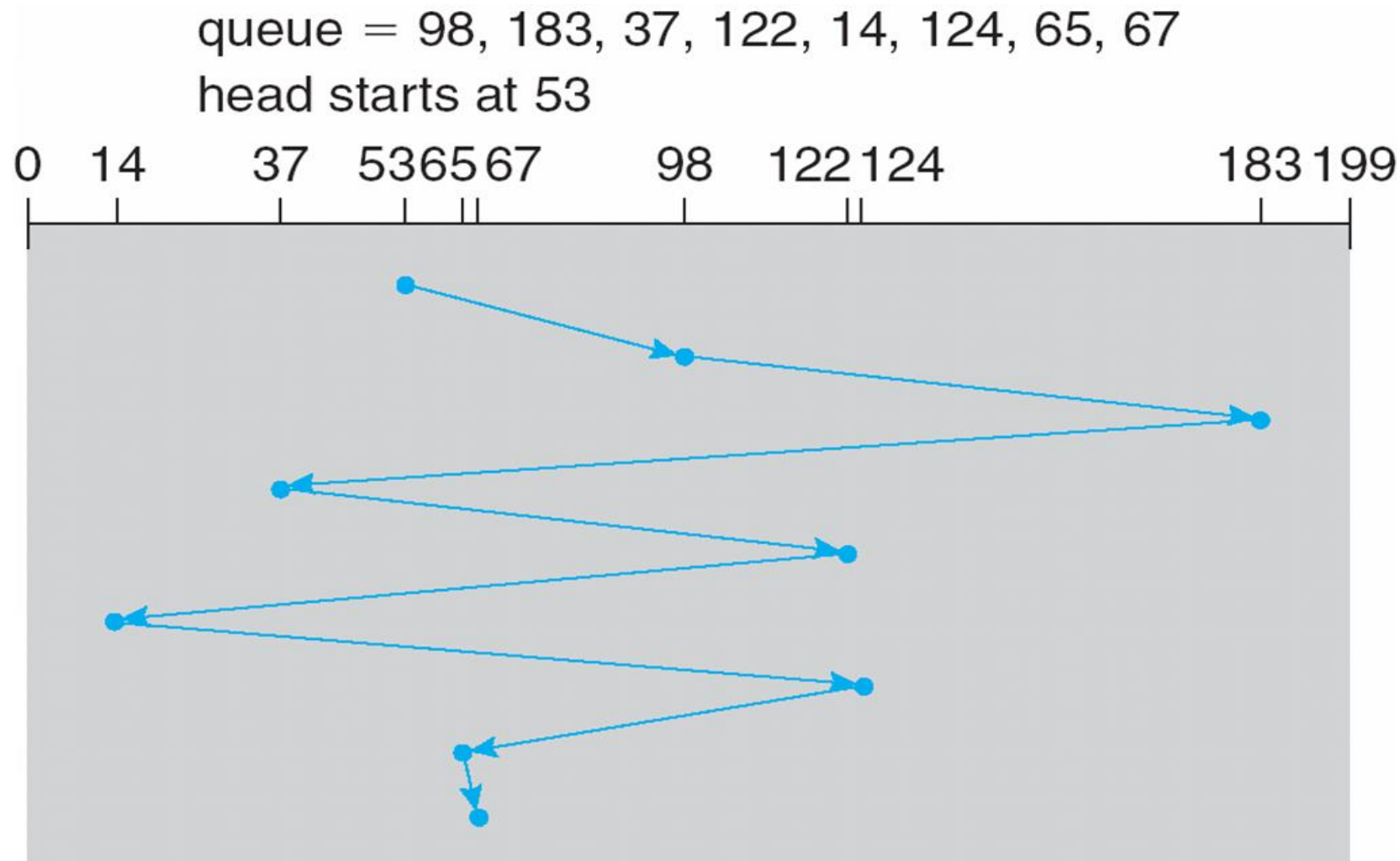
This algorithm is simple, but it does not provide the fastest service.

Consider a disk queue with requests for I/O to blocks on the cylinders 98, 183, 37, 122, 14, 124, 65, 67.

If the disk head is initially at cylinder 53, it will first move from 53 to 98, then to 183, 37, 122, 14, 124, 65, and finally to 67.

The total head movement is 640 cylinders.

This schedule is diagrammed in the following figure.


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

SSTF (Shortest Seek Time First) Scheduling

The SSTF algorithm selects the request with the least seek time from the current head position.

In other words, SSTF chooses the pending request closest to the current head position.

For the example request queue (98, 183, 37, 122, 14, 124, 65, 67), the closest request to the initial head position (53) is at cylinder 65.

After processing the request at cylinder 65, the next closest request is at cylinder 67.

From there, the request at cylinder 37 is closer, so 37 is served next.

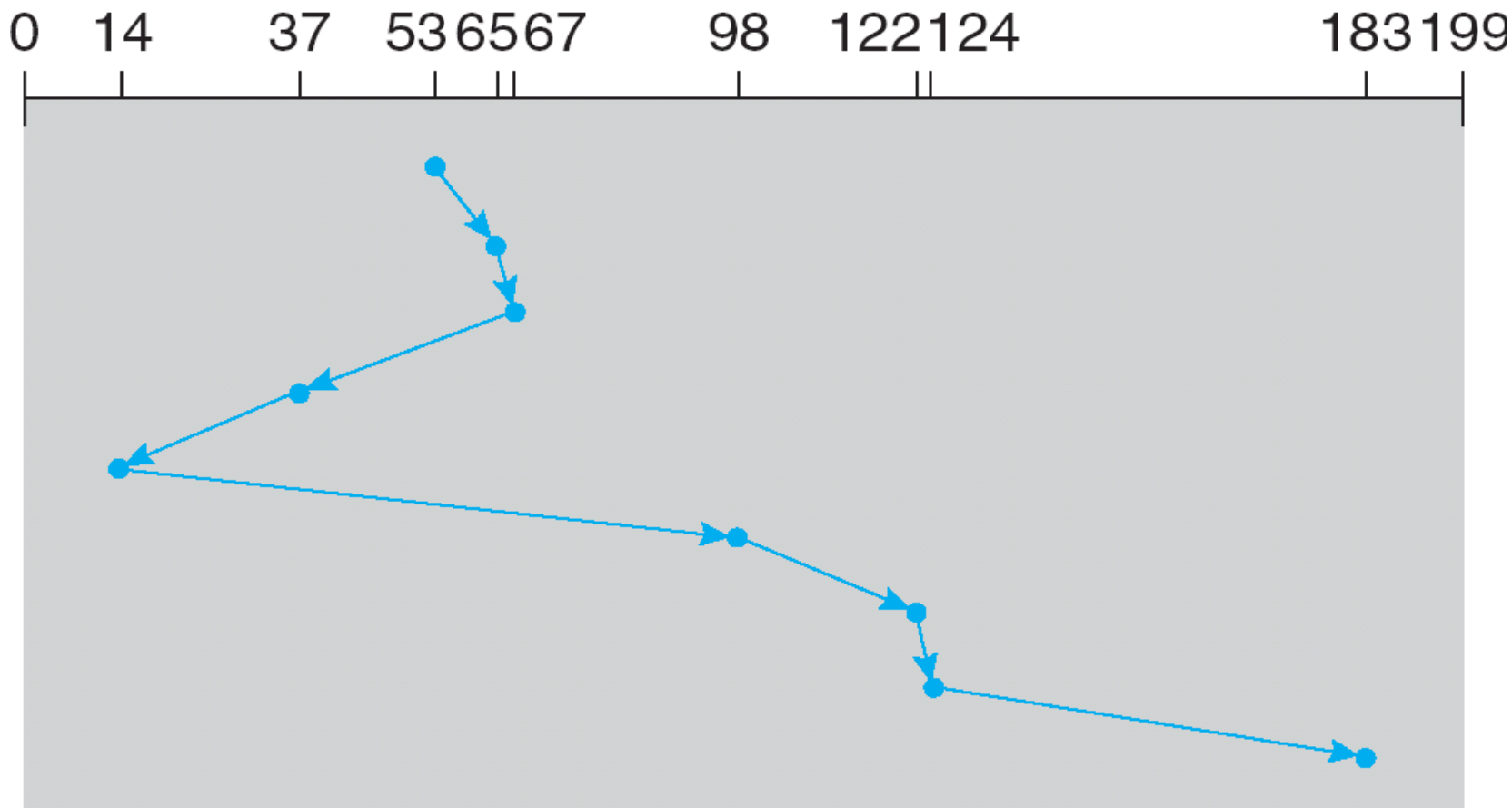
Continuing, we service the request at cylinder 14, then 98, 122, 124, and finally 183.

This scheduling method results in a total head movement of only 236 cylinders.

This schedule is diagrammed in the following figure.

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

head starts at 53



SSTF scheduling may cause starvation of some requests.

Requests may arrive at any time.

Suppose that we have two requests in the queue, for cylinders 14 and 186, and while the request from 14 is being serviced, a new request near 14 arrives.

This new request will be serviced next, making the request at 186 wait.

While this request is being serviced, another request close to 14 could arrive.

A continual stream of requests near one another can cause the request for cylinder 186 to wait indefinitely.

SCAN Scheduling

In the SCAN algorithm, the disk arm starts at one end of the disk and moves towards the other end, servicing requests along the way, until it gets to the other end of the disk.

At the other end, the direction of head movement is reversed, and servicing continues.

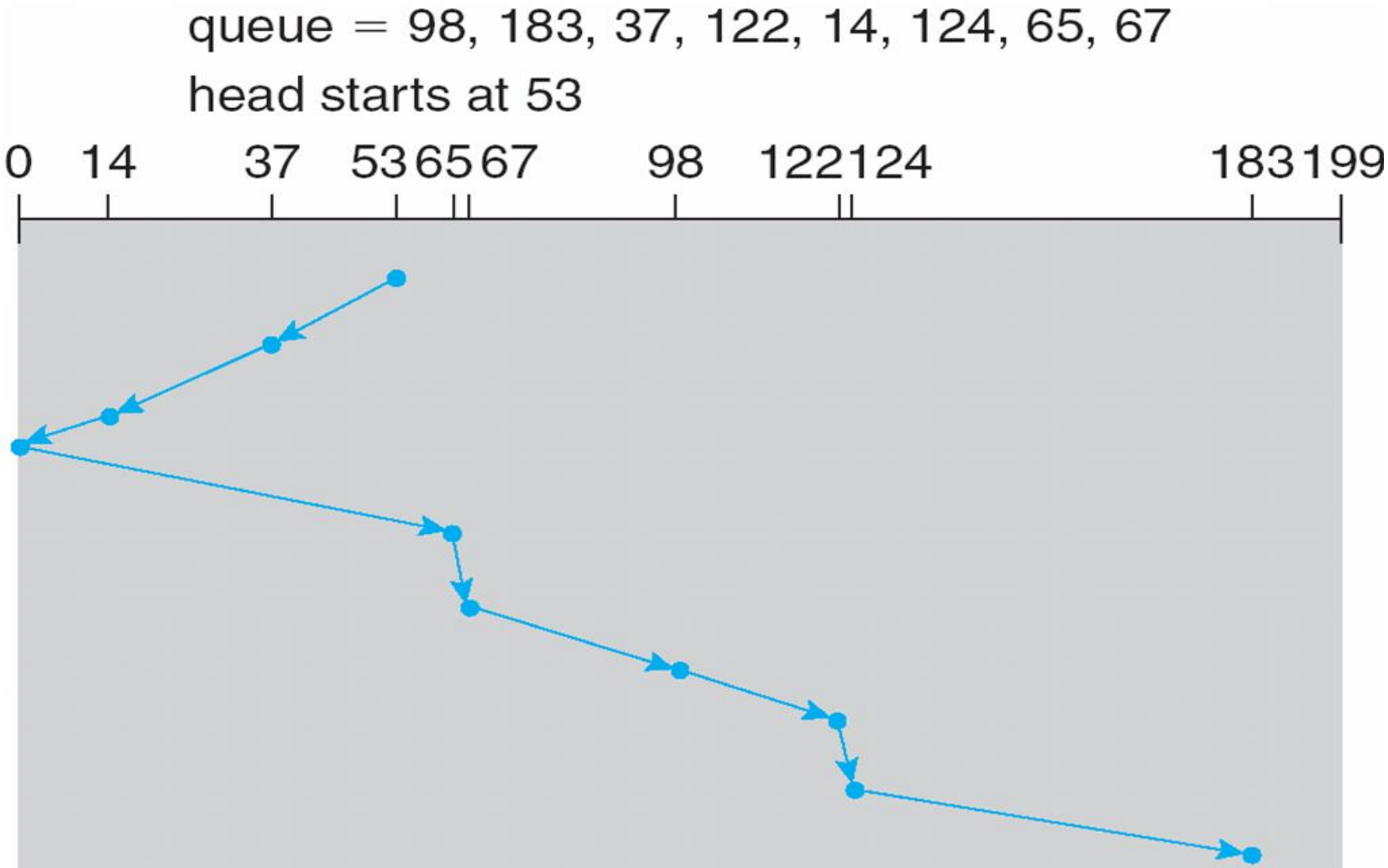
The head continuously scans back and forth across the disk.

Before applying SCAN algorithm to schedule the requests, we need to know the direction of head movement in addition to the head's current position.

Assuming that the disk arm is moving towards 0 and that the initial head position is again 53, the head will next service 37 and then 14.

At cylinder 0, the arm will reverse and will move towards the other end of the disk, servicing the requests at 65, 67, 98, 122, 124, and 183.

The total head movement is 236 cylinders. This schedule is diagrammed in the following figure.


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

When the head reaches one end and reverses direction, very few requests are in front of the head, since these cylinders have recently been serviced.

More requests will be at the other end of the disk and these requests have also waited the longest.

C-SCAN Scheduling

Circular SCAN (C-SCAN) is a variant of SCAN and provides a more uniform wait time.

Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way.

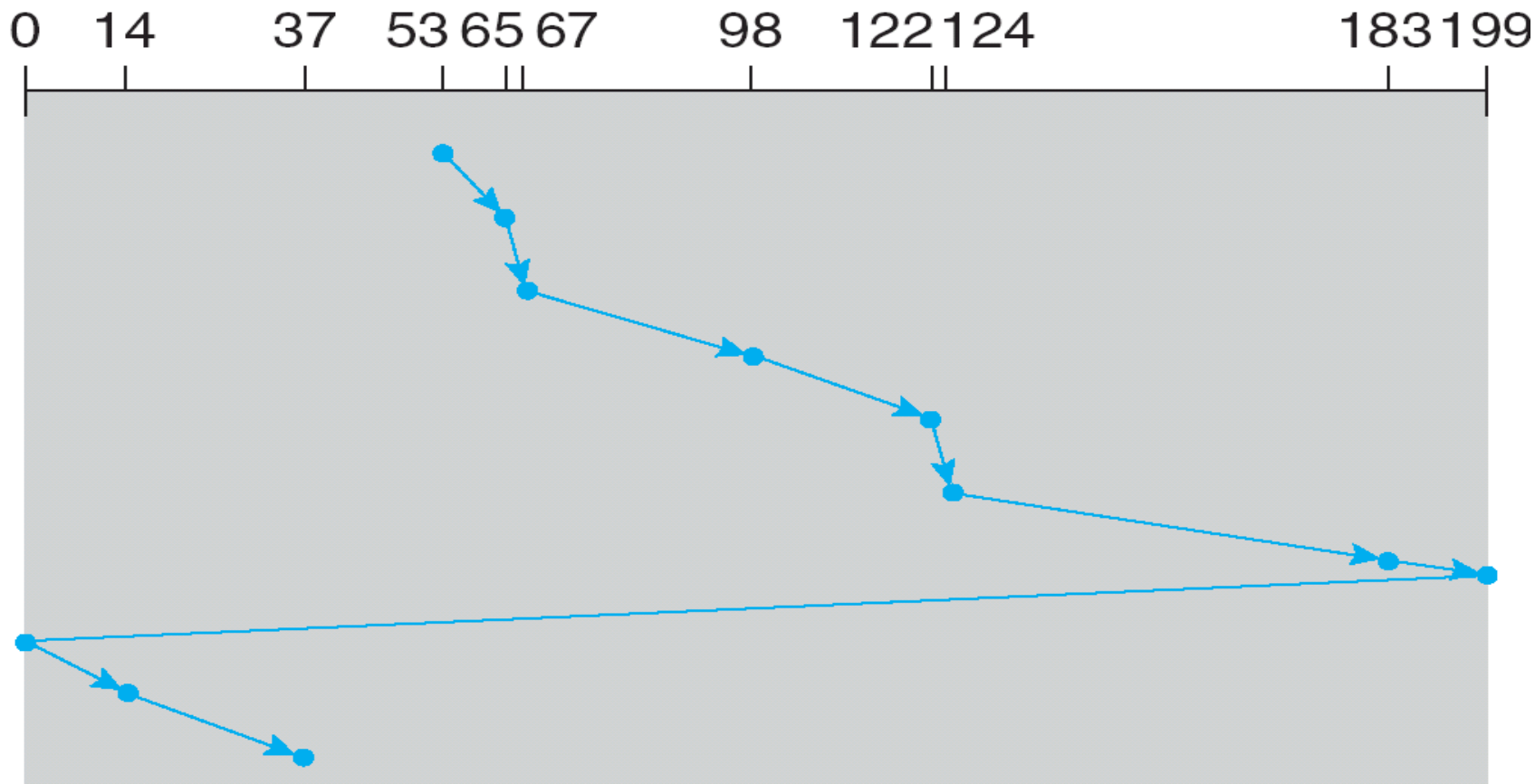
When the head reaches the other end, it immediately returns to the beginning of the disk without servicing any requests on the return trip.

This schedule is diagrammed in the following figure.

The total head movement is 383 cylinders.

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

head starts at 53



LOOK Scheduling

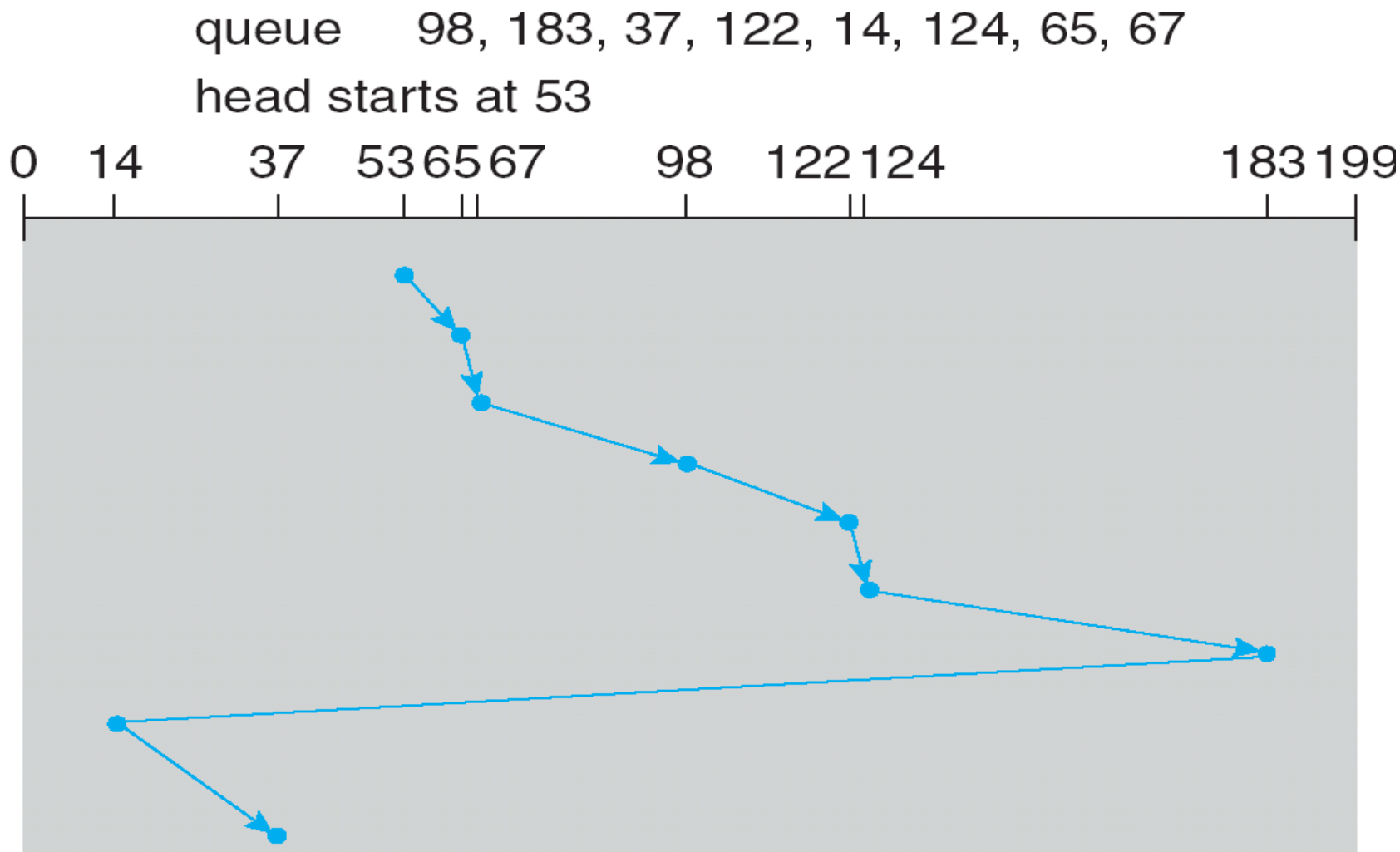
Both SCAN and C-SCAN move the disk arm across the full width of the disk.

In practice, the arm goes only as far as the final request in each direction.

Then, it reverses direction immediately, without going all the way to the end of the disk.

Versions of SCAN and C-SCAN that follow this pattern are called LOOK and C-LOOK scheduling, because they *look* for a request before continuing to move in a given direction.

The working of C-LOOK algorithm is scheduled in the following figure.


queue 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

Ex:

Consider a disk with 200 cylinders (0 to 199). Consider a disk queue with requests for I/O to blocks on the cylinders 82, 170, 43, 140, 24, 16, 190. The disk head is initially at cylinder 50. Head moves towards smallest cylinder number. Calculate total head movement under

1) FCFS
2) SSTF
3) SCAN
4) C-SCAN
5) LOOK
6) C-LOOK

Protection

Providing protection to files is avoiding improper access to files.

Protection mechanisms provide controlled access.

Access is permitted or denied depending on the type of access requested.

Types of Access

The different types of access are:

1) Read: Read from the file.

2) Write: Write or rewrite the file.

3) Execute: Load the file into memory and execute it.

4) Append: Write new information at the end of the file.

5) Delete: Delete the file and free its space for possible reuse.

6) List: List the name and attributes of the file.

Access Control mechanism for providing protection

An *Access Control List (ACL)* is associated with each file and directory.

The ACL specifies the names of users and the types of access allowed for each user.

When a user requests access to a particular file, the operating system checks the access list associated with that file.

If that user is listed for the requested access, the access is allowed.

This approach provides strong protection.

The drawbacks with this technique are:

1) Constructing ACL may be a tedious task, especially if we do not know in advance the list of users in the system.

2) The directory entry, previously of fixed size, now must be of variable size, resulting in more complicated space management.

The above problems can be resolved by using a condensed version of *Access Control List.*

To implement the condensed version of *ACL,* the users are categorized into three groups:

1) Owner: The user who created the file is the owner.

2) Group: A set of users who are sharing the file and need similar access is a group.

3) Universe: All other users in the system constitute the universe.

With this protection scheme, only three fields are needed to define protection.

Each field is a collection of bits, and each bit either allows or prevents the access associated with it.

For example, the UNIX system defines three fields of 3 bits each -rwx, where r controls read access, w controls write access, and x controls execution.

A separate field is kept for the file owner, for the file's group, and for all other users.