Module 10 - Mass storage structure (disk)

Chapter 10 (Silberschatz)

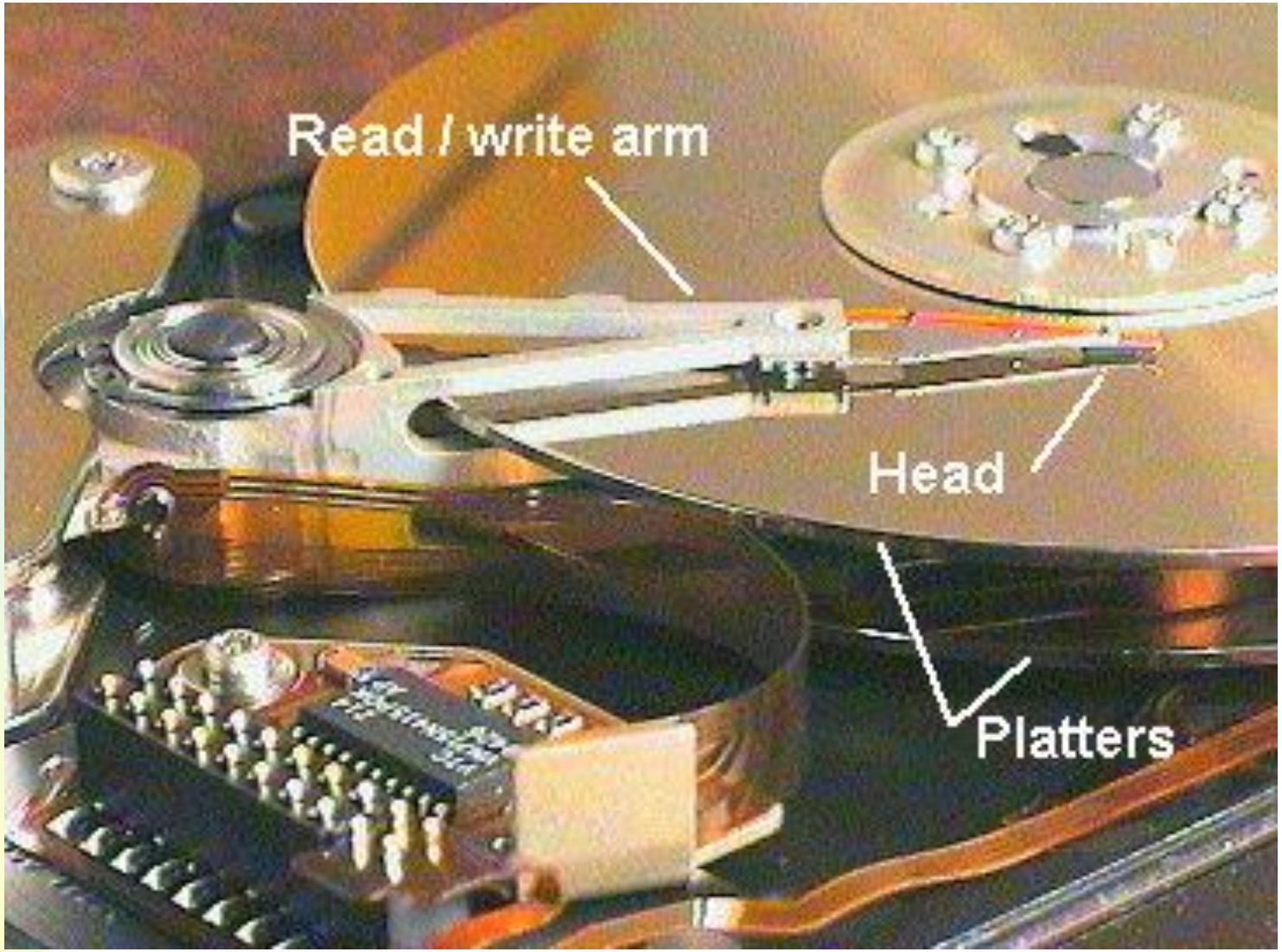
Important concepts of Module 10

- **Operation and structure of disk drives**
- **Calculating the execution time of a sequence of operations**
- **Different scheduling algorithms**
  - Operation, efficiency
- **Management of swap space**
  - Unix
- **RAID - error resistant drives**

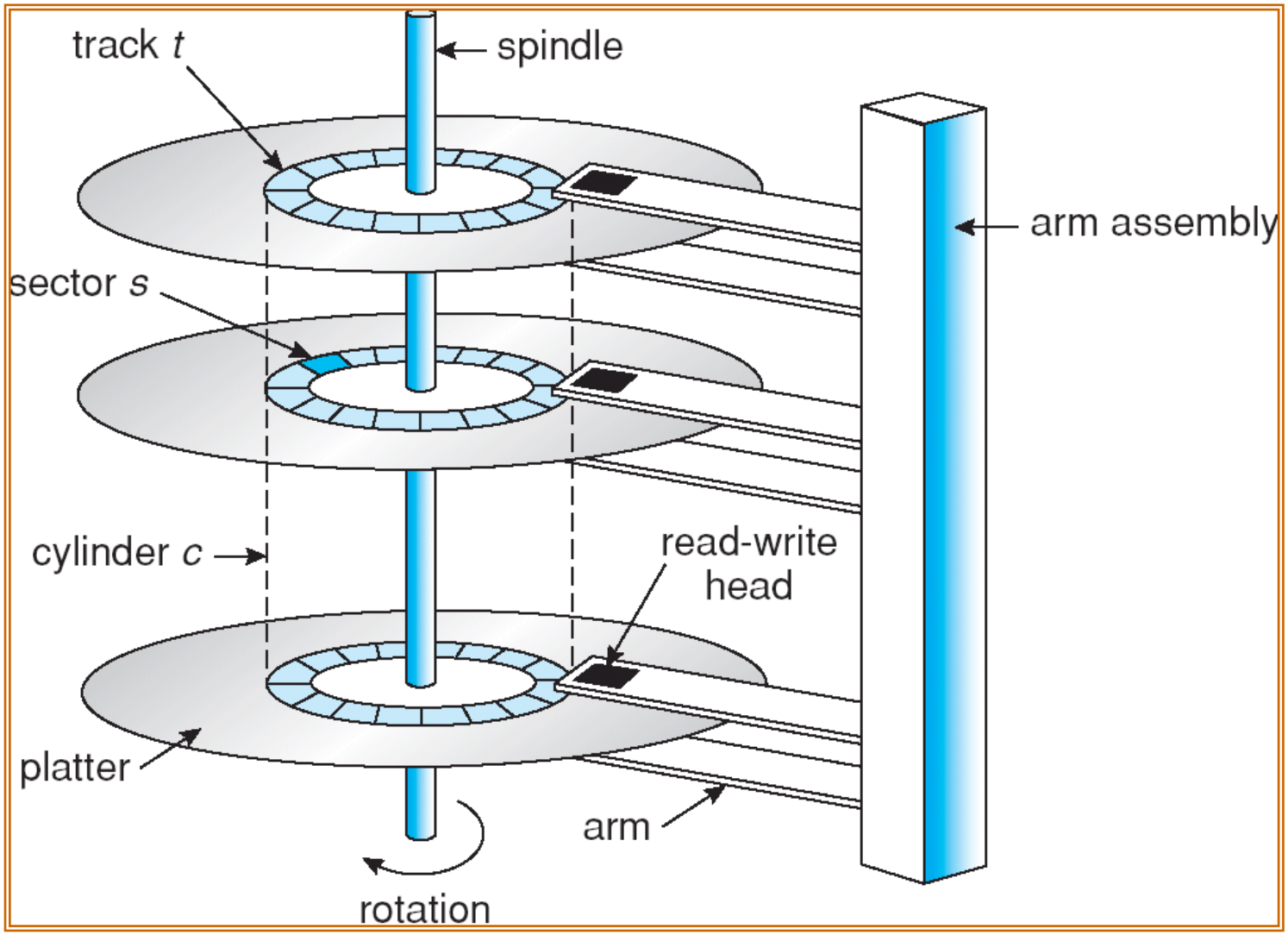
Magnetic discs

- **Rigid platters covered with magnetic recording material**
  - disk surface divided into tracks (tracks) which are divided into sectors
  - the disk controller determines the logical interaction between the drive and the computer

So, how does a disk look inside?



The insides of a hard drive!



Electronic Disks

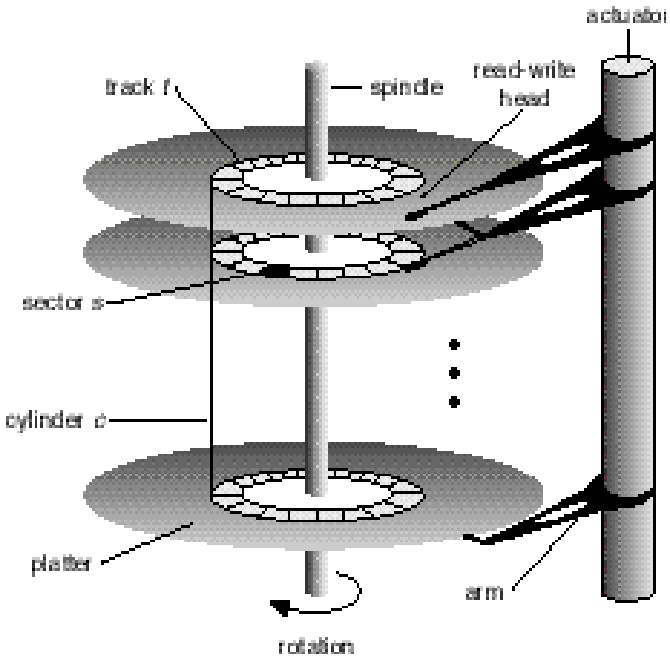
- **Today, more and more other types of memory are treated like hard disks, but are completely electronic**
  - For example, flash memory (memory sticks)
  - These devices do not contain seek time, rotational latency, etc.

Disk scheduling

- **Problem: Goal is optimal use of hardware**
- **Reduced total time of I/O requests**
  - given a queue of I/O disk requests, in what order should they be executed?

Parameters to consider

- **seek time(positioning time):**
  - the time taken for the disk drive to position itself on the desired cylinder
- **Rotation latency time**
  - the time taken by the disk drive who is on the correct cylinder to position yourself on the desired sector
- **Reading time**
  - time needed to read the track
- **Seek time is normally the most important, so it is the one that we will seek to minimize**

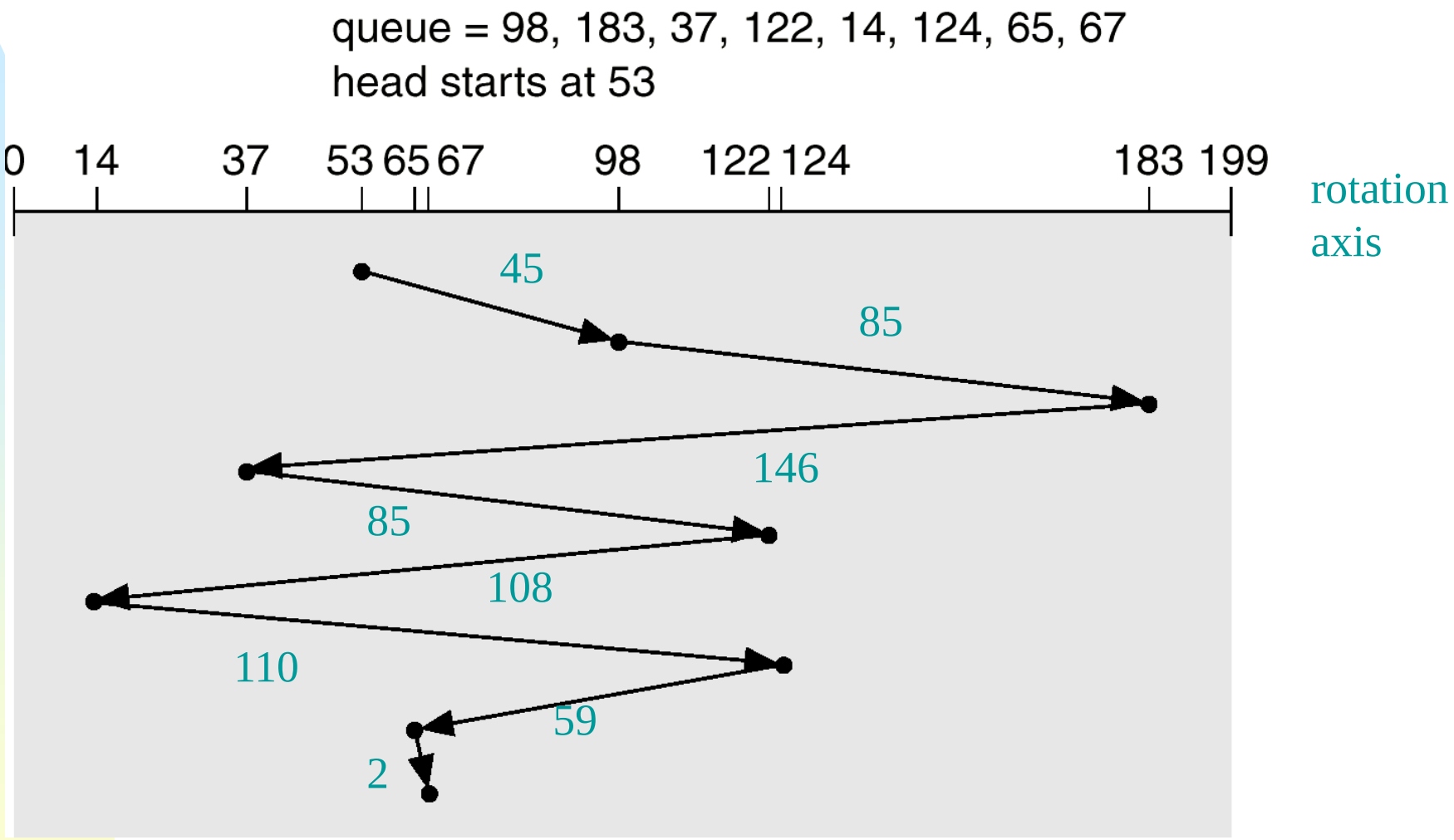


Disk I/O Queue

- In a multiprogrammed system with virtual memory, there will normally be a queue for the disk drive
- Shall study different methods using an I/O waiting queue containing requests:

  98, 183, 37, 122, 14, 124, 65, 67
- Each number represents a cylinder number.
- Must also known the starting cylinder: 53
- What order of requests will minimize the total seek time (movement of head to cylinders).
- Simple hypothesis: moving 1 cylinder takes 1 time unit.

First Come First Served: FCFS



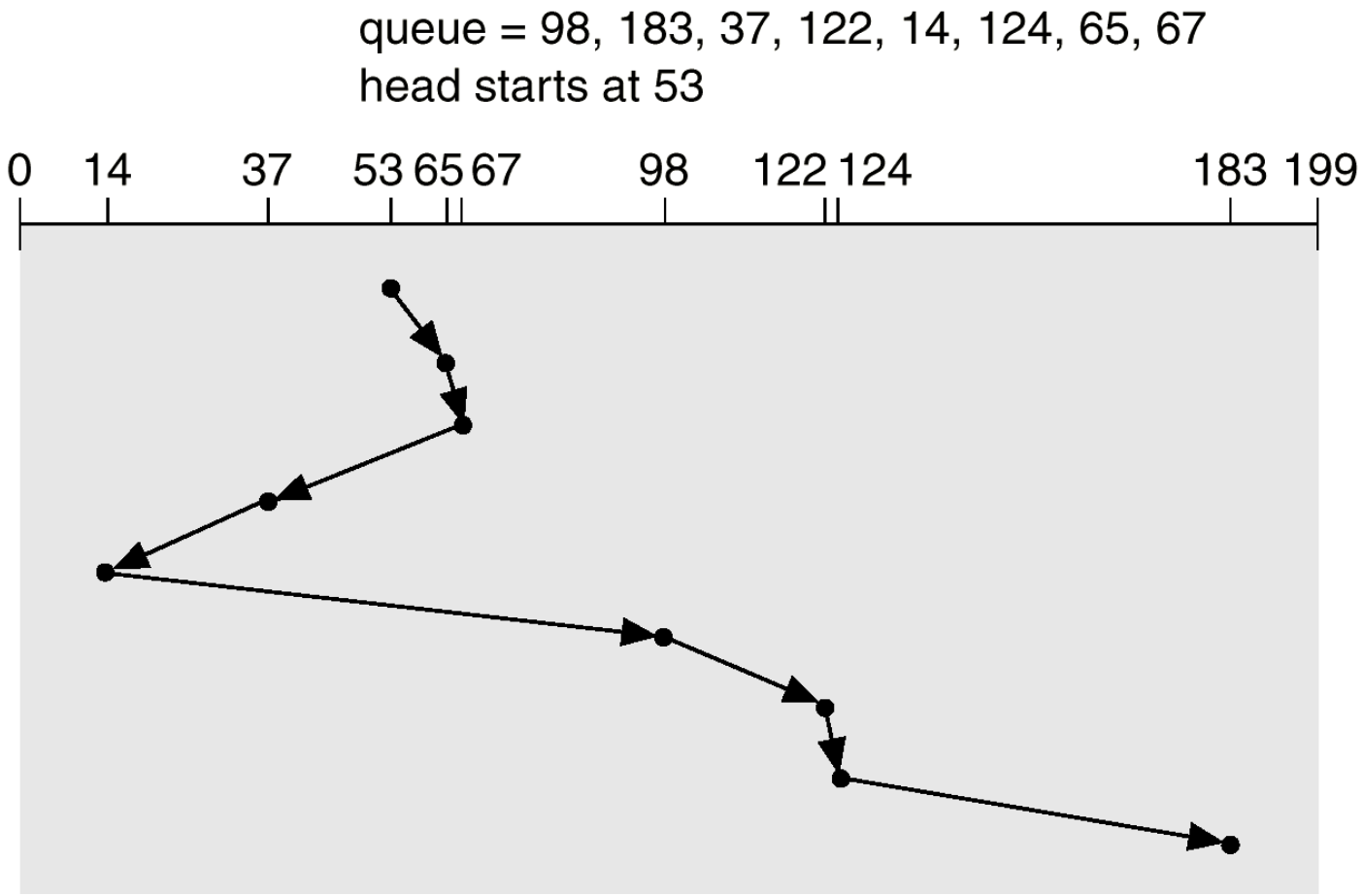
Total movement: 640 cylinders = (98-53) + (183-98) + ...

Average: 640/8 = 80

SSTF: Shortest Seek Time First

- Selects the request with the minimum seek time from the current head position
- Can be seen as a form of SJF scheduling
- Clearly better than the previous one
- But not necessarily optimal! (see manual)
- Can cause famine

SSTF: Shorter served



Total movement: 236 cylinders (640 for the previous one)

Average: 236/8 = 29.5 (80 for the previous one)

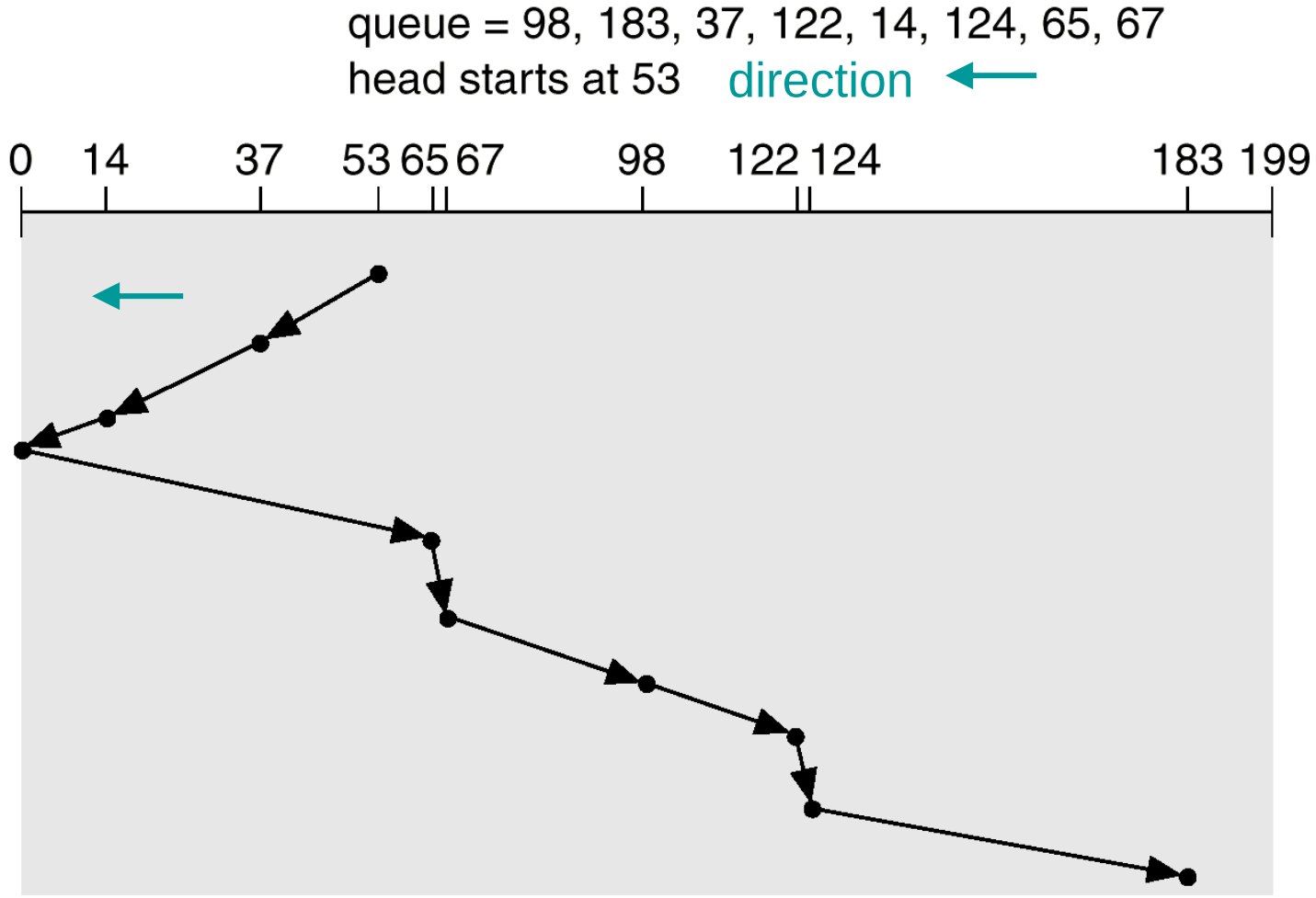
SCAN

Goal: **Avoid starvation, while still being efficient**

Idea:

- **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.**
- **Starvation is avoided, because we scan the whole disk**
- **Performance should also be reasonably good, as zig-zagging is avoided (i.e. queue: 1,200,2,199,3,198,4,197; requests will be serviced in one pass)**
- **Also called the *elevator algorithm.***
- **Problems**
  - Not much work to do when reversing directions, since requests will have accumulated at the other end of the disk.
  - Waste time moving to the end of the disk when have serviced the last request.

SCAN: the elevator



Total movement: 208 cylinders
Average: 208/8 = 26 (29.5 for SSTF)

C-SCAN

- Fast return to the start (cylinder 0) of the disc instead of reversing the direction
- Assumption: the return mechanism is much faster than the time to visit the cylinders

C-LOOK

- Same idea, but instead of going back to cylinder 0, going back to the first cylinder that has a query

C(ircular)-SCAN

First problem with SCAN:

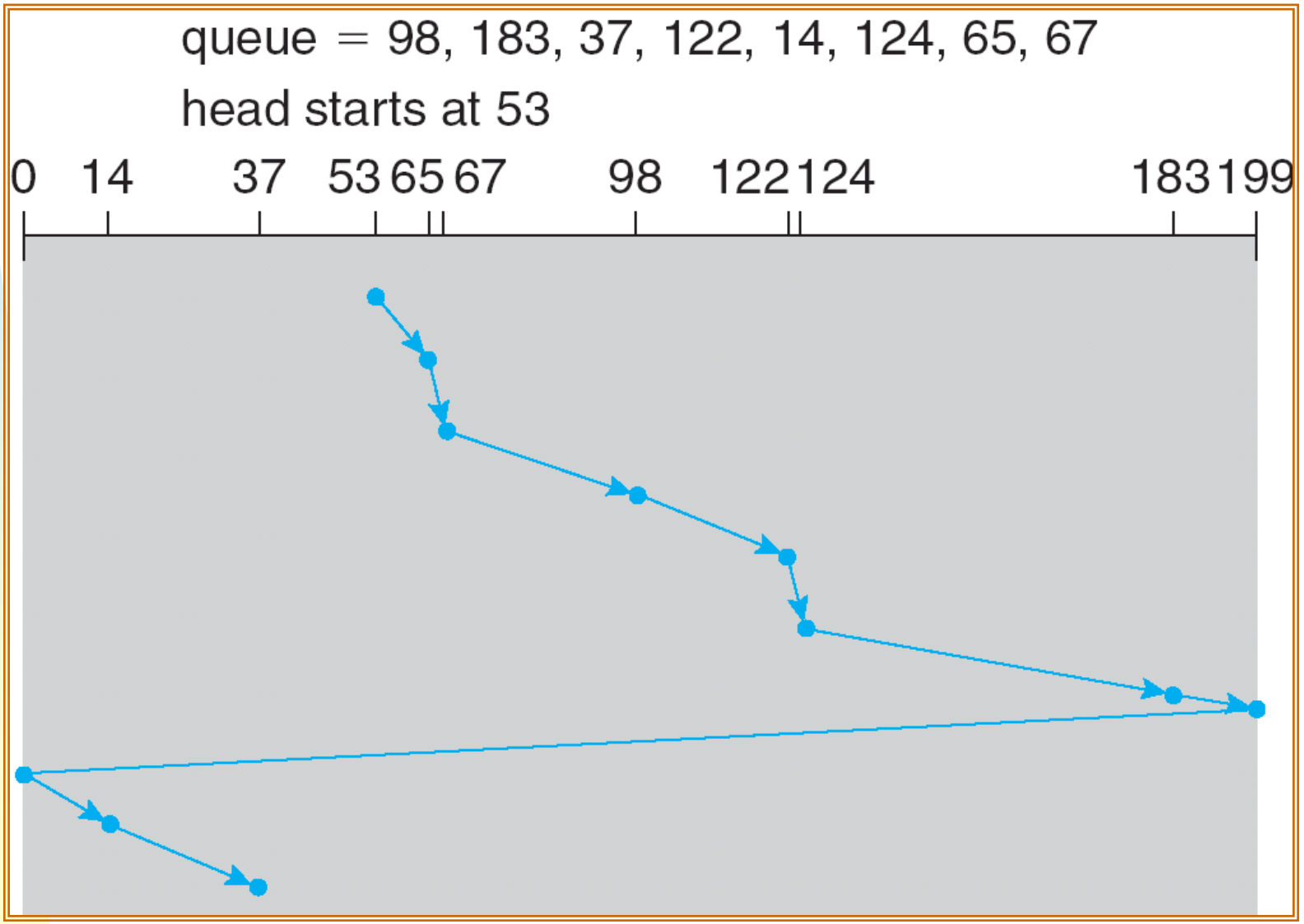
- **Not much work to do when reversing directions, since requests will have accumulated at the other end of the disk.**

Idea:

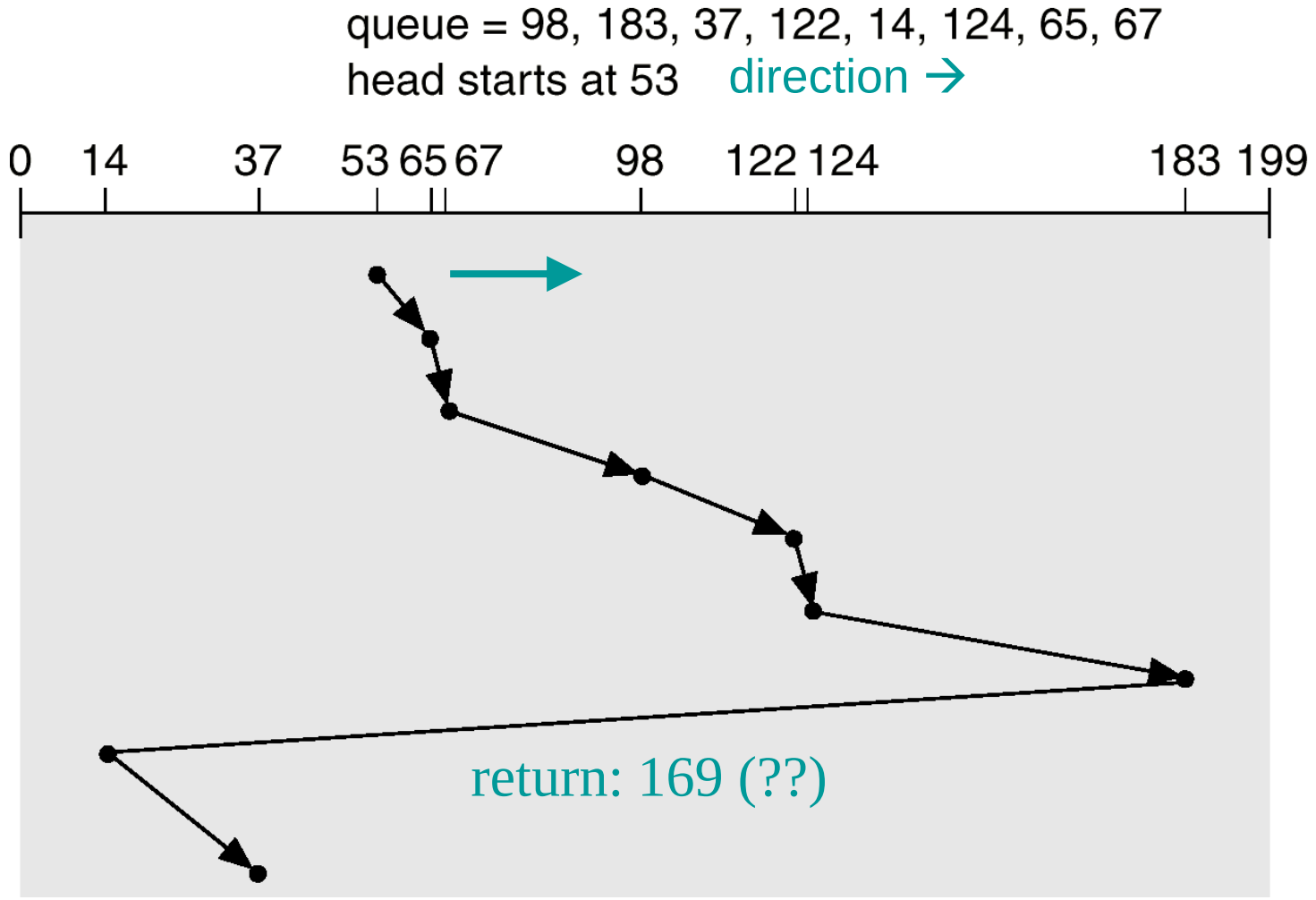
- **The head moves from one end of the disk to the other, servicing requests as it goes.**
- **When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.**
- **The return trip is relatively fast, as it does not need to accelerate/decelerate for each request**

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

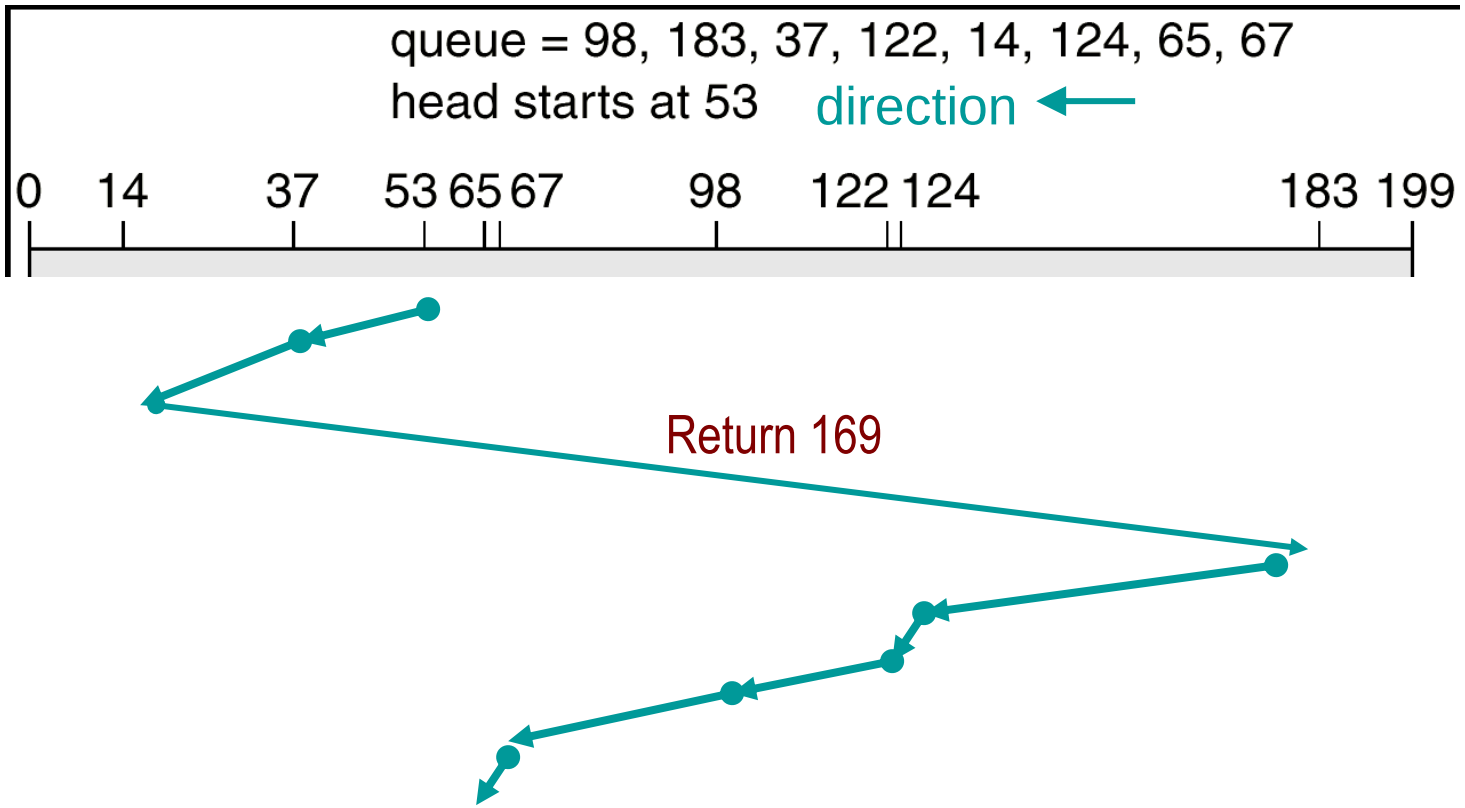


153 without considering the return (19.1 on average) (26 for SCAN)

BUT 322 with return (40.25 on average)

Normally the return will be quick so the actual cost will be between the two

C-LOOK with opposite initial direction



Very similar results:
157 without considering the return, 326 with the return

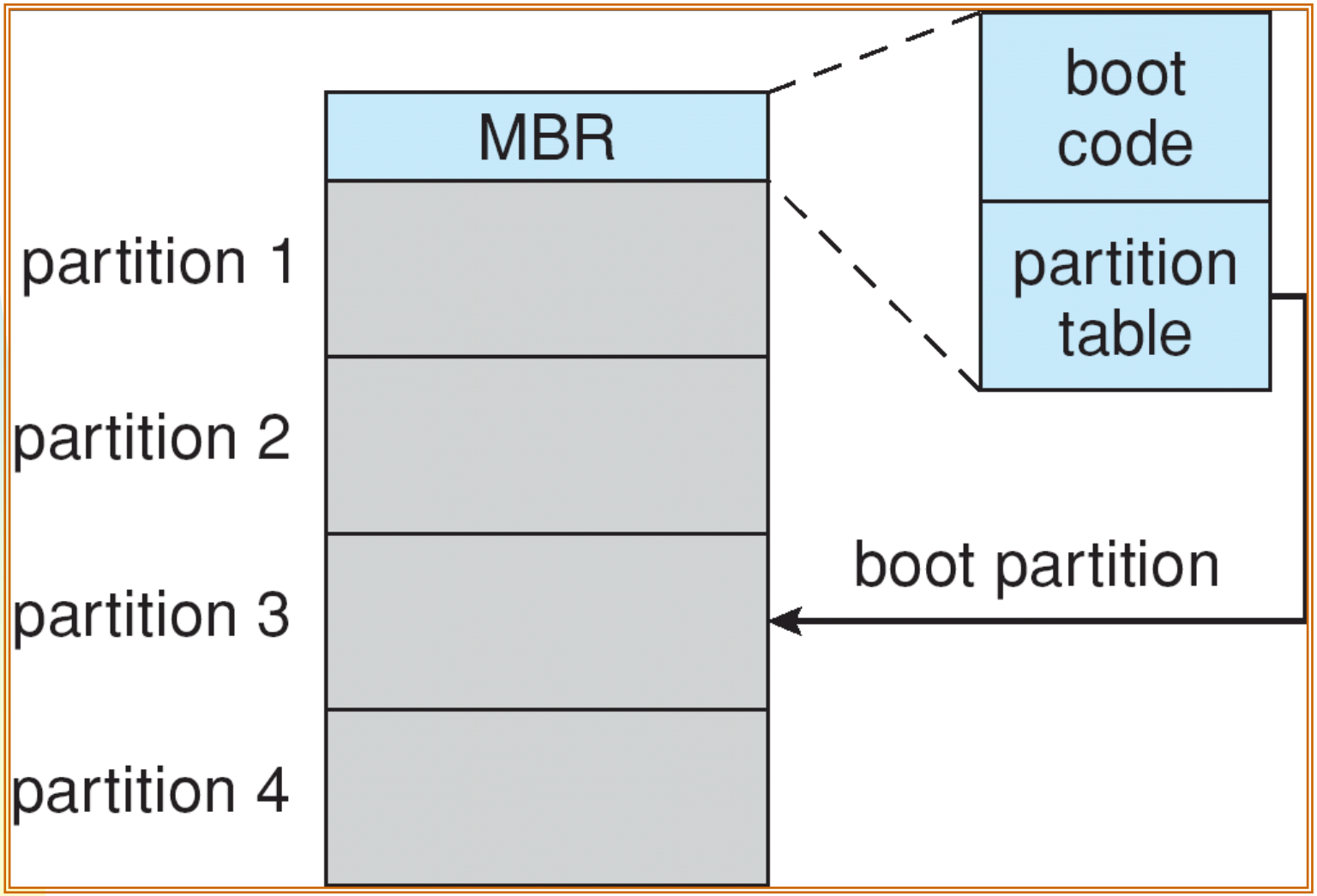
Comparison

- **If the queue often contains very few items, the FCFS algorithm should be preferred (simplicity)**
- **Otherwise, SSTF or SCAN or C-SCAN?**
- **In practice, it is necessary to take into consideration:**
  - Actual travel time and time to go back to start
  - File and directory organization
    - The directories are on disk too ...
  - The average length of the queue
  - The incoming flow of requests

Disk Management

- ***Low-level formatting*, or *physical formatting***
  - divide the disk into sectors the controller can read
  - initialize each sector (header, trailer)
  - Comes from factory
- ***Partitioning***
  - logically *partition* the disk into one or more groups of cylinders, each of them having its own FS.
  - Set the boot partition
- ***Logical formatting* or "making a file system"**
  - Write the FS data on disc
  - i.e. write FAT, the root directory, inodes, …

Booting from a Disk in Windows 2000



Bad Block Management

There are tens and hundreds of millions of blocks on a HD

Bad blocks do happen

What to do about them?

- Have some spare blocks and when you detect a bad block, use the spare block instead
  - i.e. the HD has nominally 100 blocks, but was fabricated with 110, 0..99 are used, 100-109 are spare
  - If block 50 fails, the HD controller (after being told by the OS) will use block 100 to store the logical block 50
- But that messes up the efficiency of the disc scheduling algorithms!
- Don’t panic! Just have spare blocks in each cylinder (sector sparing) or use sector slipping

Swap-Space Management

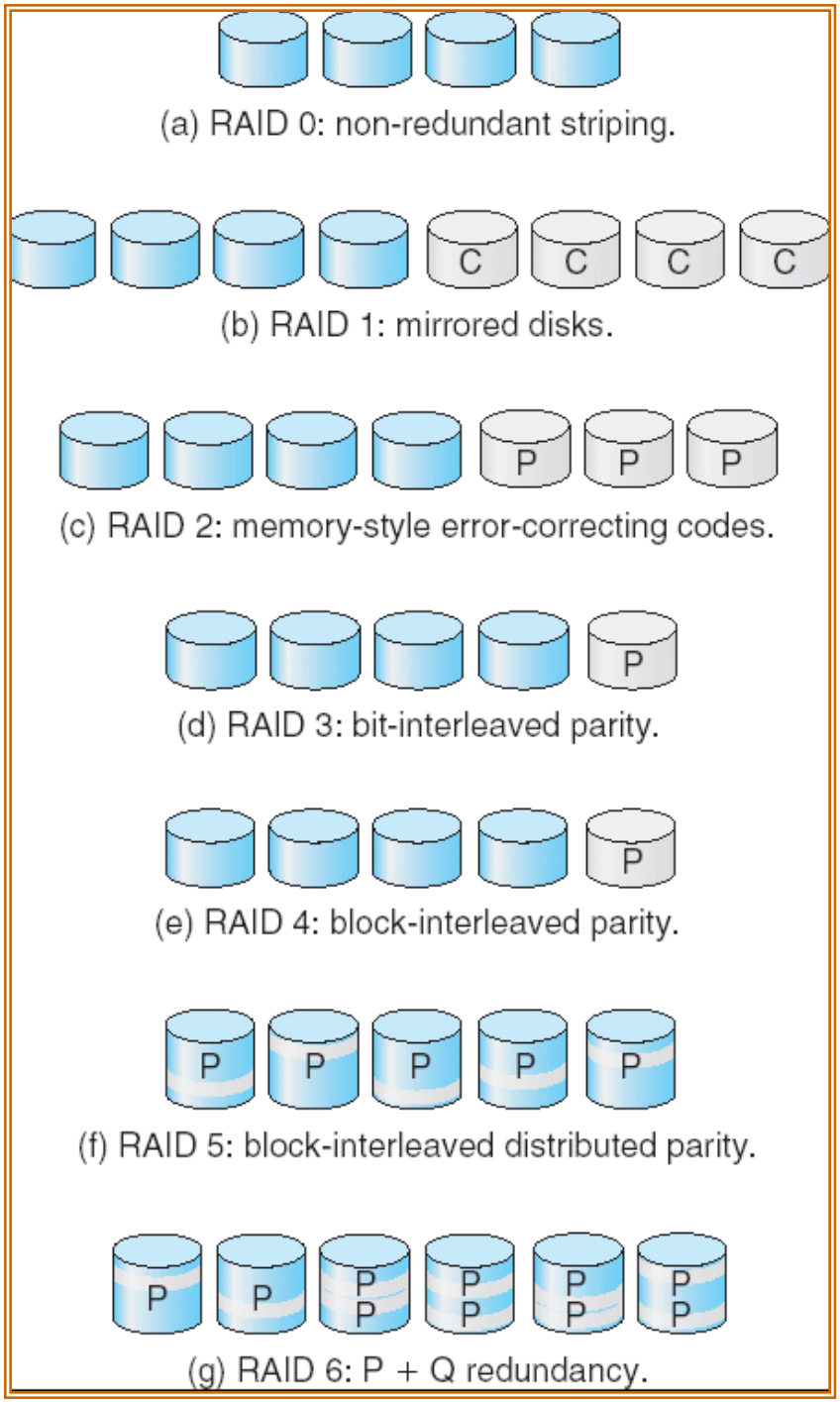
- **Swap-space — Virtual memory uses disk space as an extension of main memory**
- **Where is the swap-space on the disc?**
  - As a part of the normal file system
    - good: Simple, flexible
    - bad: SsssLOoooWwww
  - in a separate disk partition
    - Using its own, optimized algorithms and structures
    - i.e. fragmentation not such a big problem – on boot starts anew, but really want fast sequential access
    - Good: faster
    - Bad: might waste space

Swap-Space Management

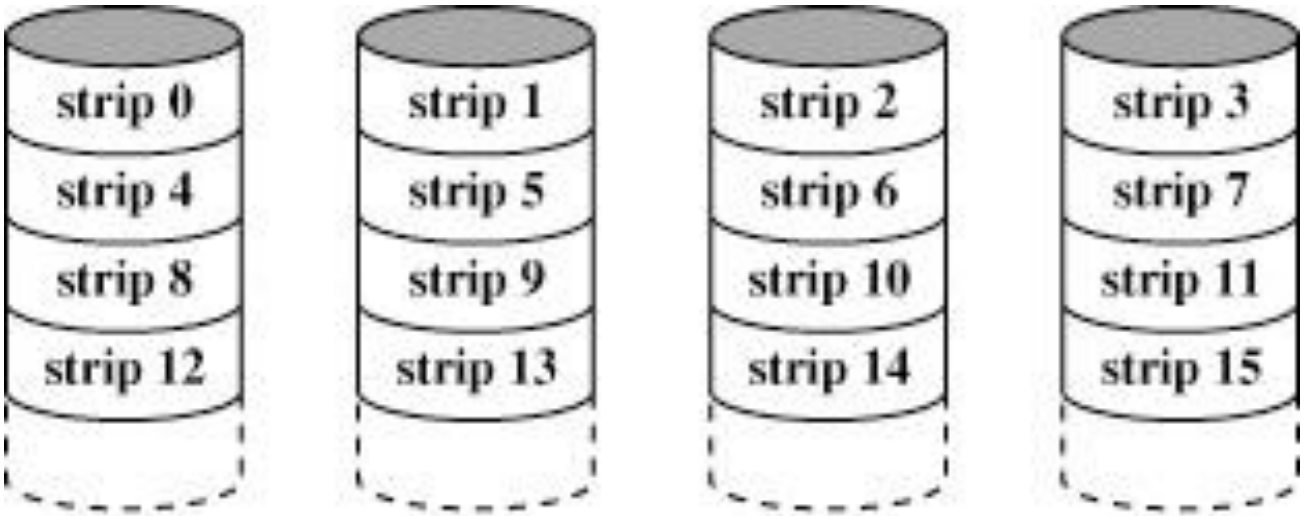
- **When to allocate space in the swap space?**
  - When the process starts/the virtual page is created
  - When the page is replaced
- **What to remember?**
  - Kernel uses *swap maps* to track swap-space use.

RAID Levels

- **Striping: reading multiple disks in parallel**
  - Bit interleaving
  - Block interleaving
- **Redundancy: redundant data to cover data,**
  - duplicate data (mirror disks)
  - Parity or correction codes
- **Examples: The six RAID levels store 4 data disks**



RAID: Redundant Array of Independent Disks

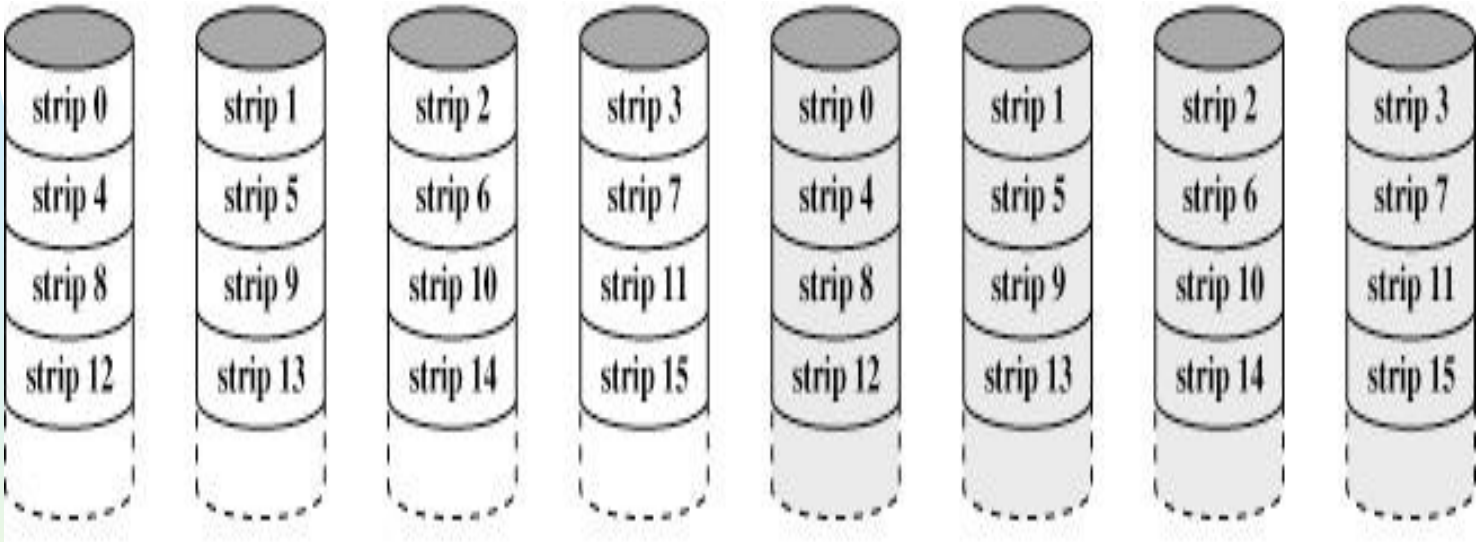


(a) RAID 0 (non-redundant)

Stallings

By distributing the data on different disks, it is likely that a big read can be done in parallel (instead of reading strip0 and strip1 in sequence, this organization allows them to be read at the same time)

Redundancy in RAID

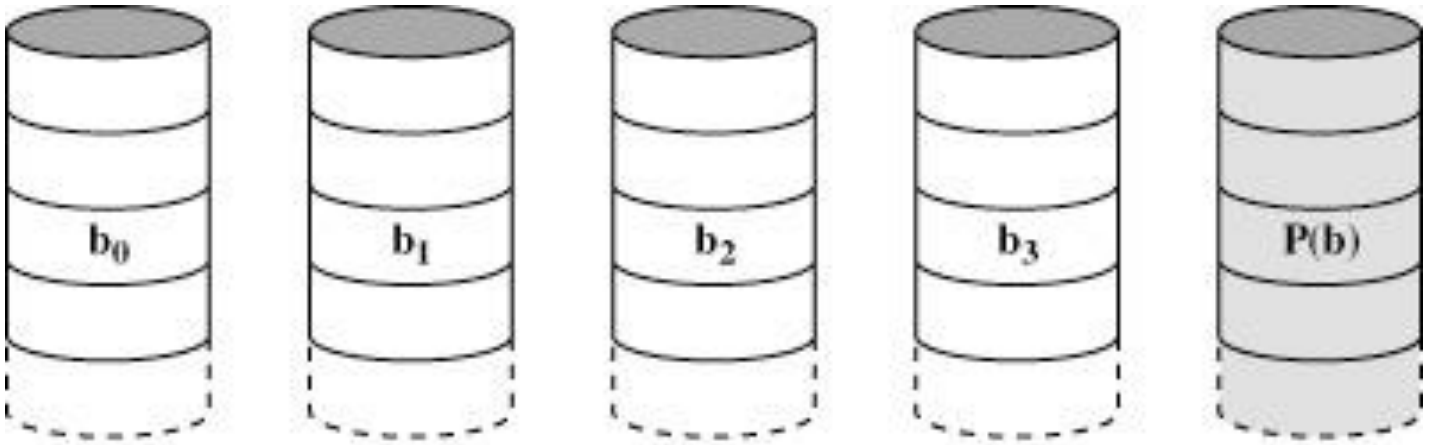


(b) RAID 1 (mirrored)

Stallings

Duplicate data to increase parallelism and remedy data loss (expensive but used in practice)

RAID: Error correction by parity



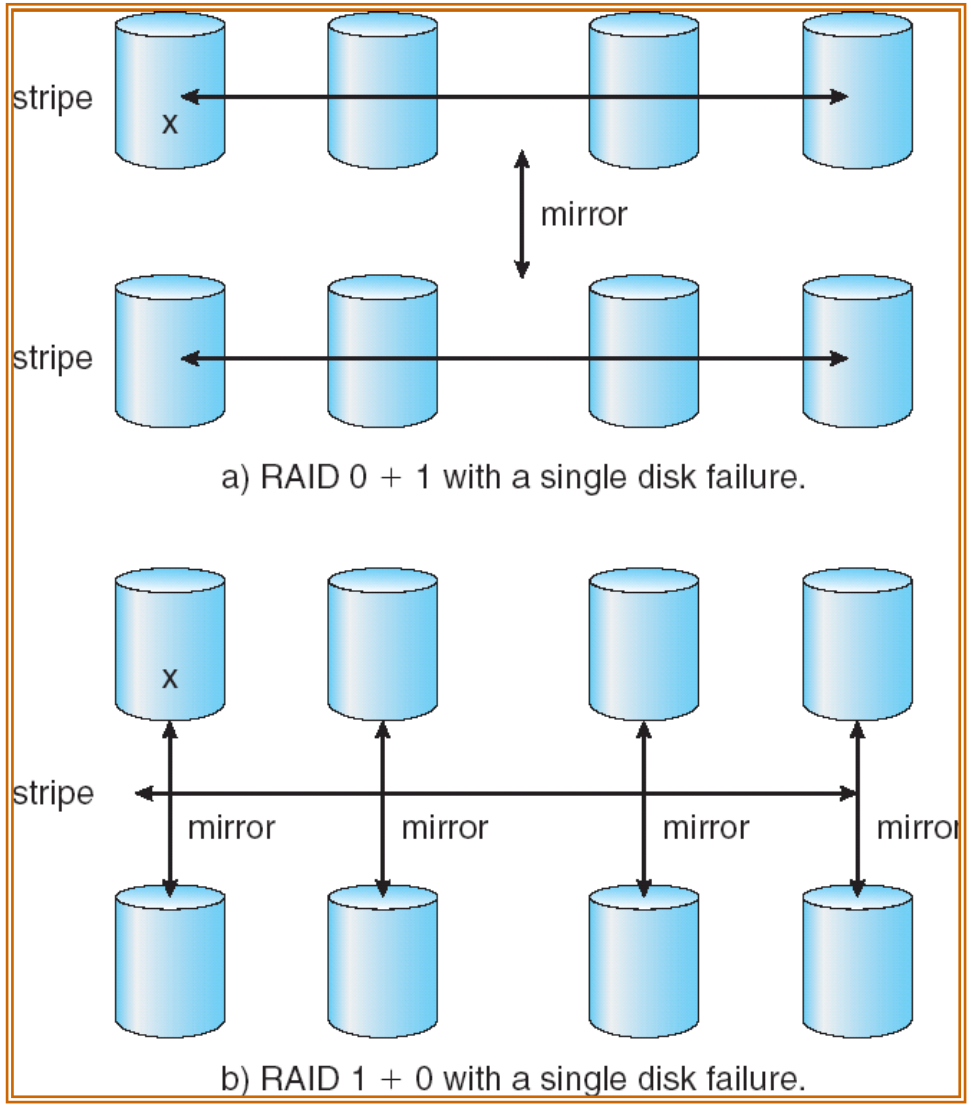
(d) RAID 3 (bit-interleaved parity)

Figure 11.9 RAID Levels (page 2 of 2)

Stallings

Parity bits for data recorded on one disk are saved to another disk

RAID (0 + 1) and (1 + 0)




Thank You!
Ευχαριστώ
ขอบคุณ
Vielen Dank
Merci
Teşekkürler
DMnvwd
شكراً
متشكرم
Gracias
Grazie
Bedankt
Dankie
THANK YOU
Köszönettel
Obrigado!
شکریا
Díky
謝謝
GADDA GUEY
WAD MAHAD
SAN TAHAY
감사합니다
Urakoze