Indian Scholar
ISSN 2350-109X
www.indianscholar.co.in
An International Multidisciplinary Research e-Journal

A COMPARATIVE ANALYSIS OF DISK SCHEDULING ALGORITHMS

Sourabh Shastri
Department of Computer Science & IT,
Bhaderwah Campus, University of Jammu,
J&K, India

Anand Sharma
Department of Computer Science & IT,
University of Jammu, J&K,
India

Prof. Vibhakar Mansotra
Department of Computer Science & IT,
University of Jammu,
J&K, India

Abstract

Disk scheduling is a policy of operating system to decide which I/O request is going to be satisfied foremost. The goal of disk scheduling algorithms is to maximize the throughput and minimize the response time. The present piece of investigation documents the comparative analysis of six different disk scheduling algorithms viz. First Come First Serve, Shortest Seek Time First, Scan, Look, C-Scan and C-look disk scheduling by comparing their head movement in different runs. The implementation is carried out in Turbo C by creating an interface to calculate total head movement of these six algorithms.

I Introduction:

The file system can be viewed logically in three different divisions i.e. user, programmer interface to the file system and secondary storage structure. The lowest level of the file system is secondary storage structure and disk is the main secondary storage device that is generally divided into tracks, cylinders and sectors and stores the data permanently. The I/O operation depends on the computer system, the operating system, and the nature of the I/O channel and disk controller hardware [1]. The user programs make use of the data on the disk by means of I/O requests. Data is stored on both surfaces of a series of magnetic disks called platters that are connected by a single spindle. The surface of a platter is logically divided into tracks that are further subdivided into sectors and the set of tracks that are at one arm position form a cylinder. One read-write head per disk surface is used to access the data and all read-write heads are

attached to a single moving arm. The segment of the disk surface where the data is read or written must revolve under the read-write head for accessing the data.

The key responsibility of the operating system is to efficiently use the hardware of the computer system. For the efficiency of the disk drives, the terms access time and disk bandwidth are associated. The access time is the total time elapsed between the access command and the read/write head positioned to the particular sector or in other words it is the combination of seek time, latency time and transfer time. Seek time is the time to move the head to the right data track. Latency time is the time taken for desired sector to rotate under head for access. Transfer time is the actual time required to transfer data between disk and main memory. Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request of the service and the finishing point of the last transfer [2]. For most disks, the seek time leads the latency time and transfer time, so reducing the mean seek time can improve system performance to a large extent [3].

In multiprogramming systems, processes running concurrently may generate requests for reading and writing disk records. The operating system handles these I/O requests from the queue and processes them one by one. The algorithm used to choose which I/O request is going to be fulfilled earliest is called disk scheduling algorithm. The different disk scheduling algorithms are First Come First Serve, Shortest Seek Time First, Scan, Look, Circular Scan and Circular Look. The main objectives for any disk scheduling algorithm are minimizing the response time and maximizing the throughput.

In this research paper, an experiment has been carried out by considering the same request queue for implementing the different disk scheduling algorithms.

II Working Procedure of Disk Scheduling Algorithms:

Disk scheduling algorithms applied to decide which I/O request is going to be satisfied first. The fundamental disk scheduling algorithms are First Come First Serve, Shortest Seek Time First, Scan, Look, Circular Scan and Circular Look. The procedures of these algorithms are described here:

First Come First Serve Disk Scheduling Algorithm (FCFS):

The functioning of this algorithm is maintained by First in First out (FIFO) queue. With this scheme, the I/O requests are served or processed according to their arrival [4]. Though this algorithm improves response time but fails to decrease the average seek time because this algorithm needs a lot of random head movements and disk rotations. Let us consider the following track requests in the disk queue to compute the total head movement of the read/write head and let us assume that the current read/write head position is at location 100 for a 200 track disk (0-199).

Track Requests: 25, 90, 135, 50, 190 and 60.

In First Come First Serve algorithm, the read/write head initially move from current location 100 to 25, then 25 to 90, then 90 to 135, then 135 to 50, then 50 to 190 and at last 190 to 60.

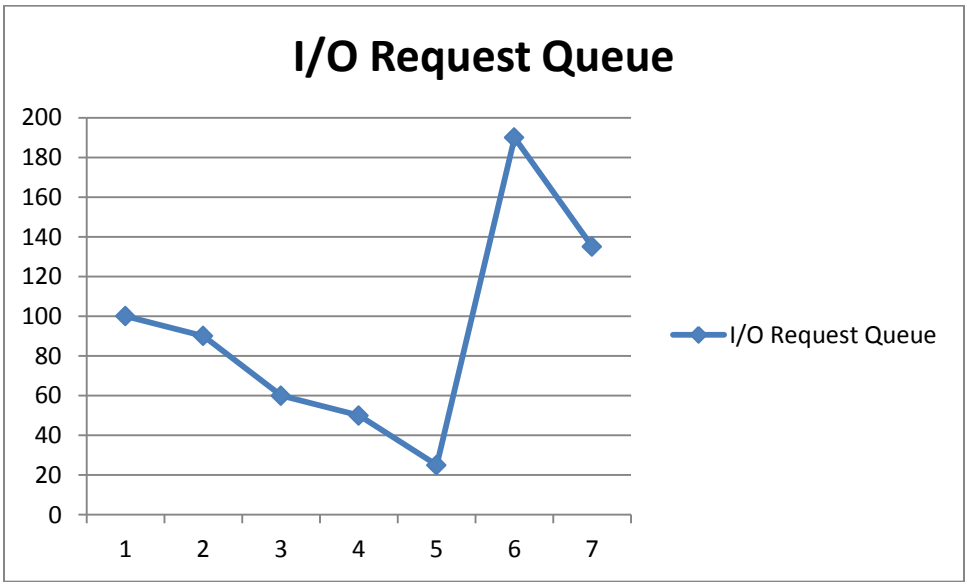

Figure 1: First Come First Serve Representation.

Total Head Movement = (100-25) + (90-25) + (135-90) + (135-50) + (190-50) + (190-60) = 540 tracks. The major disadvantage of FCFS disk scheduling algorithm is that it raises the mean seek time because disk arm has to cover long distance to accomplish a request as in this case movement of arm from 135 to 50 and then from 50 to 190.

Shortest Seek Time First Disk Scheduling Algorithm (SSTF):

In this approach, the read/write head serves the request first that have minimum seek time from the current head position. The I/O requests that are close to the read/write head are serviced first. SSTF disk scheduling algorithm is actually a form of SJF scheduling algorithm and may cause starvation of some requests [5]. SSTF disk scheduling algorithm has better throughput than FCFS disk scheduling algorithm but some requests may be delayed for a long time if lots of closely situated requests arrive just after it. Let us consider the same track requests in the disk queue to compute the total head movement of the read/write head and let us again assume that the current read/write head position is at location 100 for a 200 track disk (0-199).

Track Requests: 25, 90, 135, 50, 190 and 60.

In Shortest Seek Time First algorithm, the read/write head initially move from current location 100 to 90, then 90 to 60, then 60 to 50, then 50 to 25, then 25 to 135 and at last 135 to 190.

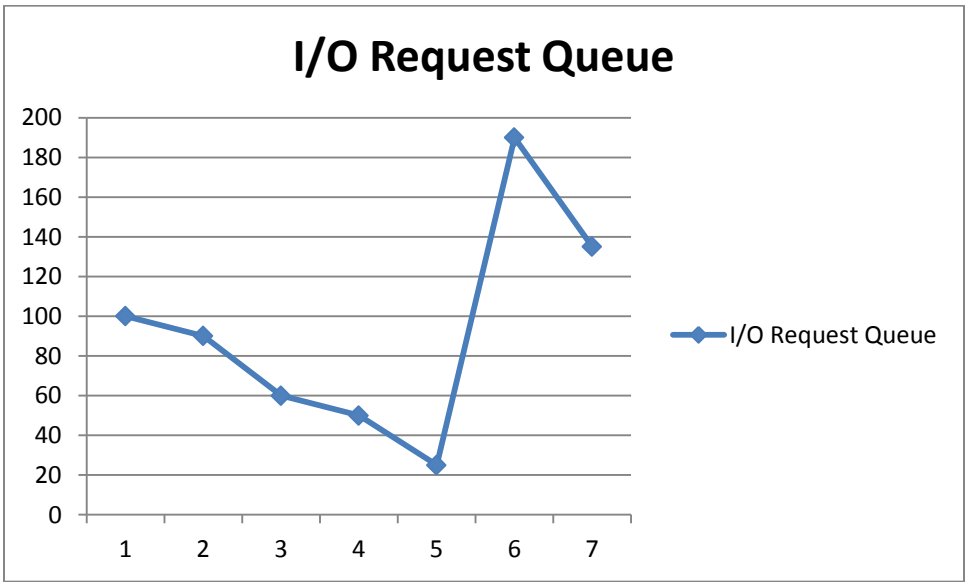

Figure 2: Shortest Seek Time First Representation.

Total Head Movement = (100-90) + (90-60) + (60-50) + (50-25) + (135-25) + (190-135) = 240 tracks. The major disadvantage of SSTF disk scheduling algorithm is that some requests have to wait for a long time if new requests with shorter seek time keep arriving but the advantage is that the throughput is higher as compared to FIFO disk scheduling algorithm and produces less head movements.

Scan Disk Scheduling Algorithm (SCAN):

In Scan disk scheduling algorithm, the read/write head starts from one end and move towards the other end and servicing requests as it reaches each track until it reaches to other end of the disk. The direction of the read/write head reverses after reaching at the other end and servicing continues. In this way, the read/write head continuously swings from end to end [6]**.** Let us consider the same track requests in the disk queue to compute the total head movement of the read/write head and let us again assume that the current read/write head position is at location 100 for a 200 track disk (0-199).

Track Requests: 25, 90, 135, 50, 190 and 60.

In Scan disk scheduling algorithm, the read/write head initially move from current location 100 to 90, then 90 to 60, then 60 to 50, then 50 to 25, then 25 to 0 and then to the other direction from 0 to 135 and at last 135 to 190. It services all the requests at either side firstly then moves the other way that's why it is sometimes called as elevator algorithm.

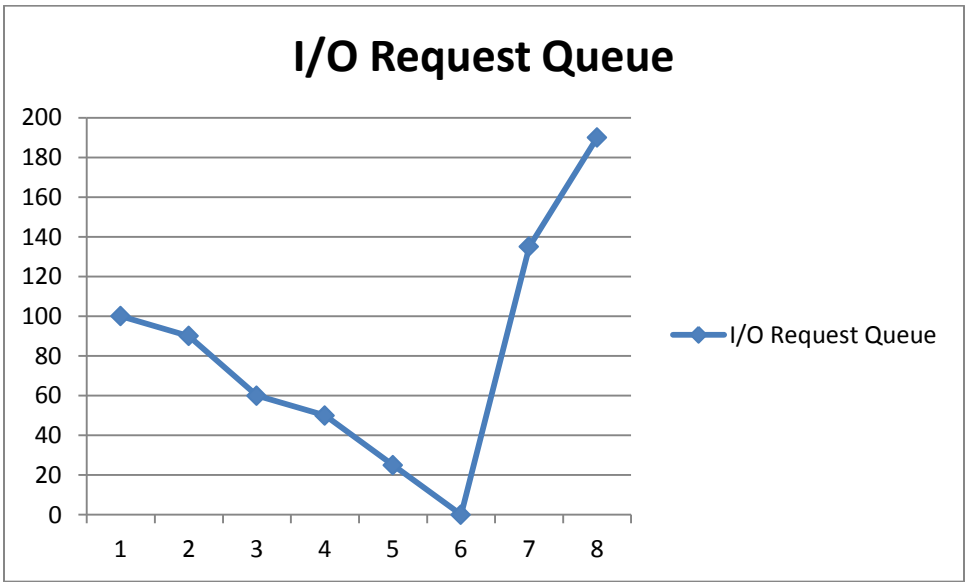

Figure 3: SCAN Representation.

Total Head Movement = (100-90) + (90-60) + (60-50) + (50-25) + (25-0) + (135-0) + (190-135) = 290 tracks. The major disadvantage of Scan disk scheduling algorithm is that disk arm always starts from the beginning towards one end even other numbers of requests are present on the other end. The throughput of Scan algorithm is better than FCFS and also prevents starvation.

Look Disk Scheduling Algorithm (LOOK):

Look disk scheduling algorithm is the improved version of Scan disk scheduling algorithm and avoids the starvation problem. In Look disk scheduling algorithm, the arm goes only as far as final requests in each direction and then reverses direction without going all the way to the end [7]. In this way it improves both response time and throughput. Let us consider the same track requests in the disk queue to compute the total head movement of the read/write head and let us again assume that the current read/write head position is at location 100 for a 200 track disk (0-199).

Track Requests: 25, 90, 135, 50, 190 and 60.

In Look disk scheduling algorithm, the read/write head initially move from current location 100 to 90, then 90 to 60, then 60 to 50, then 50 to 25, then to the other direction from 25 to 135 and then at last from 135 to 190.

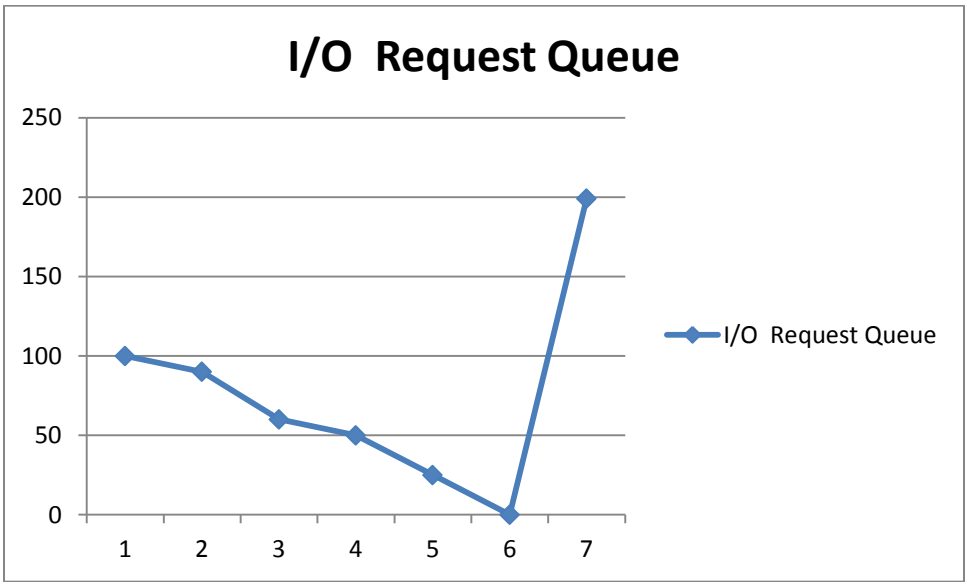

Figure 4: LOOK Representation.

Total Head Movement = (100-90) + (90-60) + (60-50) + (50-25) + (135-25) + (190-135) = 240 tracks. The major advantage of Look disk scheduling algorithm is that it improves response time and throughput than Scan disk scheduling algorithm.

Circular Scan Disk Scheduling Algorithm (C-SCAN):

Cyclic Scan or Circular Scan disk scheduling algorithm is an improved version of Scan disk scheduling algorithm and known as one directional scan. It starts its scan towards the nearest end and services the requests all the way to the end [8] Let us consider the same track requests in the disk queue to compute the total head movement of the read/write head and let us again assume that the current read/write head position is at location 100 for a 200 track disk (0-199).

Track Requests: 25, 90, 135, 50, 190 and 60.

In C-Scan disk scheduling algorithm, the read/write head initially move from current location 100 to 90, then 90 to 60, then 60 to 50, then 50 to 25, then 25 to 0 and then to the other direction from 0 to 199, then 199 to 190 and at last 190 to 135. It services all the requests at either side firstly up to the end and then moves the other end's without fulfilling any requests in the way and then start servicing as shown in the figure below.

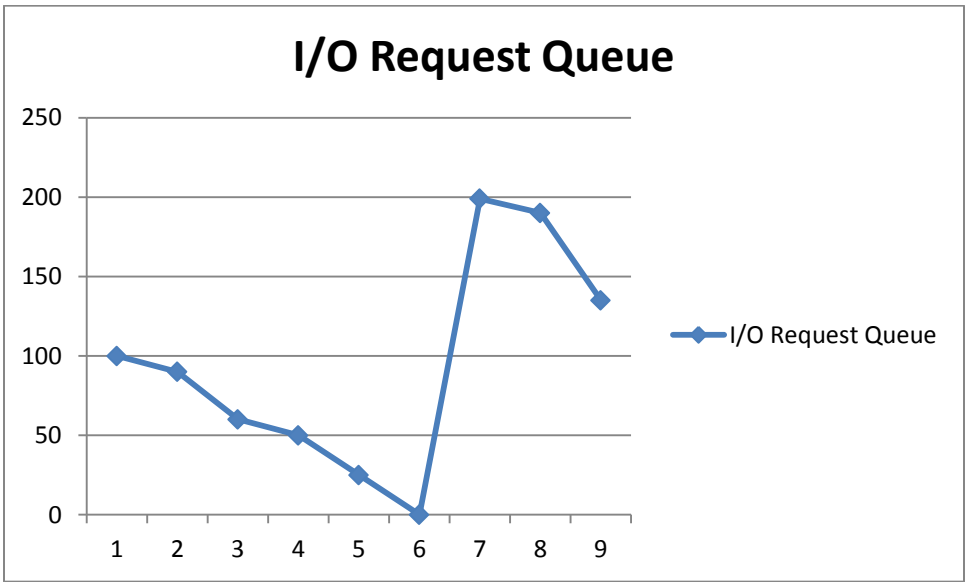

Figure 5: C-Scan Representation.

Total Head Movement = (100-90) + (90-60) + (60-50) + (50-25) + (25-0) + (199-0) + (199-190) + (190-135) = 363 tracks. The C-Scan disk scheduling algorithm satisfies requests only when the head moves in one direction and not satisfying any requests when it moves back.

Circular Look Disk Scheduling Algorithm (C-LOOK):

Cyclic Look or Circular Look disk scheduling algorithm is an improved version of C-Scan disk scheduling algorithm. Arm goes only as far as the last request in each direction and then reverses direction immediately without first going all the way to the end of the disk [9]**.** Let us consider the same track requests in the disk queue to compute the total head movement of the read/write head and let us again assume that the current read/write head position is at location 100 for a 200 track disk (0-199).

Track Requests: 25, 90, 135, 50, 190 and 60.

In C-Look disk scheduling algorithm, the read/write head initially move from current location 100 to 90, then 90 to 60, then 60 to 50, then 50 to 25 and then to the other direction from 25 to 190 and at last 190 to 135. It services all the requests at either side firstly up to the last request and then moves the other end's last request without fulfilling any requests in the way and then start servicing as shown in the figure below.

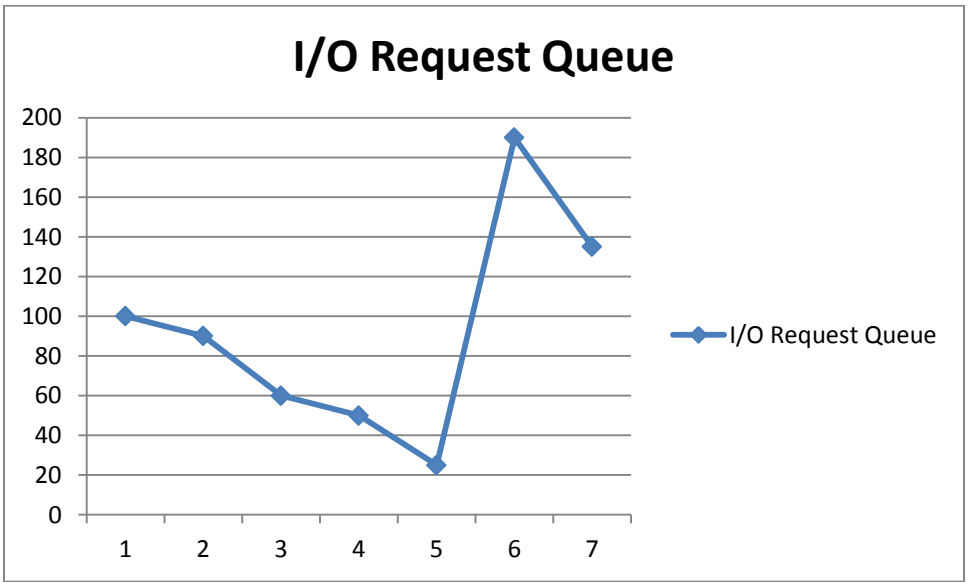

Figure 6: C-LOOK Representation.

.

Total Head Movement = (100-90) + (90-60) + (60-50) + (50-25) + (190-25) + (190-135) = 295 tracks. The major advantage of C-Look disk scheduling algorithm is that it results in higher throughput and lower response time.

III Implementation of disk scheduling algorithms using Turbo C:

The implementation is carried out in Turbo C by creating an interface to find out head movements and total head movement in all the six disk scheduling algorithms. The user has to input the choice for the algorithm and the basic details i.e. total number of requests, initial head position, and the requests for disk queue. For the C-Scan disk scheduling algorithm, additional fields are required for entering the lower limit of cylinders and upper limit of cylinders.

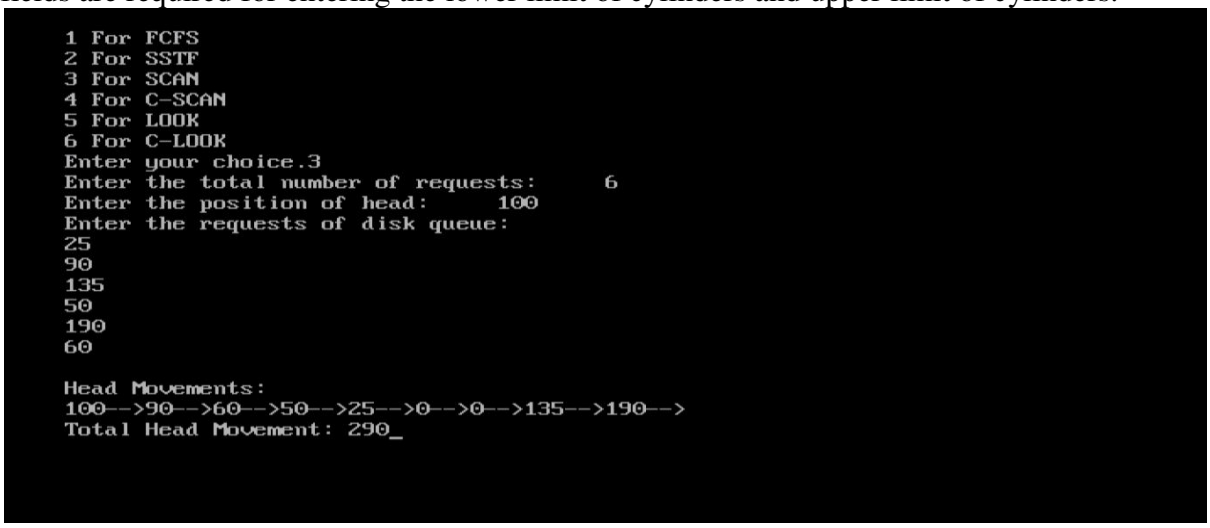

Figure 7: Snapshot of implementation of Disk scheduling algorithms using Turbo C.

IV Comparison of different scheduling algorithms:

Table -1 compares the average head movement of six disk scheduling algorithms for the first five runs and their average. Similar requests are assigned for every individual run for all six algorithms and their total head movement is calculated.

S.No.	FCFS	SSTF	Scan	Look	C-Scan	C-Look
1	540	240	290	240	363	295
2	631	276	280	276	348	306
3	264	217	270	224	393	289
4	322	189	235	189	363	189
5	640	235	275	245	378	300
Average	**479**	**231**	**270**	**235**	**369**	**276**

Table 1: Average head movement of first five runs and their average.

Shortest Seek Time First algorithm produces the minimum head movement of 240 in the first run, 276 in the second run, 217 in the third run, 189 in the fourth run, 235 in the last run and average head movement of all the runs is 231. From the above table, it is very much clear that for the different five runs, the Shortest Seek Time First disk scheduling algorithm has minimum total head movement in all cases as compared to other five algorithms. A graphical representation is shown in the figure 8 with the help of data evaluated in table 1.

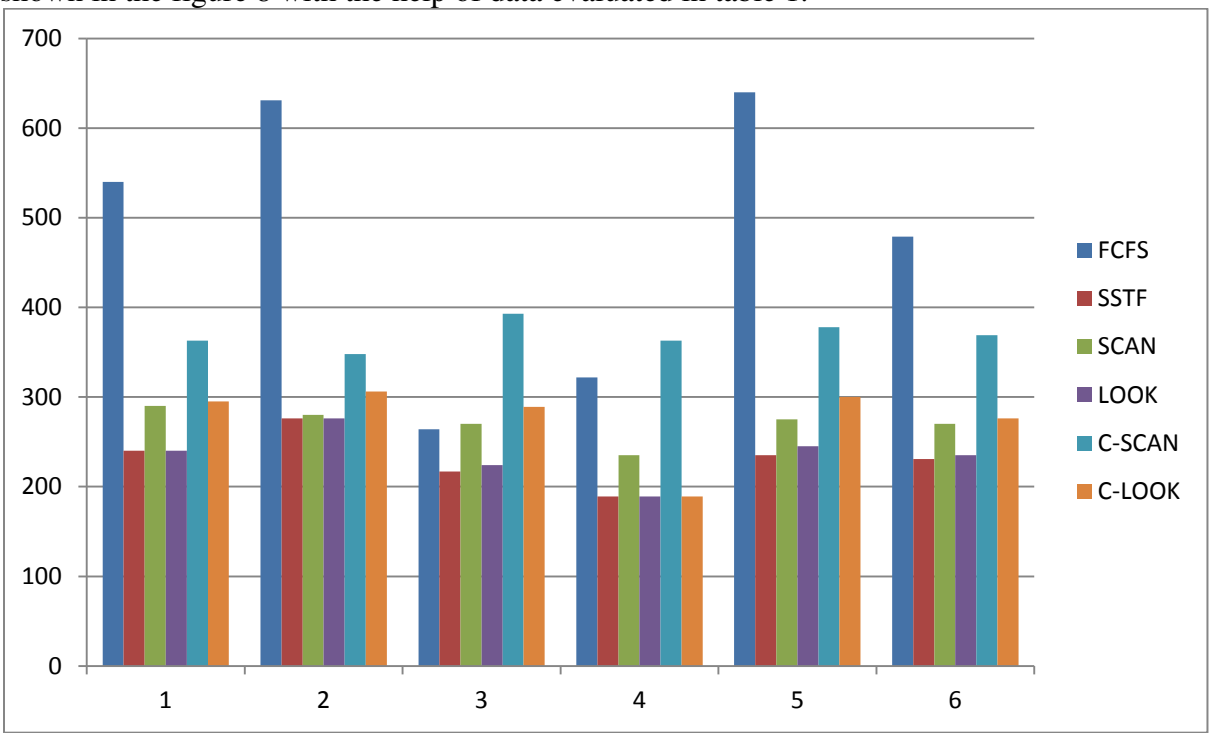

Figure 8: Comparison between six disk scheduling algorithms in graphical representation.

The figure 8 gives comparative details of the six disk scheduling algorithms. The X axis represents five runs and the average case and the Y axis is used for calculating average total head movement of each algorithm. From the various runs, it is concluded that Shortest Seek Time First has the minimum average head movement than all other five algorithms for the similar

requests. Thus Shortest Seek Time First algorithm is a good criterion for selecting the requests for I/O from the disk queue.

V Results and Discussions:

We have calculated the average total head movement after entering the various runs for the requests of different algorithms because total head movement is the criteria for analyzing the disk scheduling algorithms. After comparing the total head movement of various algorithms, we have found that the Shortest Seek Time First disk scheduling algorithm has the least average head movement than the others discussed above in context to total head movement.

VI Conclusion:

The present paper analysis the six disk scheduling algorithms viz. First Come First Serve, Shortest Seek Time First, Scan, Look, C-Scan and C-Look. The implementation was carried out using Turbo C. Different disk scheduling algorithms can be used depending upon the load of the system. The performance depends upon the number and types of requests.

VII Future Work:

In the future, we will use random requests and some other parameters to compare the different disk scheduling algorithms.

References:

[1] William Stallings, Operating Systems: Internals and Design Principles (India: Pearson, 2009).

[2] Abraham Silberschatz, Peter B. Galvin and Greg Gagne, Operating System Concepts, Eight ed., UK: Wiley, 2010.

[3] Andrew S. Tanenbaum, Modern Operating Systems (USA: Prentice-Hall, Inc., 2001).

[4] John Ryan Celis et. al., "A Comprehensive Review for Disk Scheduling Algorithms," International Journal of Computer Science, vol. 11, issue 1, no. 1, pp. 74-79, Jan 2014.

[5] Nidhi and Dayashankar Singh, "A New Optimized Real-Time Disk Scheduling Algorithm," International Journal of Computer Applications, vol. 93, no. 18, pp. 7-12, May 2014.

[6] Hetal Paida, "Disk Scheduling," International Journal of Advanced Research in Computer Science and Management Studies, vol. 1, issue 2, pp. 6-9, July 2013.

[7] Sarita Negi, Kulvinder Singh and Shubam Sahu, "Demand Based Disk Scheduling Using Circular Look Algorithm," International Journal of Computer Science and Mobile Computing, vol. 4, issue 8, pp. 364-368, Aug. 2015.

[8] Priya Hooda and Supriya Raheja, "A New Approach to Disk Scheduling Using Fuzzy Logic," Journal of Computer and Communications, vol. 2, pp. 1-5, 2014.

[9] Sandipon Saha, Md. Nasim Akhter and Mohammod Abul Kashem, "A New Heuristic Disk Scheduling Algorithm," International Journal of Scientific & Technology Research, vol. 2, issue 1, pp. 49-53, Jan. 2013.