Mass-Storage Systems

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 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 Scheduling

- 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 arm 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 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 the last transfer.

Disk Scheduling (Cont.)

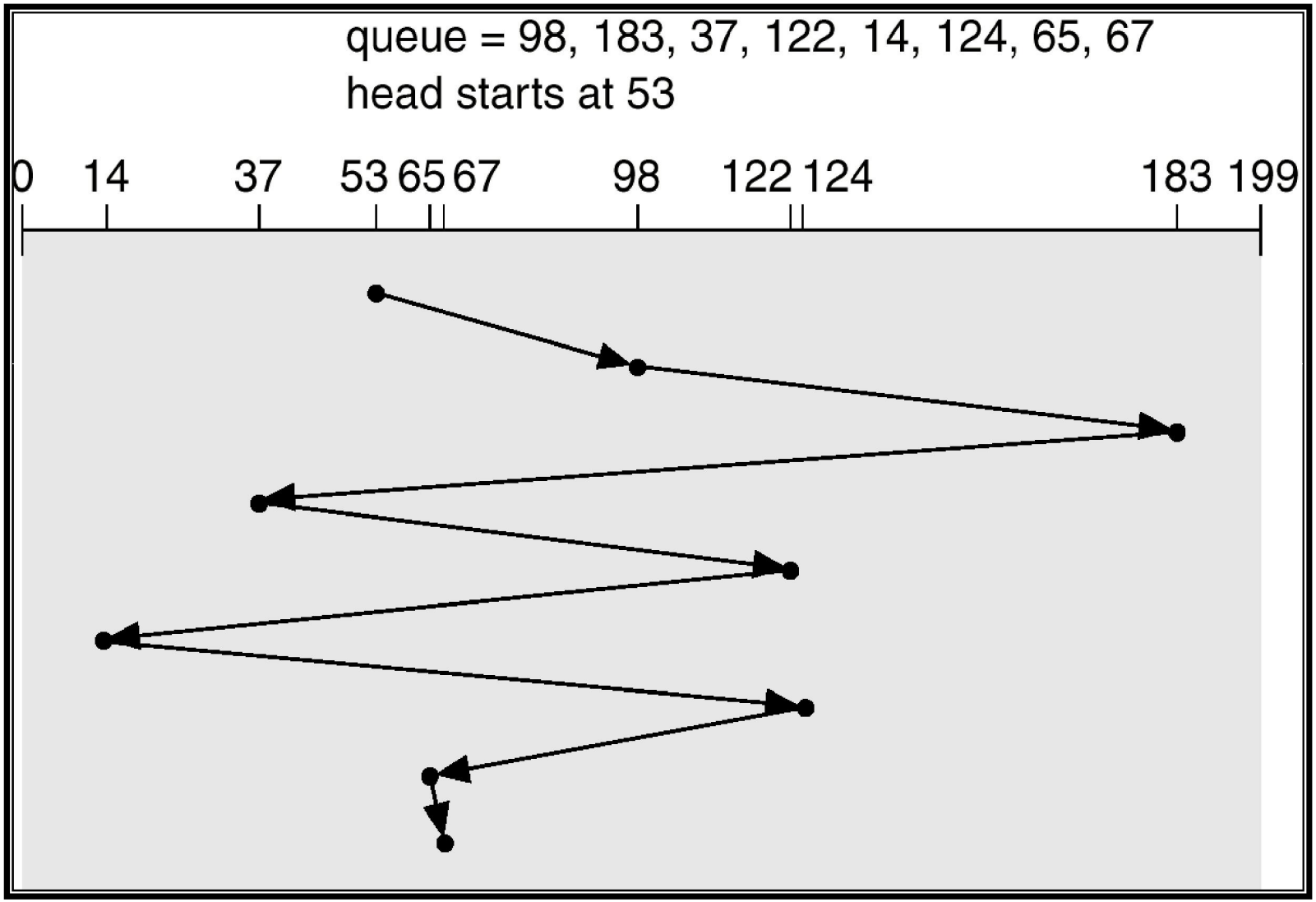
- Several algorithms exist to schedule the servicing of disk I/O requests.
- Example: request queue (0-199).

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

Head pointer 53

FCFS

Illustration shows total head movement of 640 cylinders.



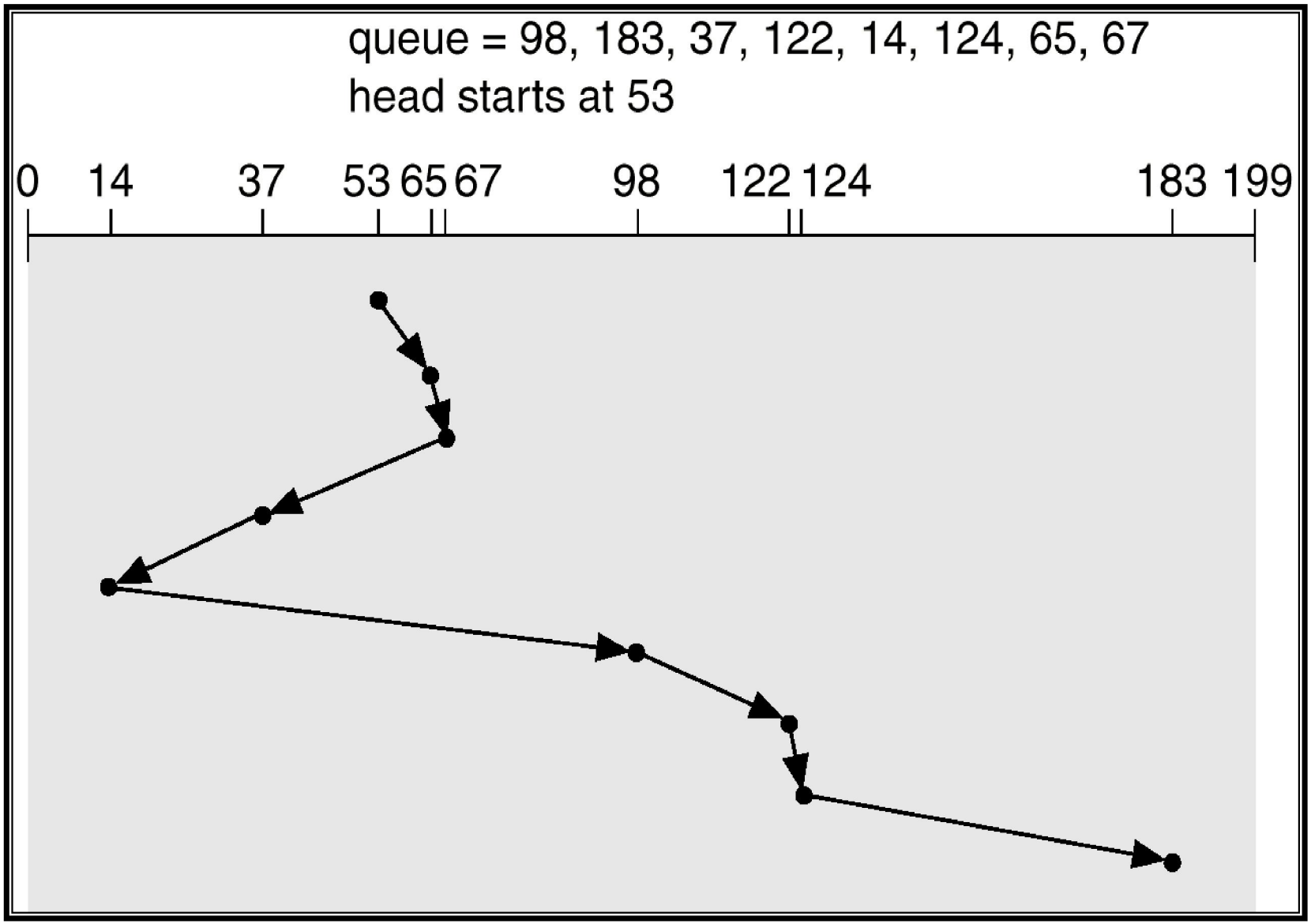
FCFS is fair

Does not provide fast service

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.
- Illustration shows total head movement of 236 cylinders.

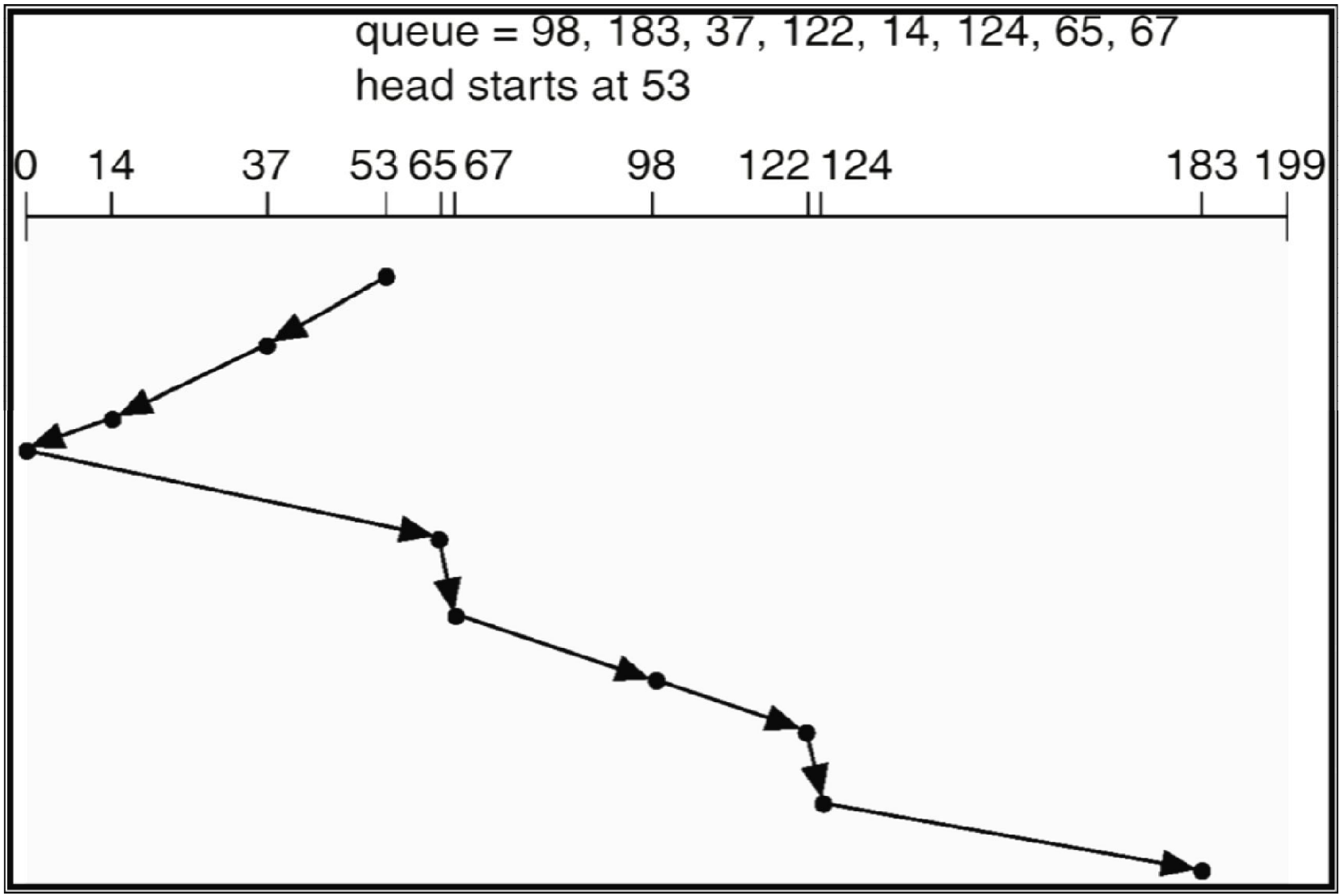
SSTF (Cont.)


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

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*.
- Illustration shows total head movement of 208 cylinders.

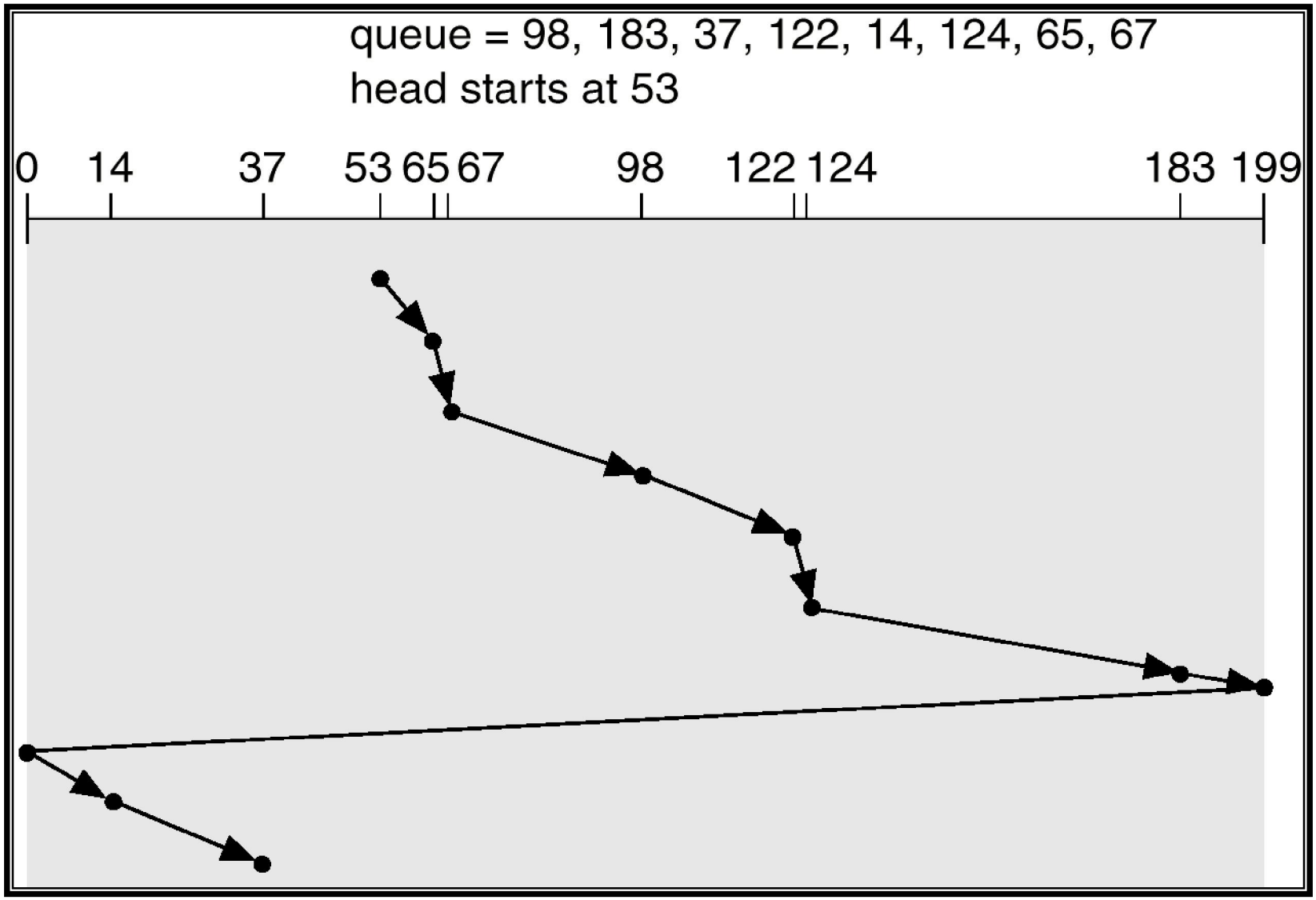
SCAN (Cont.)


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.

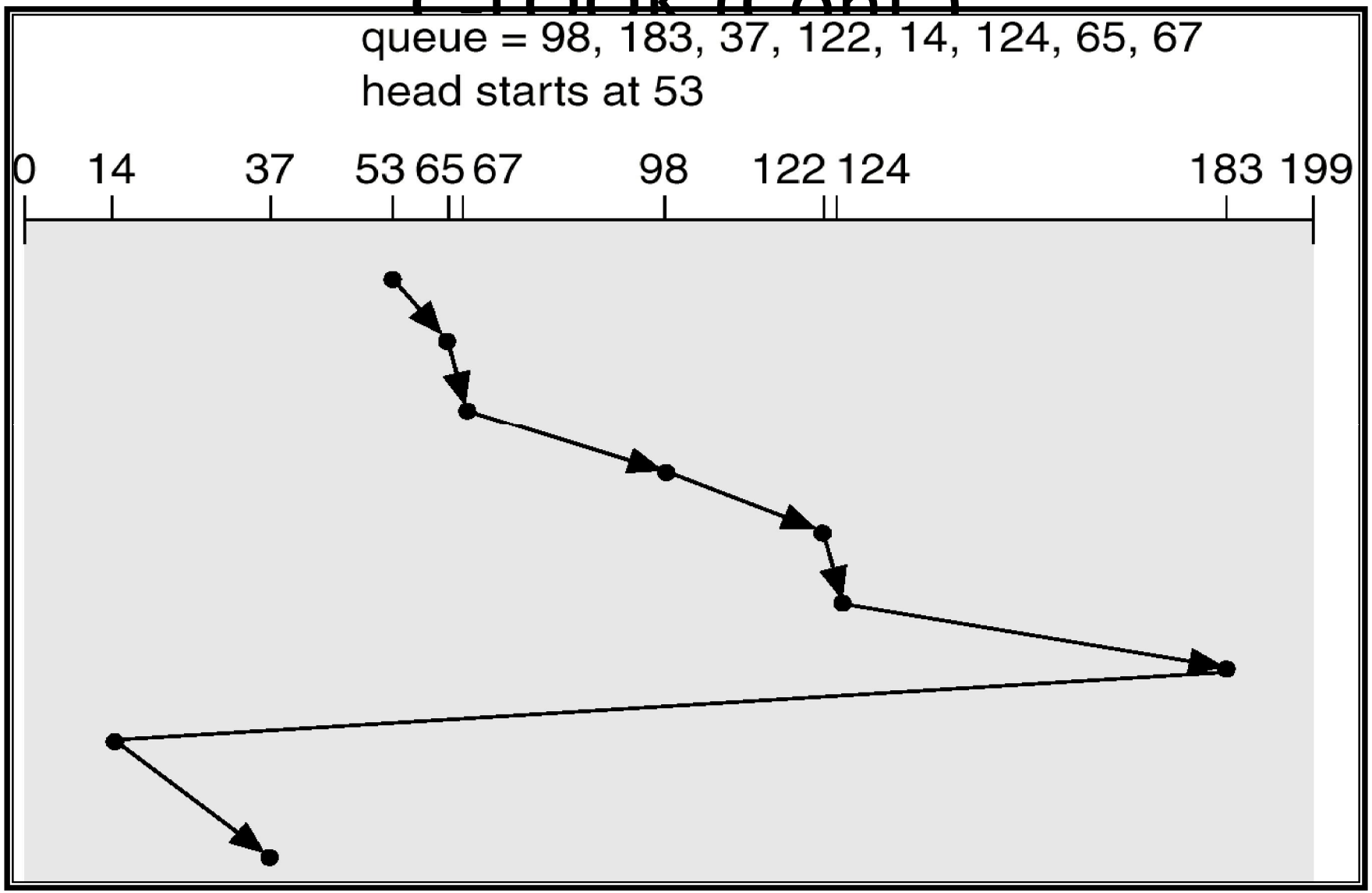
C-SCAN (Cont.)


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 LOOK is a reasonable choice for the default algorithm.

RAID Structure

- **RAID** – multiple disk drives provides **reliability** via **redundancy**
- RAID is arranged into seven different levels.

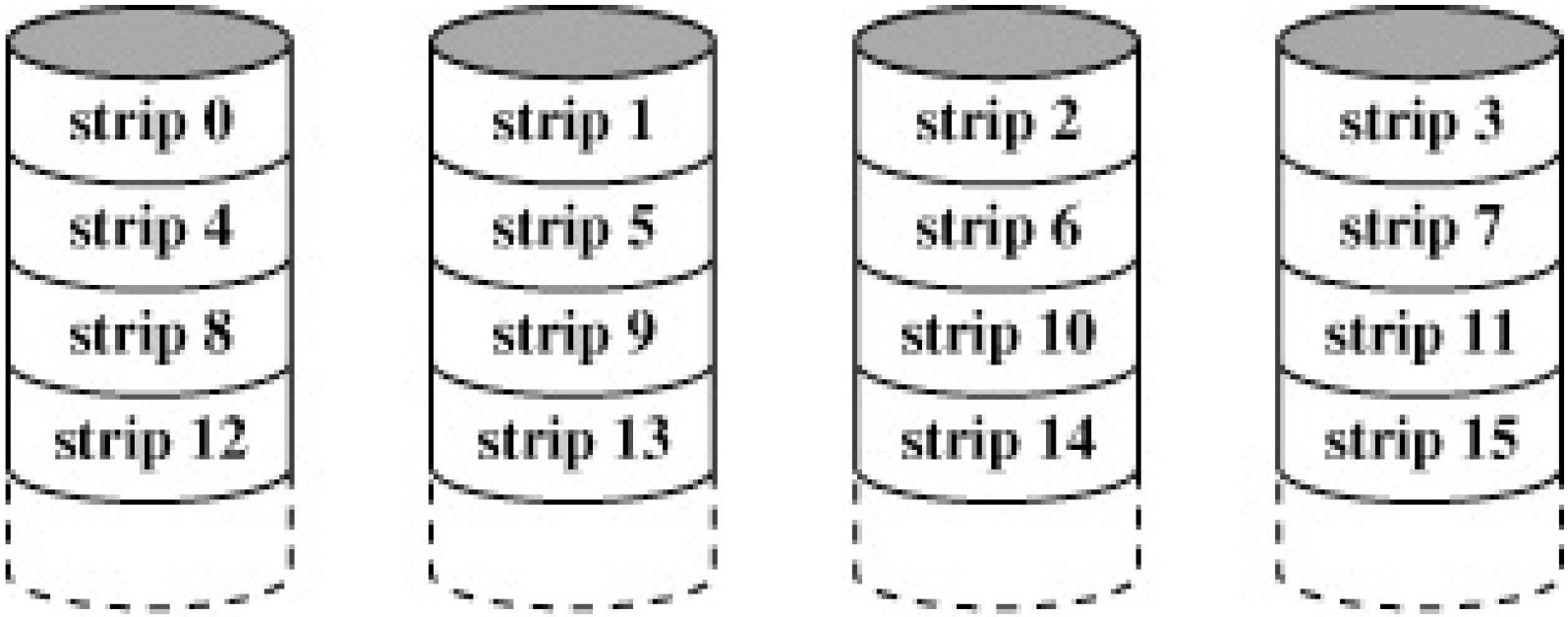
RAID (cont)

- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively.
- Disk striping uses a group of disks as one storage unit.
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data.
  - *Mirroring* or *shadowing* keeps duplicate of each disk.
  - *Block interleaved parity* uses much less redundancy.

- RAID is a set of physical drives viewed by OS as a single Logical drive
- Data is distributed across the physical drives of a array
- Redundant disk capacity is used for parity for data recovery
- RAID 0 and 1 do not have the third characteristics
- Should increase I/O handling capacity
- Should increase data handling capacity

RAID 0

- Does not include any redundancy
- Two different I/O requests can be issued in parallel
- The logical disk is divided into strips
- These strips may be physical blocks, sectors or some other unit
- Increases I/O handling capacity
- Increases data handling capacity



(a) RAID 0 (non-redundant)

Figure 11.9 RAID Levels (page 1 of 2)

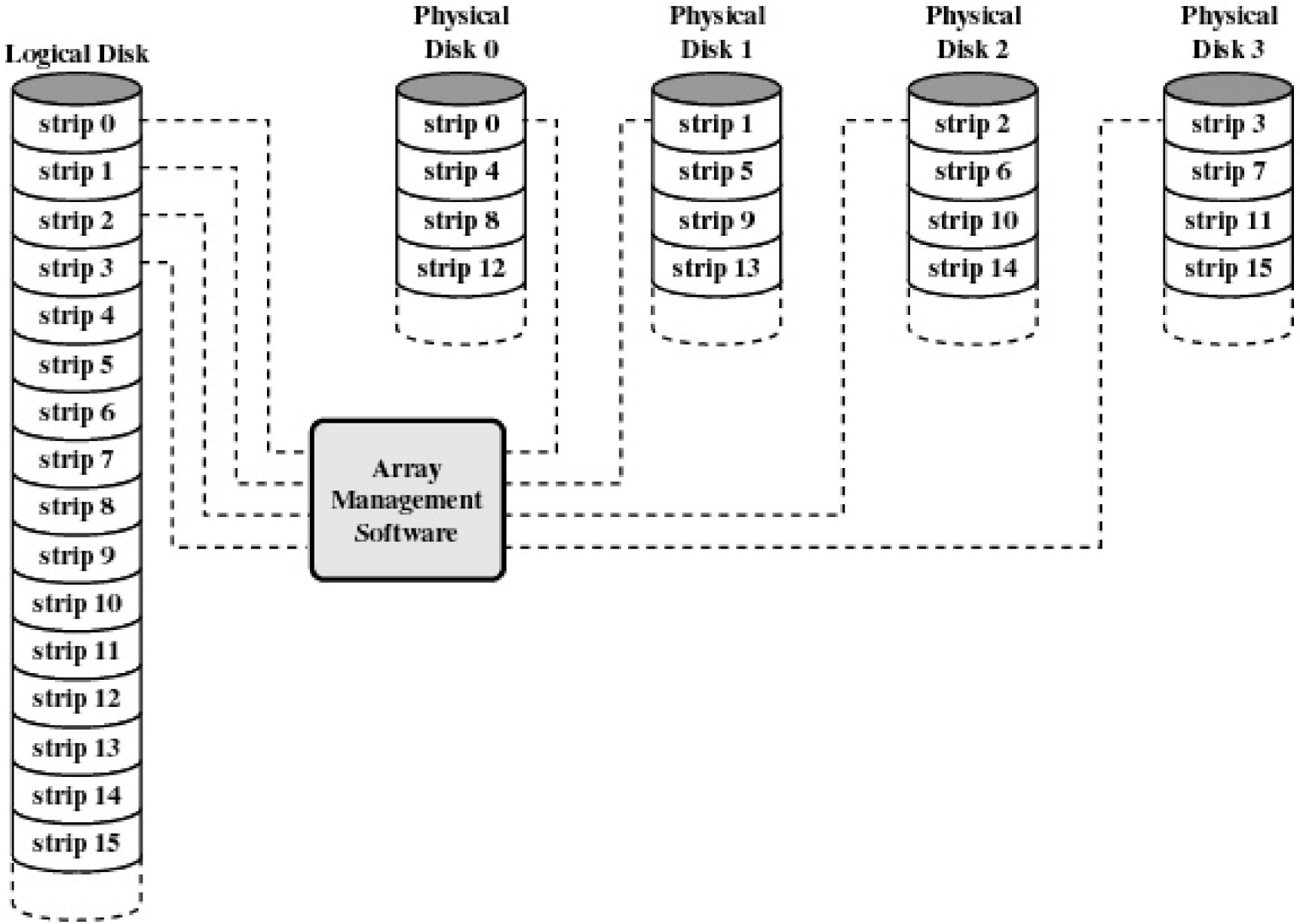
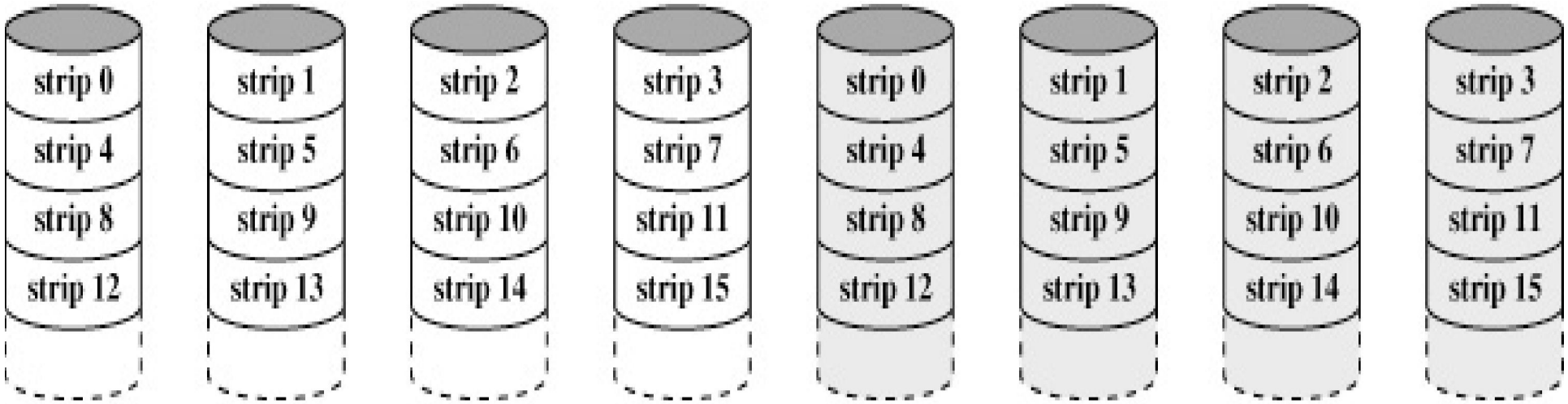


Figure 11.10 Data Mapping for a RAID Level 0 Array [MASS97]

RAID 1

- Uses Mirroring
- A read request can be serviced by either of two disks whichever has lesser service time
- A write request requires that both corresponding strips be updated. This can be done in parallel. There is no write penalty as no parity is calculated.
- Recovery from failure is simple.
- High data transfer and I/O handling rate.

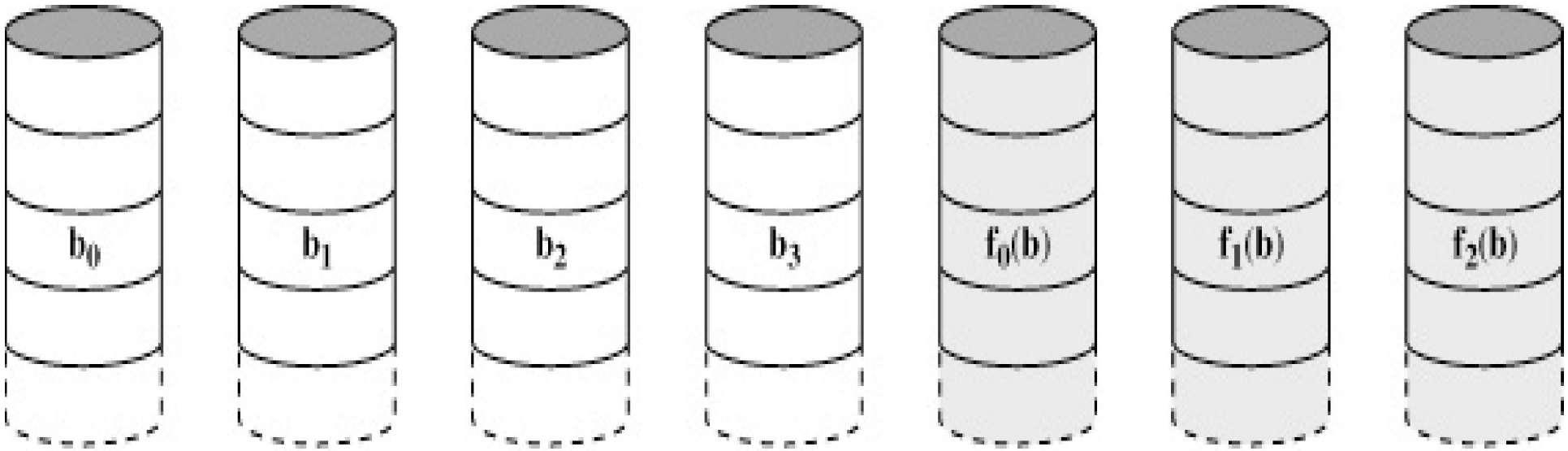


(b) RAID 1 (mirrored)

Figure 11.9 RAID Levels (page 1 of 2)

RAID 2

- RAID 2 and 3 make use of parallel accessing technique.
- All member disks participate in every I/O request
- The disk heads are synchronized
- In RAID 2 and 3 the strips are very small (byte or word)
- Multibit parity is calculated using Hamming code
- Number of disks used is less than RAID 1 but still costly
- All disks are simultaneously accessed
- RAID 2 is not used

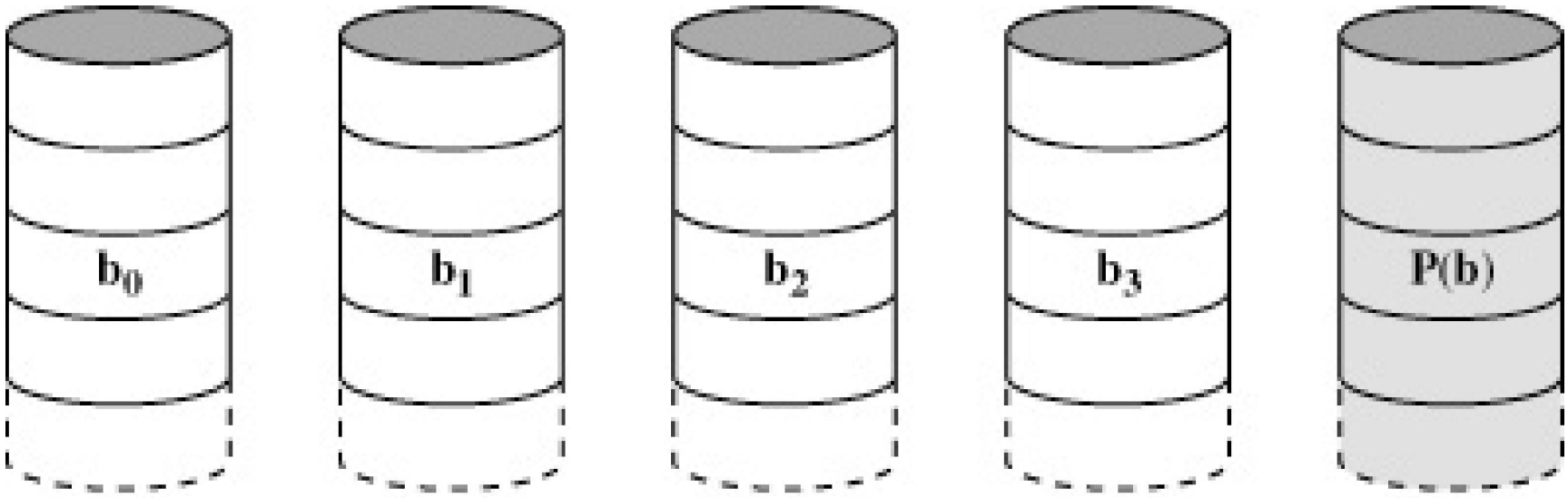


(c) RAID 2 (redundancy through Hamming code)

Figure 11.9 RAID Levels (page 1 of 2)

RAID 3

- Require only 1 redundant disk
- Simple parity bit is used
- High data transfer rate
- Only 1 I/O request can be handled at a time



(d) RAID 3 (bit-interleaved parity)

Figure 11.9 RAID Levels (page 2 of 2)

$$X4(i) = X3(i) \oplus X2(i) \oplus X1(i) \oplus X0(i)$$

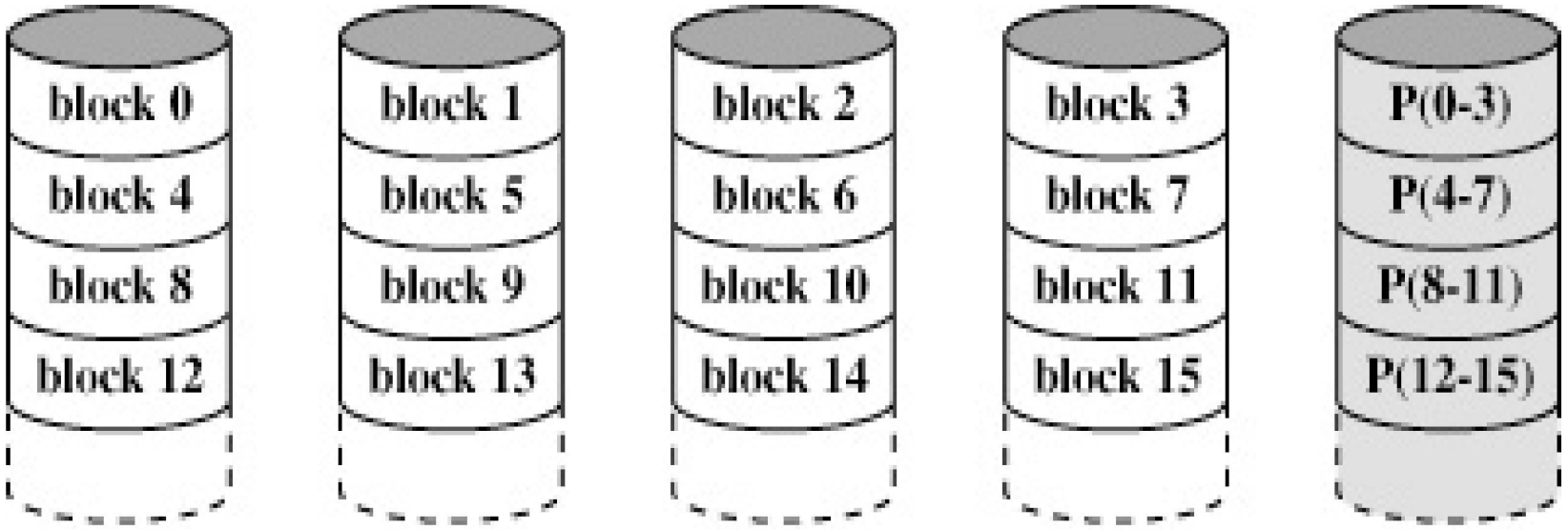
Suppose Disk X1 fails

By adding $X4(i) \oplus X1(i)$ on both sides we get

$$X1(i) = X4(i) \oplus X3(i) \oplus X2(i) \oplus X0(i)$$

RAID 4

- Each member disk operates independently
- Can handle high I/O request rates
- Each WRITE requires two READS and 2 WRITES, which is a bottleneck



(e) RAID 4 (block-level parity)

Figure 11.9 RAID Levels (page 2 of 2)

$X4(i) = X3(i) \oplus X2(i) \oplus X1(i) \oplus X0(i)$

Suppose a write is performed which only involves a strip on disk X1.

Thus

$X4'(i) = X3(i) \oplus X2(i) \oplus X1'(i) \oplus X0(i)$

$= X3(i) \oplus X2(i) \oplus X1'(i) \oplus X0(i) \oplus X1(i) \oplus X1(i)$

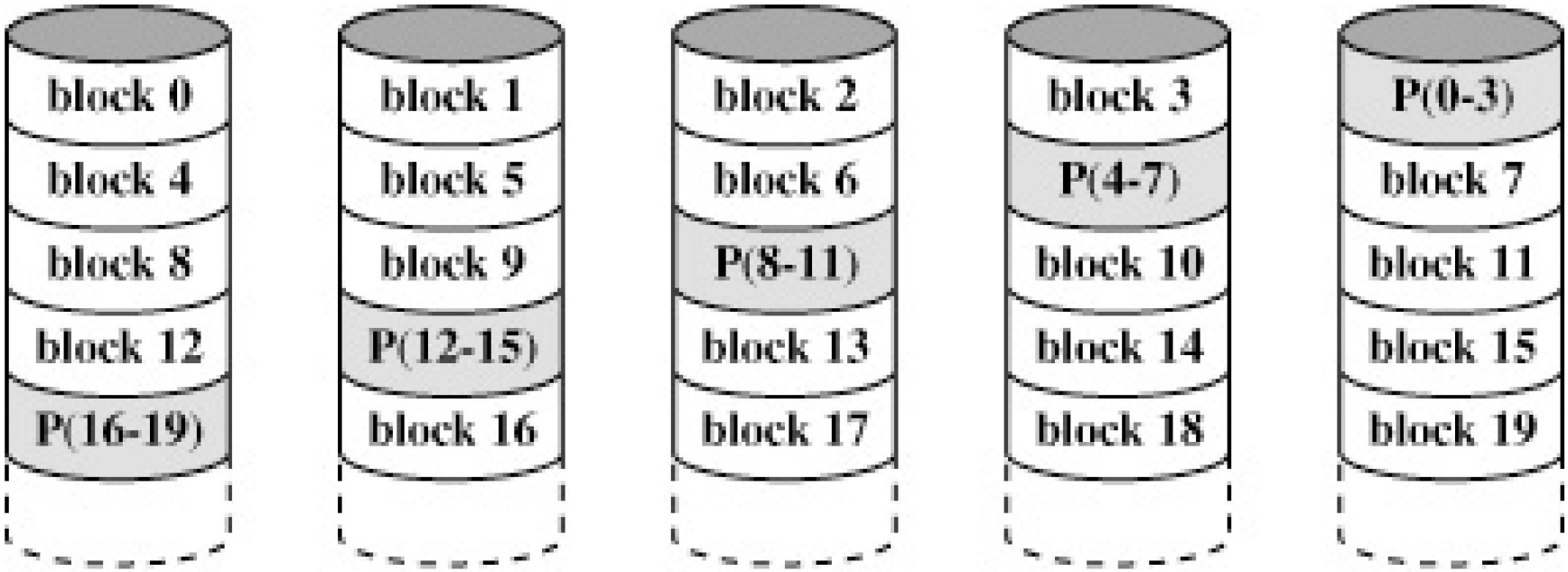
$= X4(i) \oplus X1(i) \oplus X1'(i)$

RAID 5

Data distribution similar to RAID 4

Parity is rotated

I/O bottleneck of single parity disk is avoided



(f) RAID 5 (block-level distributed parity)

Figure 11.9 RAID Levels (page 2 of 2)

RAID 6

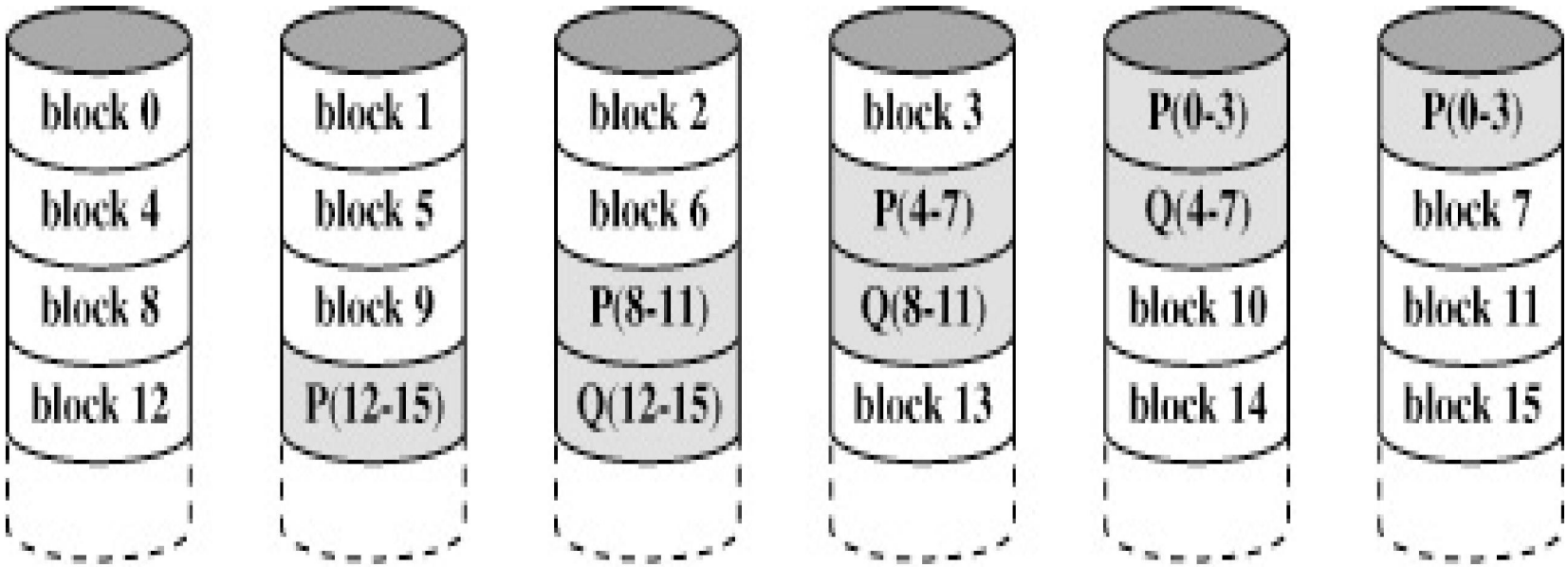
Two different parity calculations are carried out and stored in separate blocks on different disks

N+2 disks are required

Can take care of two disk failures

Provides extremely high data availability

Incurs a substantial write penalty



(g) RAID 6 (dual redundancy)

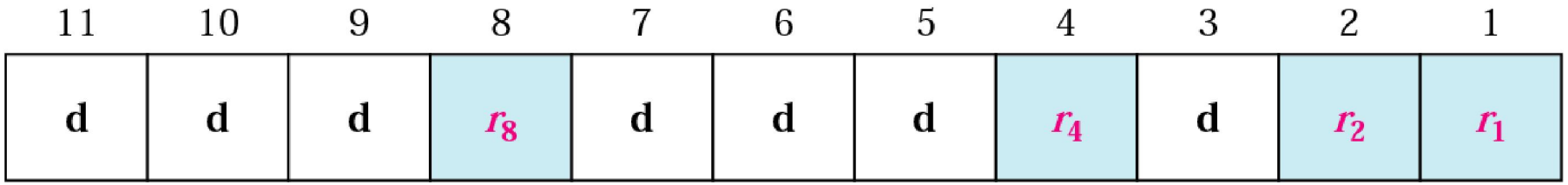
Figure 11.9 RAID Levels (page 2 of 2)

Hamming Code

Data and redundancy bits

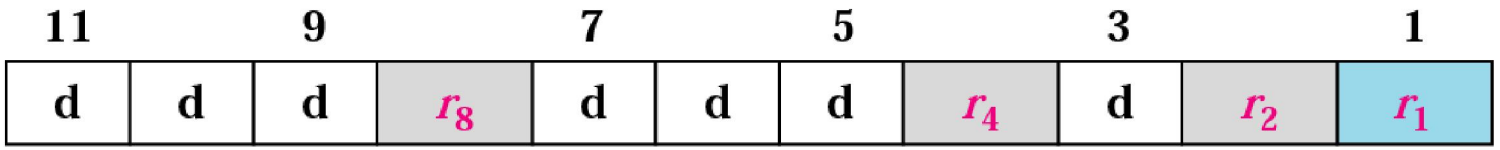
Number of data bits m	Number of redundancy bits r	Total bits m + r
1	2	3
2	3	5
3	3	6
4	3	7
5	4	9
6	4	10
7	4	11

Positions of redundancy bits in Hamming code

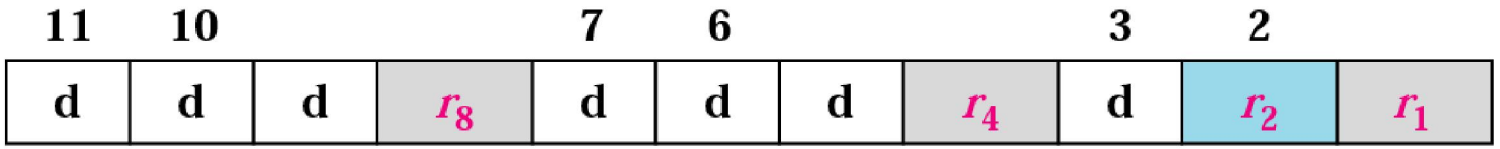


Redundancy bits calculation

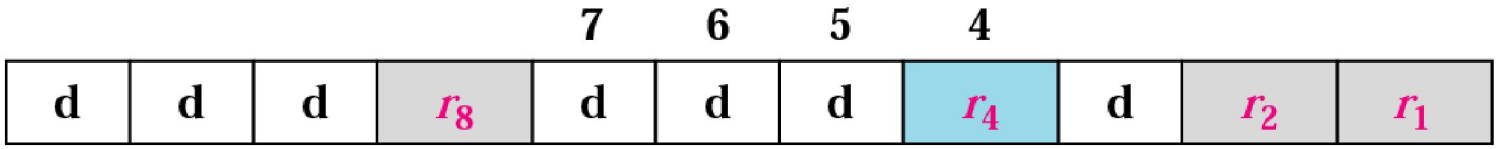
r_1 will take care of these bits.



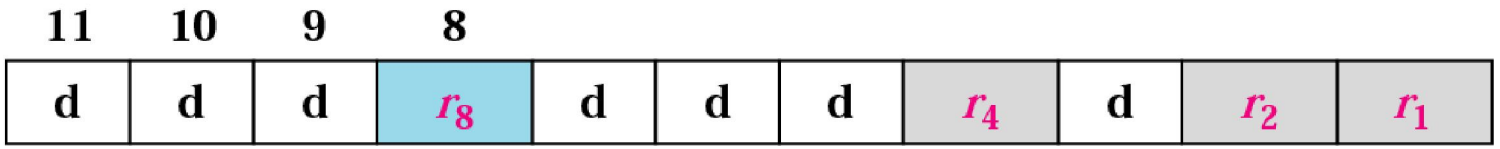
r_2 will take care of these bits.



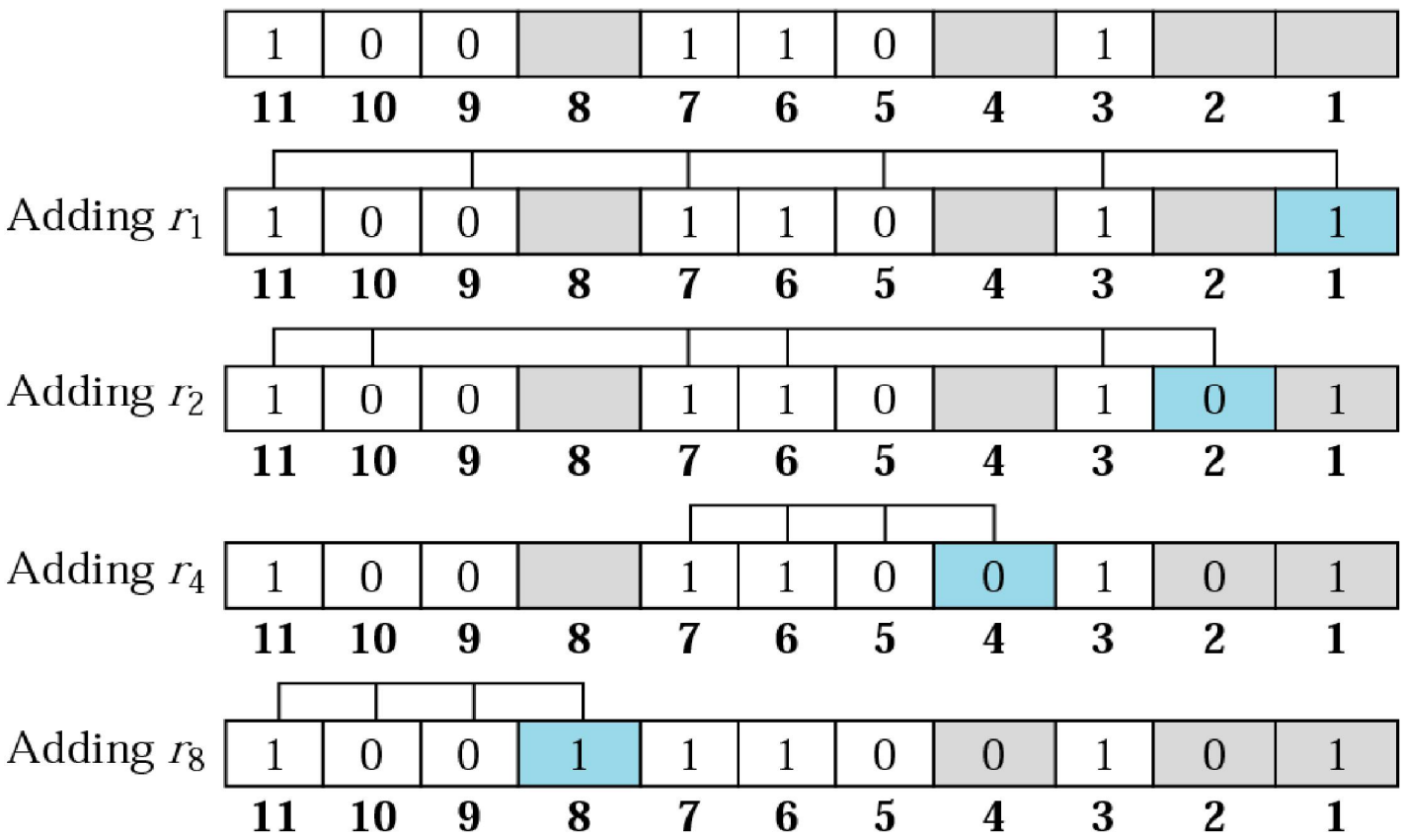
r_4 will take care of these bits.



r_8 will take care of these bits.



Example of redundancy bit calculation



Data:
1 0 0 1 1 0 1

Code:
1 0 0 1 1 1 0 0 1 0 1

Error detection using Hamming code