7.10.2 Disk Scheduling Algorithms

One of the responsibilities of the operating system is to use the hardware efficiently, which means that we should have fast disk access time and a broader disk bandwidth. Whenever a process needs I/O to or from the disk, it issues a system call to the operating system. The request specifies several pieces of information, such as:

1. Whether this operation is input or output?
2. What the disk address for the transfer is?
3. What the memory address for the transfer is?
4. What the number of bytes to be transferred is?

If the desired disk drive and controller are available, the request can be serviced immediately. If the driver or controller is busy, any new requests for service will be placed on the queue of pending requests for that device. We require disk-scheduling algorithms to service these pending requests.

Problem 7.14: Consider, for example, a disk queue with requests for I/O to blocks on cylinders

88, 83, 7, 13, 90, 79, 11, 63

The disk head is initially at cylinder 53.

Solution:

The head will first move from 53 to 88 (being the first request), then to 83, 7, 13, 90, 79, 11, and finally to 63, for a total head movement of 339 cylinders. This schedule is shown in Fig. 7.36.

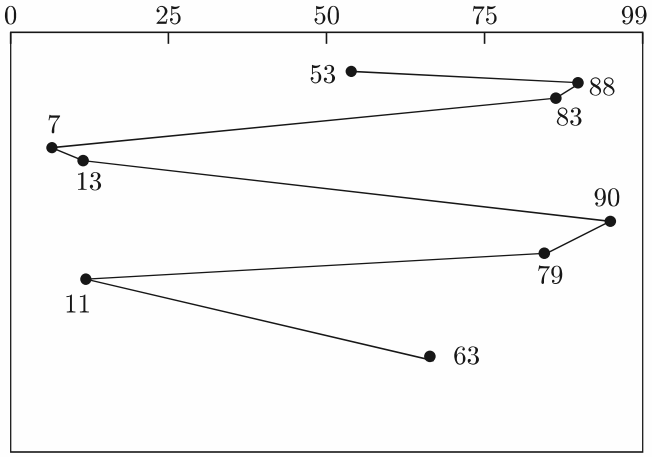

Figure 7.36 FCFS disk-scheduling algorithm.

Total head movement = (53∼88) + (88∼83) + (83∼7) + (13∼7) + (99∼13) + (79∼90) + (11∼79) + (11∼63)
= 35 + 5 + 76 + 6 + 86 + 11 + 68 + 52
= 339 cylinders

7.10.2.1 FCFS Scheduling

The simplest form of the disk scheduling is, of course, the first-come, first-served (FCFS) algorithm. This algorithm is intrinsically fair, but it generally does not provide the fastest service.

7.10.2.2 SSTF Scheduling

A variation of shortest job first (SJF), applied in CPU scheduling. It serves the request that has a minimum seek time from the current head position. To calculate the minimum seek time, we look for the minimum number of cylinders to be traversed. The major limitation is — starvation, as encountered in SJF algorithm.

Problem 7.15: Consider, the same problem, a disk queue with requests for I/O to blocks on cylinders

88, 83, 7, 13, 90, 79, 11, 63

The disk head is initially at cylinder 53.

Solution:

The head is positioned at cylinder 53, the minimum distance it has to move is 10 when it moves to cylinder 63, which is minimum, then from 63 the next access will be 79, then 83, to 88, then to 90, to 13, 11 and finally 7. This schedule is shown in Fig. 7.37.

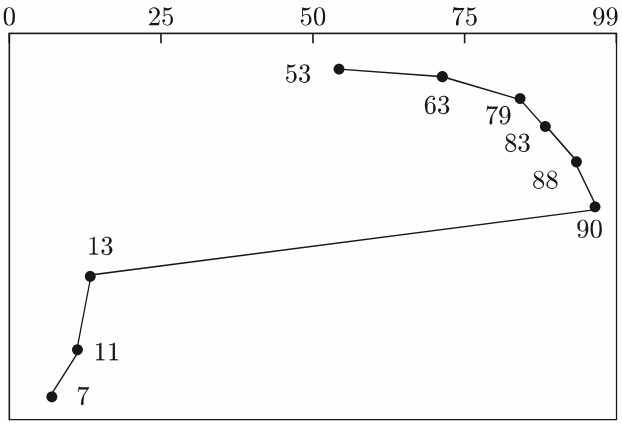

Figure 7.37 SSTF disk-scheduling algorithm.

Total head movement = (53∼63) + (63∼79) + (79∼83) + (83∼88) + (88∼90) + (90∼13) + (13∼11) + (11∼7)
= 10 + 4 + 4 + 5 + 2 + 77 + 2 + 4 = 108 cylinders

7.10.2.3 SCAN Scheduling

Also, called the elevator algorithm, here the disk arm begins serving the requests at one end and moves to the other end (it does not come back, as in earlier algorithms) and then returns back serving the requests, similar to an elevator. This technique eventually results in reduced variance as compared to SSTF.

Problem 7.16: Consider, the same previous problem, a disk queue with requests for I/O to block on cylinders

88, 83, 7, 13, 90, 79, 11, 63

The disk head is initially at cylinder 53 and the disk arm is moving towards 0.

Solution:

The head is positioned at cylinder 53 and the disk arm is moving towards 0. So, the cylinders accessed will be 12, 11 and 7, the disk arm will touch 0 and then will read 63, then 79, 83, 88 and finally 90. This schedule is shown in Fig. 7.38.

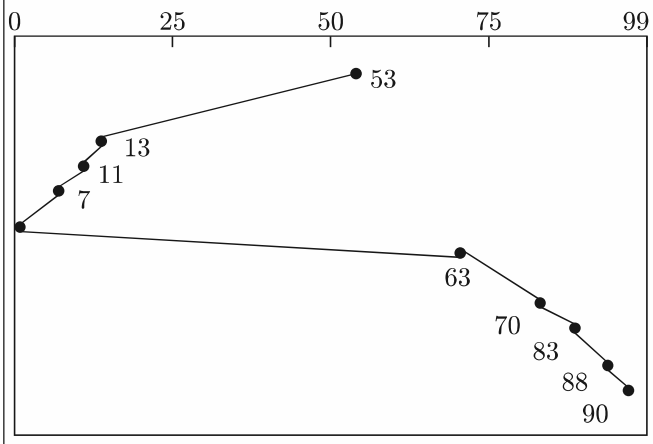

Figure 7.38 SCANdisk-scheduling algorithm.

Total head movement = (53∼13) + (13∼11) + (11∼7) + (7∼0) + (0∼63) + (63∼79) + (79∼83) + (83∼88) + (88∼90)

= 40 + 2 + 4 + 7 + 63 + 16 + 4 + 5 + 2
= 143 cylinders

It should take into account the head movement from cylinder 7 to 0 and 0 to 63 also.

7.10.2.4 C-SCAN Scheduling

This algorithm treats the cylinders as a circular list, hence provides a uniform wait time as compared to the SCAN algorithm discussed above. The major difference is that the requests are served in any one direction of the motion of the head, not in both the directions as in SCAN. This can be compared by looking at the following numerical problem.

Problem 7.17: Consider, the same problem, a disk queue with requests for I/O to blocks on cylinders

88, 83, 7, 13, 90, 79, 11, 63

The disk head is initially at cylinder 53 and the disk arm is moving towards 0.

Solution:

The head is positioned at cylinder 53 and the disk arm is moving towards 0. So, the cylinders accessed will be 13,11 and 7, the disk arm will touch 0 and then will return to the end (in this case) and start reading 90, 88, 83, 79, and finally read 63. This schedule is shown in Fig. 7.39.

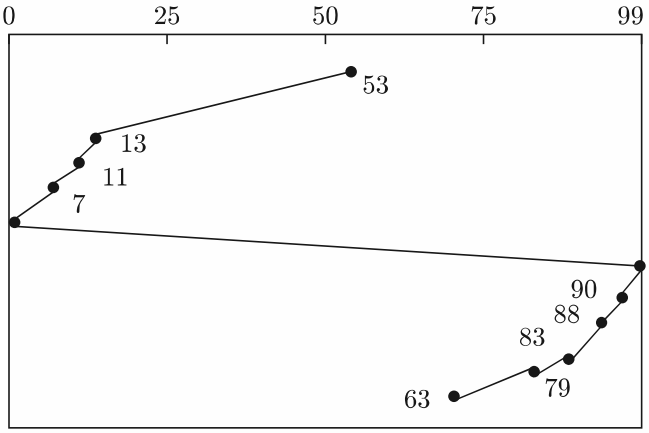

Figure 7.39 C-SCAN disk-scheduling algorithm.

Total head movement = (53∼13) + (13∼11) + (11∼7) + (7∼0) + (0∼99) + (99∼90) + (90∼88) + (88∼83) + (83∼79) + (79∼63)

= 40 + 2 + 4 + 7 + 99 + 9 + 2 + 5 + 4 + 16
= 188 cylinders

7.10.2.5 LOOK and C-LOOK Scheduling

LOOK and C-LOOK are the variants of SCAN and C-SCAN algorithm, respectively. Here the head does not move till the end of the cylinder, it only moves to the farthest cylinder which requires to be served. After serving the farthest request, it changes the direction (in case of LOOK) or jumps to another end (in case of C-LOOK). The same is illustrated below.

Problem 7.18: Consider, the same problem, a disk queue with requests for I/O to blocks on cylinders

88, 83, 7, 13, 90, 79, 11, 63

The disk head is initially at cylinder 53 and the disk arm is moving towards 0.

Solution:

The head is positioned at cylinder 53 and the disk arm is moving towards 0. So, the cylinders accessed will be 13, 11 and 7, now the disk arm will not touch 0 (as in SCAN algorithm) but will read 63, then 79, 83, 88 and finally 90. This schedule is shown in Fig. 7.40.

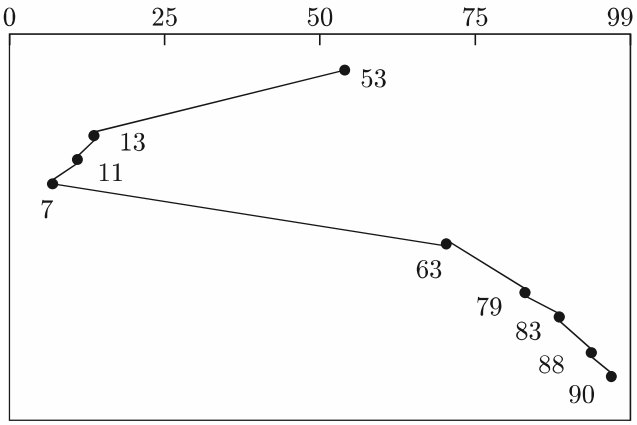

Figure 7.40 | LOOK disk-scheduling algorithm.

Total head movement = (53∼13) + (13∼11) + (11∼7) + (7∼63) + (63∼79) + (79∼83) + (83∼88) + (88∼90)

= 40 + 2 + 4 + 56 + 16 + 4 + 5 + 2 = 129 cylinder

Problem 7.19: Consider, the same example, a disk queue with requests for I/O to blocks on cylinders

88, 83, 7, 13, 90, 79, 11, 63

The disk head is initially at cylinder 53 and the disk arm is moving towards 0. Simulate the C-LOOK disk-scheduling algorithm.

Solution:

The head is positioned at cylinder 53 and the disk arm is moving towards 0. So, the cylinders accessed will be 13,11 and 7, the disk arm will not touch 0 but will return to 90, then read 88, 83, 79 and finally read 63. This schedule is shown in Fig. 7.41.

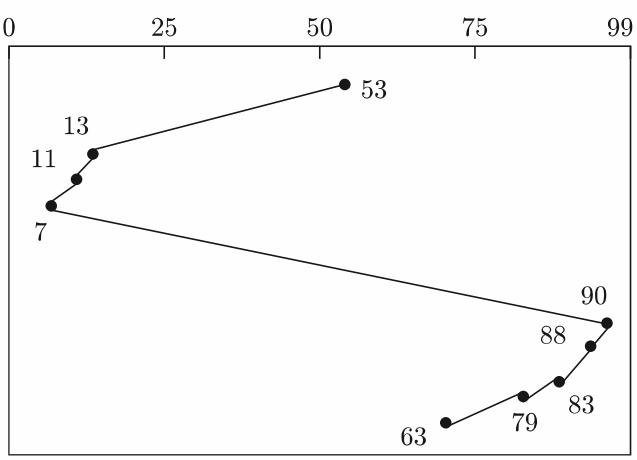

Figure 7.41 | C-LOOK disk-scheduling algorithm.

Total head movement = (53∼13) + (13∼11) + (11∼7) + (7∼90) + (90∼88) + (88∼83) + (83∼79) + (79∼63)

= 40 + 2 + 4 + 83 + 2 + 5 + 4 + 16 = 156 cylinder

7.10.2.6 Selection of a Disk-Scheduling Algorithm

Given so many disk-scheduling algorithms, how do we choose the best one? SSTF is common and has a natural appeal because it increases performance over FCFS. SCAN and C-SCAN perform better for system that place a heavy load on the disk, because they are less likely to have starvation problem. Therefore, it is dependent on the type of application we want to use.

7.11 PROTECTION AND SECURITY

As huge information is stored in computer systems, the need to protect it is equally important. Usually, security and protection are used interchangeably.

7.11.1 Security

There are three important aspects, known as CIA triad, which are basic to providing security. These are:

1. **Confidentiality:** It involves data confidentiality as well as user privacy.
2. **Integrity:** It involves two terms—data integrity (which means that data is handled in an authorized manner) and system integrity (which means