Operating Systems

Unit - 4

Mass Storage Systems

* Overview of mass-storage structures

(1) Magnetic Disks

→ provide bulk of secondary storage of modern computers

→ each disk platter has a flat circular shape, covered in magnetic material.

→ information is stored by recording it magnetically on the platters

(a) Transfer rate – rate at which data flows between drive and computer

(b) Positioning Time (random-access time) – time to move disk arm to desired cylinder (seek time) and time for desired sector to rotate under the disk head (rotational latency)

(c) Head crash – results from disk head making contact with disk surface

→ Disks are attached to computer via I/O bus

different buses:

EIDE - Enhanced Integrated Drive Electronics

ATA - Advanced Technology Attachment,

SATA, USB, SAS etc

→ Host controller in computer uses bus to talk to disk controller built into drive or storage array.

* Performance of hard disks

measured by access latency = average access time =
average seek time + average rotational latency

* Solid State Disks (SSDs)

→ nonvolatile memory that is used like a hard drive

→ more reliable than traditional hard disks because there are no moving parts - ~~Ex~~.

→ faster because they have no seek time or latency

→ consume less power

→ more expensive than traditional hard disks

→ Buses can be too slow → connect directly to PCI (peripheral component interconnect)

* Magnetic Tape

→ was used as an early secondary-storage medium

→ relatively permanent & can hold large quantities of data, but has slow access time

→ random access = 1000X slower than magnetic disk

→ mainly used for backup, storage of infrequently used data, transfer medium between systems.

→ kept in spool & wound or rewound past read-write head

* Disk Structure

→ Magnetic disk drives are addressed as large one-dimensional arrays of logical blocks, where the logical block is the smallest unit of transfer.

→ The 1D array of logical blocks is mapped into the sectors of the disk sequentially.

eg. 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 from the outermost to innermost cylinder.

→ Theoretically, should be easy to convert a logical block into a disk address but practically there are 2 problems:

(i) disks have some defective sectors

(ii) no. of sectors per track is not constant on some drives.

CLV = constant linear velocity = density of bits per track is uniform

CAV = constant angular velocity = disk rotation speed is constant

* Disk Attachment

→ Computers access disk storage in 2 ways. One way is through I/O ports (or host-attached storage), that is common on small systems

→ Another way is via a remote host in a distributed file system, this is called network-attached storage.

A. Host - Attached Storage

→ storage accessed through I/O ports

→ a typical desktop PC uses an I/O bus architecture called IDE/ATA.

→ High end workstations & servers use more sophisticated I/O architecture such as fibre channels.

→ Storage devices called for host - attached storage include hard disk drives, CD, DVD, tape drives.

B. Network - Attached Storage (NAS)

→ a special purpose storage system that is accessed remotely over a data network.

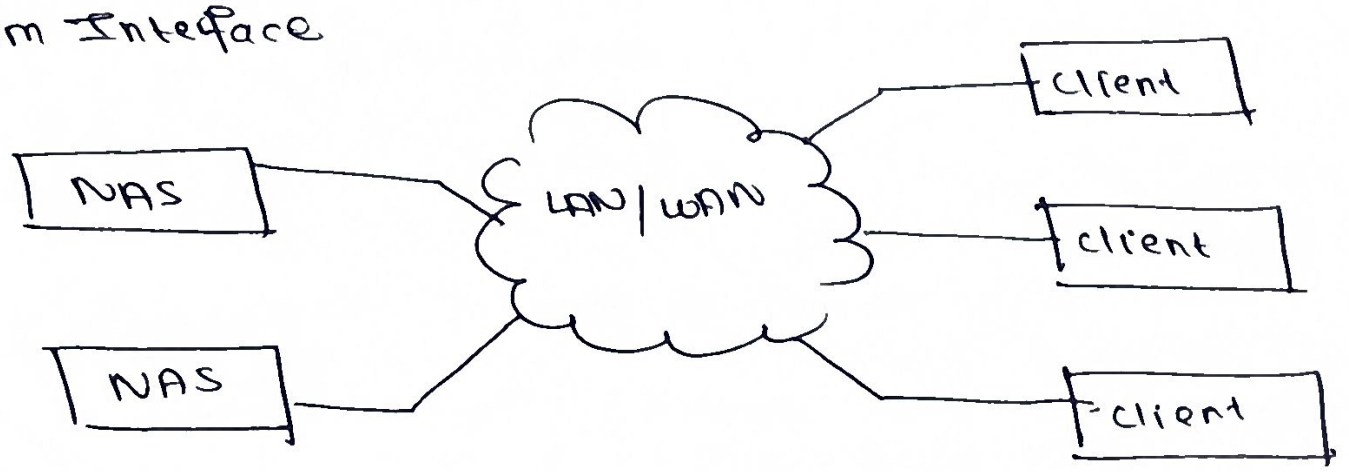
→ Clients access NAS via a remote-procedure - call interface such as NFS for UNIX & CIFS for Windows.

→ The remote procedure calls (RPCs) are carried via TCP or UDP over an IP network - usually in the same LAN.

→ iSCSI is the latest NAS protocol - uses IP to carry the SCSI protocol

↓

Small Computer System Interface



Network - attached storage

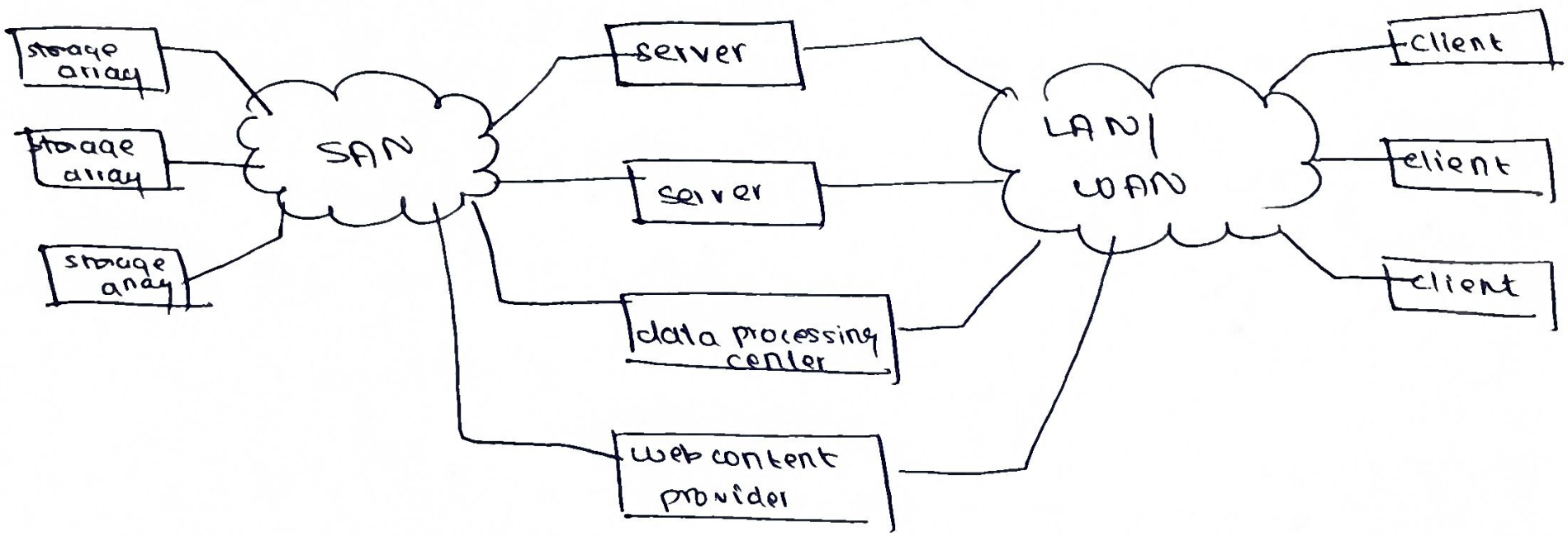
c. Storage Area Network

Disadvantage of network attached storage systems: consume bandwidth on the data network, increases latency of network communicating

SAN = a private network using storage protocols rather than networking protocols connecting servers & storage units

→ SANs = very flexible

→ multiple hosts & storage arrays can attach to the same SAN.



* Disk Scheduling

→ The OS is responsible for using hardware efficiently - for disk drives, it means having a fast access time & disk bandwidth.

→ minimize seek time, where seek time ≈ seek distance

→ minimize rotational latency - additional time for the disk to rotate the desired sector to the disk head.

→ disk bandwidth - total no. of bytes transferred / total time between first request for service & completion of last transfer

→ I/O requests include input or output mode, disk address, memory address, no. of sectors to transfer

→ Os maintains a queue of requests

① FCFS Scheduling

→ First come, first serve algorithm

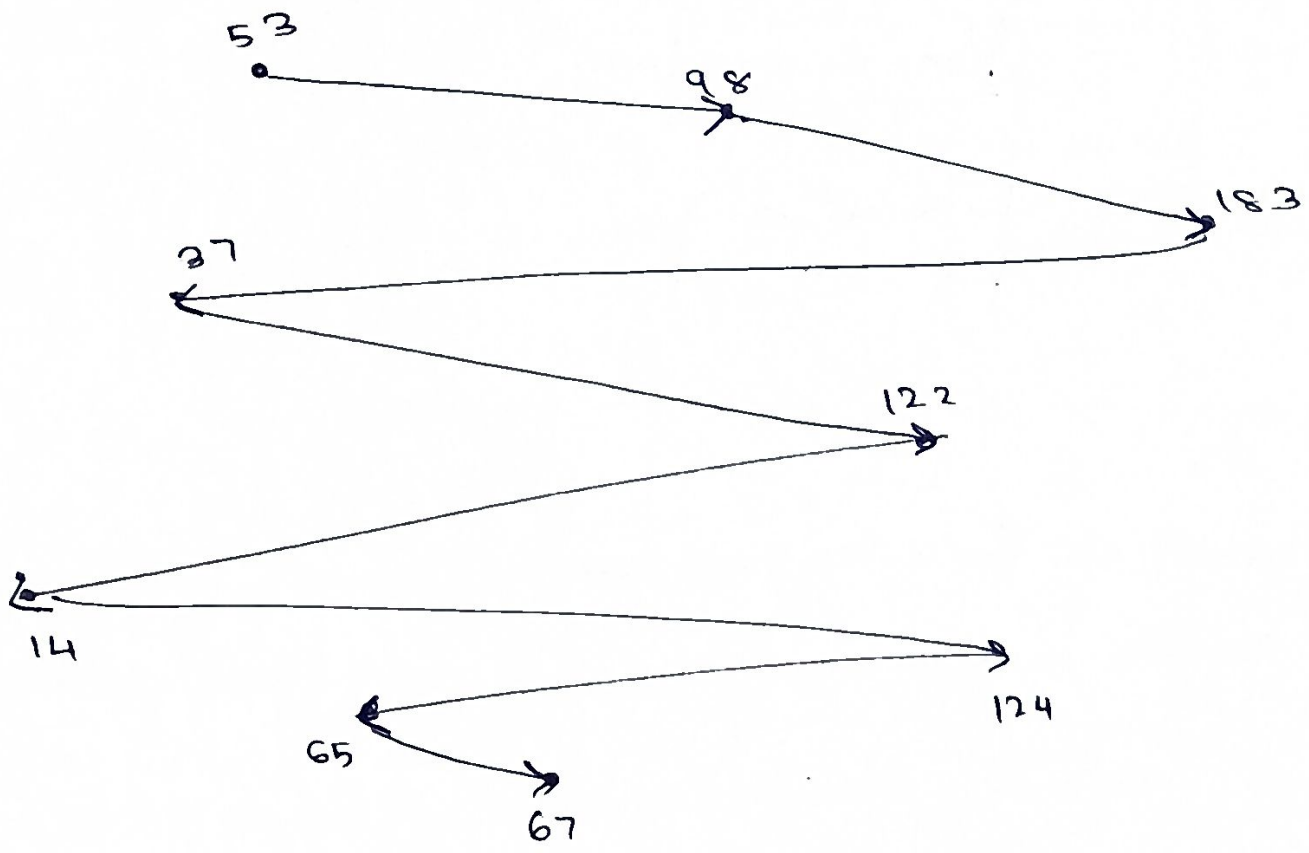
→ fair, but does not provide the fastest service

Example: Request Queue : 0-199

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

head pointer = 53.

0 14 37 53 65 67 98 122 124 183



Calculating Total no. of cylinders

(i) 53 → 98 = 45

(ii) 98 → 183 = 85

(iii) 183 → 37 = 146

(iv) 37 → 122 = 85

(v) 122 → 14 = 108

(vi) 14 → 124 = 110

(vii) 124 → 65 = 59

(viii) 65 → 67 = 2

Total = 640

(2) SSTF Scheduling

→ Shortest seek time first algorithm selects the request with the least seek time from the current head position

i.e chooses it chooses the pending request closest to the current head position

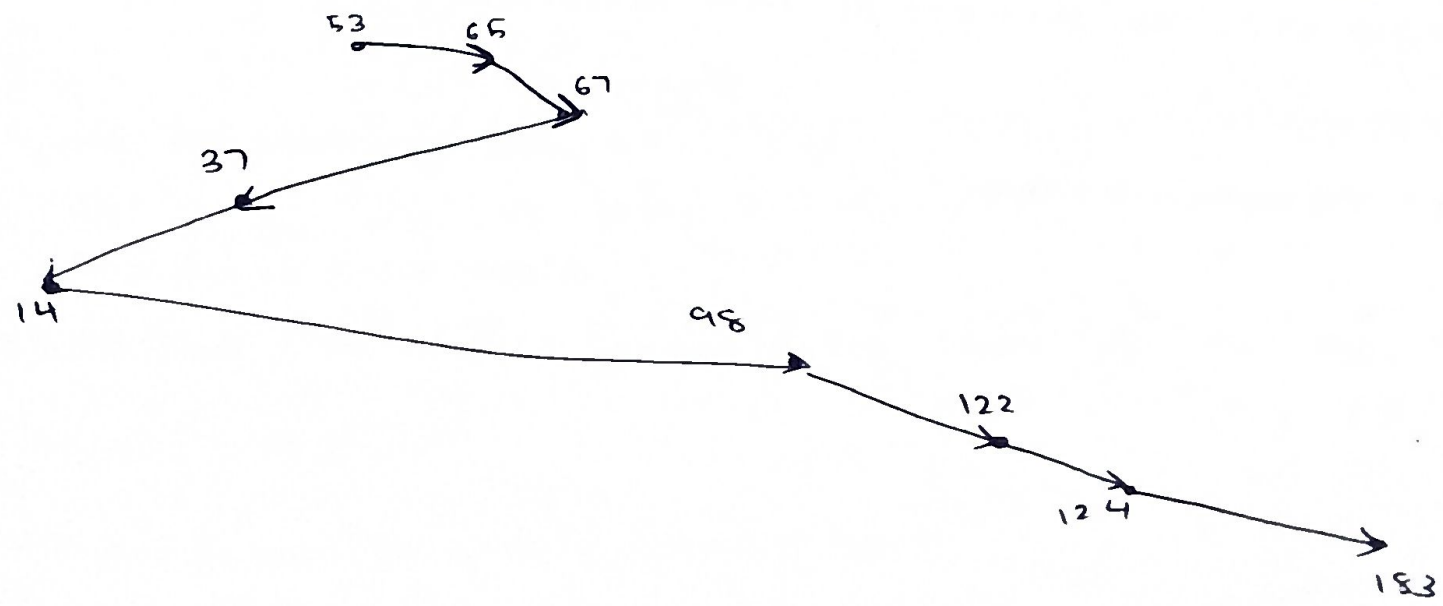
Example: Requests: 98, 183, 37, 122, 14, 124, 65, 67

head pointer = 53

in ascending order:

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

0 14 37 53 65 67 98 122 124 183 199.



Calculating Total no. of cylinders

(i) 53 → 65 — 12

(ii) 65 → 67 — 2

(iii) 67 → 37 — 30

(iv) 37 → 14 — 23

(v) 14 → 98 — 84

(vi) 98 → 122 — 24

(vii) 122 → 124 — 2

(viii) 124 → 183 — 59

Total = 236

Disadvantages of SSTF

→ may cause starvation of some requests

(3) SCAN Algorithm

→ disk arm starts at one end of the disk & moves toward the other end, service requests as it reaches each cylinder, until it reaches the other end of the disk.

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

→ called the elevator algorithm.

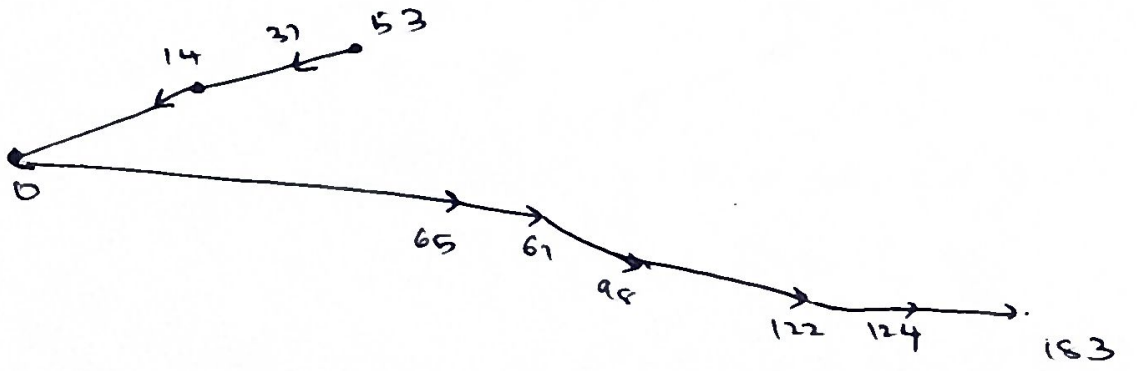
Example

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

given head moves to lowest value

0, 14, 37, 53, 65, 67, 98, 122, 124, 183, 199.

0, 14, 37, 65, 67, 98, 183, 122, 124



Calculating no. of ~~head~~ cylinders

(i) 53 → 37	16
(ii) 37 → 14	23
(iii) 14 → 0	14
(iv) 0 → 65	65
(v) 65 → 67	2
(vi) 67 → 98	31
(vii) 98 → 122	24
(viii) 122 → 124	2
(ix) 124 → 183	59
	236 cylinders

Disadvantage: If requests are uniformly dense, largest density at the other end of disk and those wait the longest

(4) C-SCAN

→ provides a more uniform wait time than SCAN

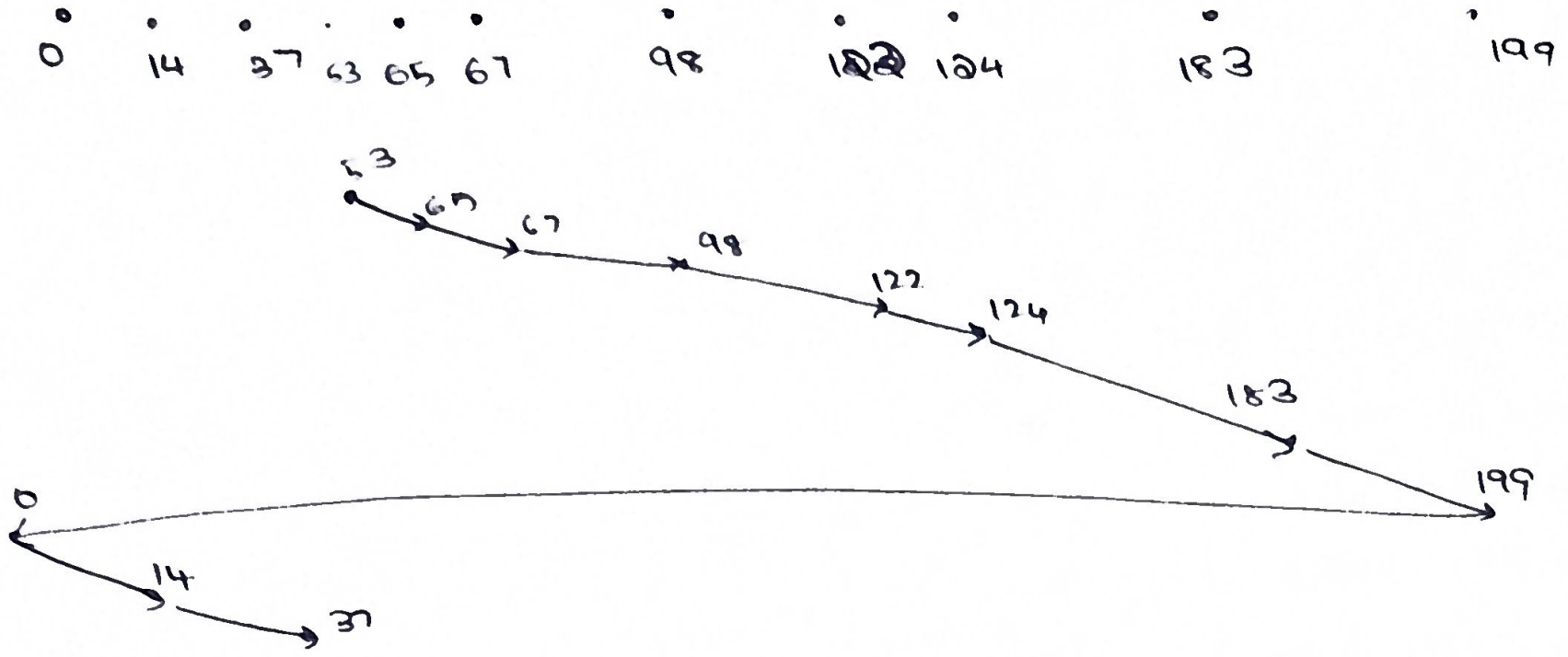
→ head moves one end of the disk to the other, servicing requests as it goes, when it reaches the other end, it immediately returns to the beginning of the disk, w/o servicing any requests on the return trip

→ Treats cylinders as a circular list that wraps around the last cylinder to the first one

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

head starts at 53

head moves towards highest value



Calculating Total no. of cylinders

(i) 53 → 65 — 12

(ii) 65 → 67 — 2

(iii) 67 → 98 — 31

(iv) 98 → 122 — 24

(v) 122 → 124 — 2

(vi) 124 → 183 — 59

(vii) 183 → 199 — 16

(viii) 199 → 0 — 199

(ix) 0 → 14 — 14

(x) 14 → 37 — 23

183

including backward movement

⇒ 183 + 199 = 382

(5) LOOK Algorithm

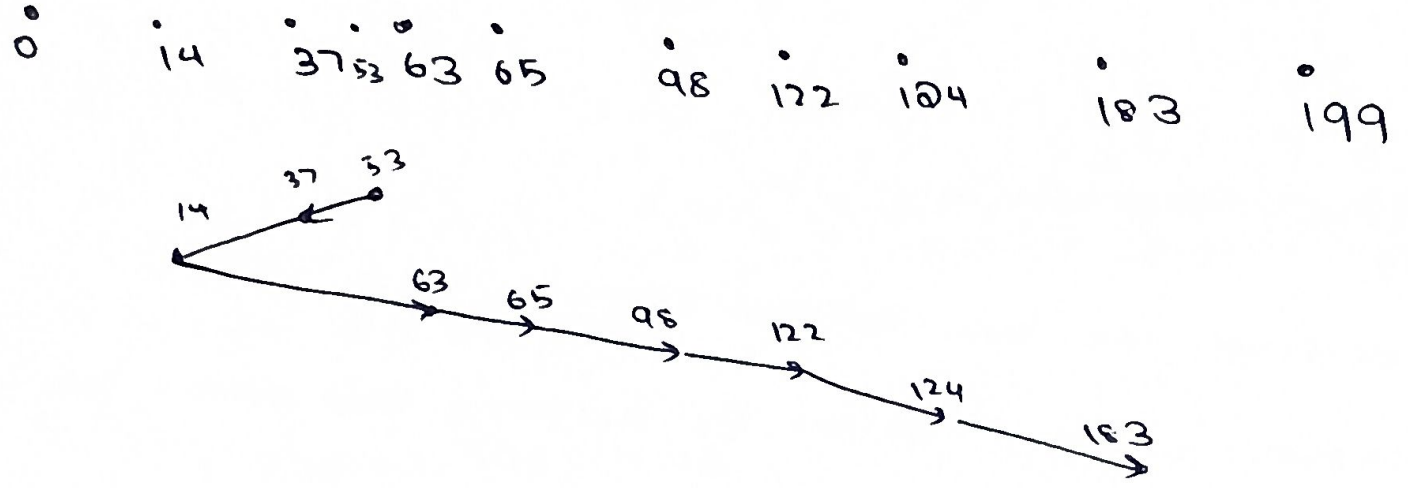
→ a version of scan, arm goes only as far as the last request in each direction, then reverses direction immediately w/o going all the way

to the end of the disk

Example: request: 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53

head moves toward lowest value



Calculation of total no. of cylinders = (53-14) + (183-14)

= 208

(6) C-LOOK Algorithm

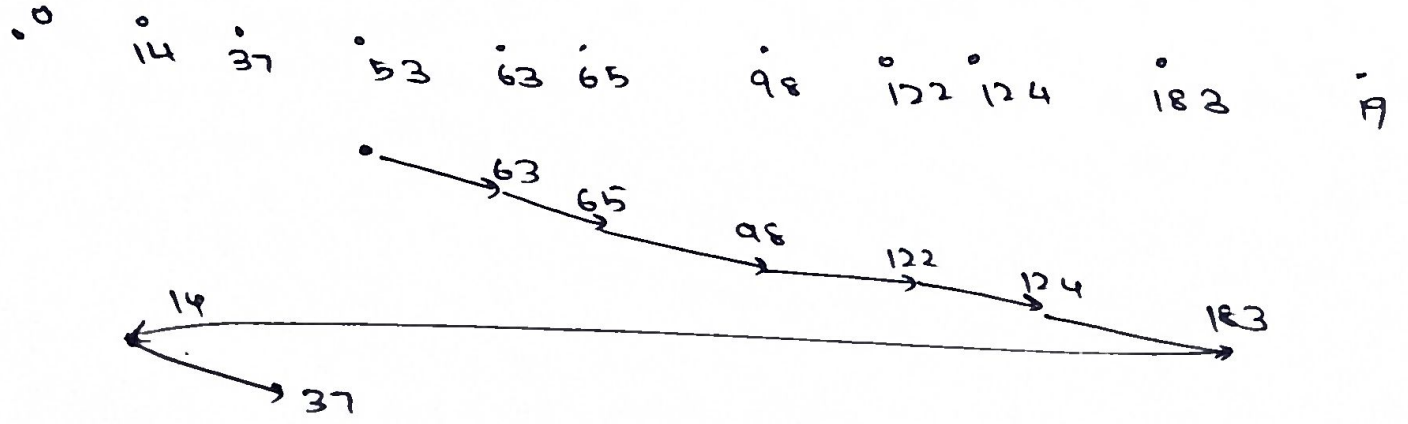
→ a version of C-SCAN

→ arm goes as far as the last request in each direction, then reverse direction immediately, w/o going all the way to the end of the disk. does not service requests in the backward direction

Example requests : 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53

head moves towards highest value



Calculation of Total no. of cylinders

$= (183 - 53) + (37 - 14)$

$= 153$

including backward movement: $153 + (183 - 14)$

$= 322$

* Selecting a Disk - Scheduling Algorithm

→ SSTF is common and has natural appeal

→ SCAN & C-SCAN perform better for systems that place a heavy load on the disk (less starvation)

→ Performance depends on number & type 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 OS, allowing it to be replaced w/ a diff. algo if necessary

→ Either SSTF or LOOK is a reasonable choice for the default algorithm.