Simulation and performance comparison of four disk scheduling algorithms

Conference Paper · February 2000

DOI: 10.1109/TENCON.2000.888379 · Source: IEEE Xplore

2 authors, including:

Muhammad Younus Javed
HITEC University, Taxila, Pakistan
326 PUBLICATIONS **3,248** CITATIONS

SEE PROFILE

Simulation and Performance Comparison of Four Disk Scheduling Algorithms

Dr. Muhammad Younus Javed and Mr. Ihsan Ullah Khan

Computer Engineering Department
College of Electrical and Mechanical Engineering
Peshawar Road, Rawalpindi – 46000
(National University of Sciences and Technology - PAKISTAN)
E-mail : myjaved@yahoo.com
Fax : 051-476190, 474306
Telephone : 051- 478783, 051-561-32966

ABSTRACT

Hard disks are being used to store huge information/data in all modern computers. Disk drives must provide faster access time in order to optimize speed of I/O operations. In multitasking system with many processes, disk performance can be improved by incorporating a scheduling algorithm for maintaining several pending requests in the disk queue. This paper describes development of a simulator which uses four disk scheduling algorithms (FCFS, SSTF, LOOK for both upward and downward direction, and C-LOOK) to measure their performance in terms of total head movement. Five different types of test samples, containing request tracks from 8 to 50, have been used to obtain simulation results. Developed simulator runs successfully in a multiprogramming environment and the tabulated results demonstrate that LOOK (downward direction) algorithm provides the best results for given test samples due to reduction of a large number of unnecessary head movements. As several wild swings are experienced by FCFS scheme so it gives the worst scheduling performance. SSTF is much better compared to LOOK (upward direction) and C-LOOK. It has also been noticed that LOOK is more efficient than C-LOOK al all loads whereas C-LOOK is better at high loads only as it reduces starvation problem. Performance of each algorithm, however, heavily depends on the number and type of requests.

1. DISK SCHEDULING

Since most jobs depend heavily on the disk for loading and I/O operations, it is important that disk service be as fast as possible. Disk speed is composed of three parts: seek time, latency time and transfer time. Every I/O device, including each disk drive, has queue of pending requests. Whenever a process needs I/O to or from the disk, it issues a system call to the operating system. This request specifies several pieces of information: (1) type of I/O operation, (2) address of disk (drive, cylinder, surface, block), (3) address of memory, and (4) amount of information is to be transferred (a byte or word count). If the desired disk drive and controller are available, the request can be serviced immediately. However, while the drive or controller is serving one request, any additional requests will need to be queued. For multiprogramming system with many processes, the disk queue may often be nonempty. Thus, when a request is complete, a new request is picked from the queue and it is serviced. A disk service requires that the head be moved to the desired track, it must wait for latency and seek time, and finally transfer the data to memory. Different algorithms such as FCFS, SSTF, LOOK and C-LOOK are used for selecting request for servicing from the queue of requests. First-Come, First-Served (FCFS) [1-5] is the simplest form of disk scheduling. This algorithm is easy to implement using FIFO queue. Wild swing of FCFS can increase total head movements. Shortest-Seek-Time-First (SSTF) scheme [1-4] seems reasonable to service together all requests close to the current head position, before moving the head far away to service another request. It selects the request with the minimum seek time from the current head position. Since the seek time is generally proportional to the track difference between the requests, it is implemented by moving the head to the closest track in the request queue. Some of the requests may wait indefinitely in this approach. Recognition of the dynamic nature of the request queue leads to the LOOK algorithm [1-4]. The read-write head starts from its present position, services requests while moving towards one end of the disk, after servicing the last request of this end, the direction of the head movement is reversed and servicing continues as far as the last request in this direction. LOOK scheme continuously scans the disk from one direction to the other keeping the swing to the last request on either side. In a real-time system, if a request arrives just in front of the head, it will be serviced almost immediately, whereas a request arriving just behind the head will have to wait until the head moves to the last request of one end of the disk, reverses direction and returns to service the leftover requests on its way to the other end. In practical systems the head is moved as far as the last request in each direction. As soon as there are no requests in the current direction, the head movement is reversed. A variant of LOOK scheduling that is designed to provide a more uniform wait time is C-LOOK (Circular LOOK) scheduling [1,4]. C-LOOK moves the head towards one direction from its

0-7803-6355-8/00/ $10.00 © 2000 IEEE

present position, after servicing the last request in the current direction it reverses its direction and immediately returns to the first request of the other end, without servicing any requests on the return trip. C-LOOK scheduling essentially treats the disk as though it were circular, with the last track adjacent to the first one.

2. SOFTWARE DEVELOPMENT

A simulator has been developed in C language for disk scheduling algorithms which contains four major modules (i.e. FCFS, SSTF, LOOK and C-LOOK). This simulator runs each and presents results based upon service requests and number of tracks involved in the given test sample. Tracks requests are read by the program from the relevant file. Algorithm automatically offers reordered list of the read requests and a queue is displayed to service the requests. When all requests are serviced, then the result in the form of total head movement is displayed. After this the graph is shown on screen to check performance of each disk scheduling algorithm. Each module can be run to get new data of requests from the keyboard. Once the user has completed the input data, the concerned module stores it in a separate file for making analysis of performance of scheduling algorithms. Any number of test samples can be stored or run by the simulator to check behaviour of different algorithms. Total head movement is calculated based upon selected test sample or entered data. Various data structures such as linked lists, arrays, queues and records have been used for successful execution, recording of simulation results and graphical display of comparison results. Simulator offers six options (i.e. FCFS, SSTF, LOOK, C-LOOK, comparison results, and quit) whereas every disk scheduling algorithm has five options (i.e. enter data, read from a file, run. back to main menu, and quit). Simulator quickly rearranges the requests received from a file/keyboard on the basis of sequence needed for each algorithm (e.g. shortest seek time first is provided to the head keeping in view its present position in order to maximize the performance).

3. TEST SAMPLES AND SIMULATION RESULTS

Five test samples were selected such that each sample consists of different type and number of tracks. The number of tracks varies from 8 to 50. Each sample is fed in to the simulator in the form of a separate file and is executed using selected algorithm to obtain total head movements. Test sample 1 consists of 8 tracks (98, 183, 37, 122, 14,124, 65, 67), where track 98 contains the first received request and track 67 has the last received request. The initial head position was supposed to be at track 53. When this sample is executed the total head movement for FCFS is 640 tracks, for SSTF is 236 tracks, and for C-LOOK is 322 tracks. In case of LOOK there are two possibilities (i.e. either the head can first travel upward or downward). Total head movement for upward movement is 299 tracks and for downward movement is 208 tracks. Test sample 2 contains 12 tracks: 2,5,7, 10, 15, 17, 19,23,26,31&35. Track 2 is the first request and track 35 is the last request. Test sample 2 is unique in the sense that all requests are situated below the initial head position (53) as well as all requests received are in the ascending order. When this test sample is executed, the total head movement obtained for FCFS is 84 tracks, for SSTF is 51 tracks and for C-LOOK is 84 tracks. In case of LOOK algorithm for this example, only downward movement is considered as all requests are situated below the initial head position. So the total head movement in case of LOOK is 51 tracks.

There are 24 tracks (i.e. 47, 35, 180, 132, 156, 23, 34, 5, 210, 83, 96, 103, 88, 76, 113, 128, 73, 120, 169, 123, 111, 179, 136& 40) in test sample 3. Track 47 is the first and track 40 is the last request and the initial head position at track 53. On its execution, the total head movement obtained for FCFS is 1255 tracks, for SSTF is 253 tracks and for C-LOOK is 404 tracks. LOOK gives the total head movement of 362 tracks and 253 tracks for upward and downward head movement respectively. 24 tracks (i.e. 77, 5, 11, 175, 25, 38, 1, 20, 250, 17, 211, 15, 37, 285, 149, 42, 19, 32, 2, 31, 6, 34, 48 &125) are associated with test sample 4. Supposing that the requests are received for tracks in the given order and initial position of head is at track 53, the total head movement for FCFS is 2290 tracks, for SSTF is 336 tracks, C-LOOK is 563 tracks, for upward movement in LOOK is 516 and for downward movement is 336. Test sample consists of 50 tracks: 272, 69, 23, 58, 190, 205, 39, 117, 25, 213, 121, 290, 310, 19, 80, 8, 94, 222, 40, 176, 312, 303, 260, 150, 167, 32, 27, 253, 122, 73, 134, 11, 5, 83, 50, 2, 240, 240, 301, 4, 263, 138, 7, 291, 319, 9, 15, 230, 88 & 1. The total head movement obtained for FCFS is 5804 tracks, for SSTF is 590 tracks, for C-LOOK is 633 tracks, for C-LOOK (upward) is 584 tracks and for LOOK (downward) is 370 tracks.

4. DISCUSSION AND COMPARISON OF RESULTS

Once the simulator was successfully developed for each module to constitute a usable system, the next step was to demonstrate that the implemented system executes various disk scheduling algorithms and provides simulation results to check its effectiveness. Five test samples containing tracks ranging from 8 to 50 were used to obtain results. In the FCFS scheduling, the first request to arrive is the first one serviced. FCFS is fair in the sense that once a request has arrived, its place in the queue is fixed. A request can not be displaced because of the arrival of a high priority request. FCFS will actually go for a lengthy seek to service a distant waiting request even though another request may have just arrived near the track where the read-write head is currently positioned. Considering the fist test sample, the read-write head will first move from 53 (current position) to track 98 (first request), then to 183, 37, 122, 14, 124, 65 and finally to track 67 for a total head movement of 640 tracks. It provides the worst performance compared to other algorithms as shown in Table 1(entries 1-5) due to the wild swing from track 122 to track 14and then back to track 124. Had the requests for tracks 37 and 14 been serviced together, before or after the requests at 122 and 124, the total head movement could have been decreased substantially. Performance of FCFS is equivalent to C-

LOOK for sample 2 but its performance is lower compared to SSTF and LOOK as depicted by 6-10 entries of Table 1. The head moves from track 53 to 2, 5, 7, 10, 15, 17, 19, 23, 26, 31 and 35 which results a total head movement of 84 tracks. The problem with this schedule is clear by the wild swing from the initial head position (53) to the first request (2). If the requests are ordered in such a way that request (35) is serviced first then 26,23,19,17,15,10,7 and 5 are serviced and finally request 2 is entertained, the total head movement decreases from 84 to 51 tracks, thus improving disk throughput. FCFS also provides worst results for test samples 3, 4 and 5 due to presence of wild swings. A Large number of wild swings experienced in test samples 5 are shown in Figure 1.

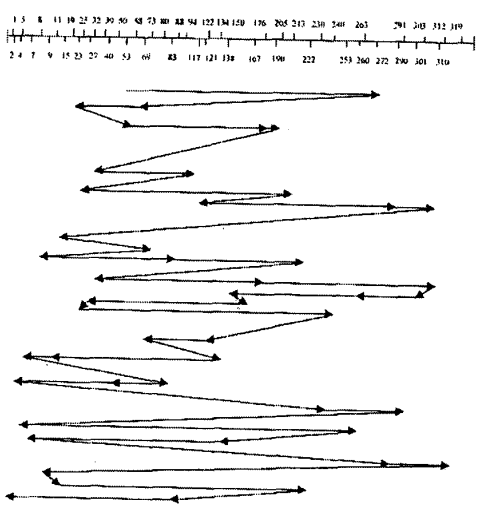

Figure 1 FCFS Disk Scheduling for Test Sample 5

In SSTF algorithm the request that results in the shortest seek distance (and thus the shortest seek time) is serviced first even if that request is not the first one in the queue. Looking at the test sample1, the reordered queue on the basis of SSTF becomes 65, 67, 37, 14, 98, 122, 124 &183 if initial position of the head is at track 53. Track 65 will be serviced first then tracks 67, 37, 14, 98, 122, 124 and finally track 183 will be serviced for a total head movement of 236 tracks. The SSTF algorithm, although a substantial improvement over the FCFS algorithm, is not optimal. For example, if the head is moved from 53 to 37, even though the later is not closest, and then to 14, before turning around to service 65, 67, 98, 122, 124, and 183, the total head movement can be reduced to 208 tracks. Applying SSTF on test sample 2, the re-ordered queue becomes: 35, 31, 29, 26, 23, 19, 17, 15, 10, 7, 5 & 2. Track 35 will be serviced first then 31, 29, 26, 23, 19, 17, 15, 10, 7, 5 and finally track 2 will be serviced. Total head movement is 51 tracks. The beauty of this algorithm is that the swings noticed in case of FCFS algorithm are almost eliminated by SSTF algorithm due to which total head movement is reduced which results in better disk throughput as shown in Table 1. For test sample 3, the reordered queue of SSTF algorithm is: **47, 40,** 35, 34, 23, 5, 73, 76, 83, 88, 96, 103, 111, 113, 120, **123,** 128, 132, 136, 156, 169, 179, 180 and 210. Majority of wild swings have been eliminated except one that is the swing from track 5 to track 73. Resultantly, the total head movement has been decreased substantially as compared to FCFS, LOOK and C-LOOK algorithms. Similar arguments apply for excellent performance of SSTF algorithm for test samples 4 and 5. The smooth curve of Figure 2, obtained for test sample 5, is a manifestation of the fact that this algorithm does not compel the read-write head to move unnecessarily, thus saving precious head movements and time.

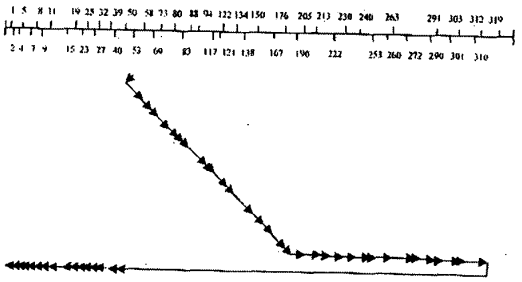

Figure 2 SSTF Disk Scheduling for Test Sample 5

LOOK algorithm operates like SSTF except that it chooses the requests that results in the shortest seek distance in a preferred direction. The read-write head starts movement in the preferred direction (upward or downward), and moves towards the other end of the disk servicing requests as it reaches each track, until there are no further requests pending. At the other end, the direction of head movement is reversed and servicing continues till the availability of requests in this direction. The head continuously scans the disk from end to end. If the initial head position is at track 53 and the head movement is upward for test sample 1 then the reordered queue on the basis of LOOK algorithm becomes: 65, 67, 98, 122, 124, 183, 37 &14. Track 65 will be serviced first and track 14 will be serviced in the end, for a total head movement of 299 tracks. All swings except one (i.e. from track 183 to 37) have been eliminated after the application of LOOK algorithm. Due to this reason LOOK (upward direction) is better than FCFS and C-LOOK for test sample 1. LOOK (downward direction) rearranges queue as: 37, 14, 65, 67, 98, 122, 124 & 183. This reduces the swing size. As a result, all given tracks of test sample 1 are serviced for a total head movement of 208 tracks which provides the best performance as compared to results for FCFS, SSTF, LOOK (upward direction) and C-LOOK algorithms given in Table 1. Reordered queue for test sample 2 becomes: 35, 31, 29, 26, 23, 19, 17, 15, 10, 7, 5, & 2 for LOOK (downward direction). Total head movement of 51 tracks. LOOK (upward direction) is applicable in this example as all tracks are below track 53 (initial head position). In this case its performance is similar to SSTF, which is better than FCFS and C-LOOK. Table results indicate that LOOK (downward direction) gives a

total head movement of 253 tracks (similar to SSTF), which is far better than FCFS (1255 tracks), LOOK upward direction (362 tracks) and C-LOOK (404 tracks). Major cause of improved results is that it saves a lot of head movements due to efficient rearrangement of tracks before its execution. LOOK (upward direction) in this case is more efficient than FCFS and C-LOOK. Results of test samples 4 and 5 also show that LOOK (downward direction) provides best results (i.e. 336 and 370 tracks for test samples 4 and 5 respectively) due to elimination of swings whereas performance of LOOK (upward direction remains better than FCFS and C-LOOK. Figures 3 and 4 show that all swings have been removed as observed in Figure 1 by using LOOK (downward direction) and LOOK (upward direction) respectively. There is only one minor swing (from track 319 to 50) in Figure 3 and a swing (from track 1 to 58) in Figure 4.

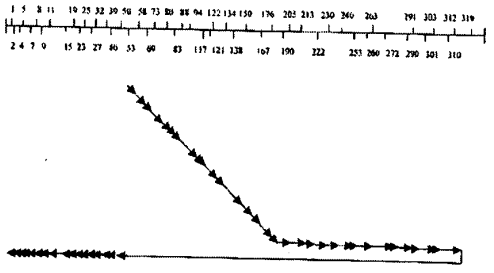

Figure 3 LOOK (downward direction) Disk Scheduling for Test Sample 5

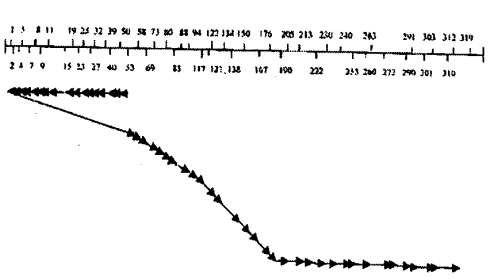

Figure 4 LOOK (upward direction) Disk Scheduling for Test Sample 5

In C-LOOK scheduling the head is moved from its present position (moving towards the upward direction) to one end of the disk, reverses its direction after servicing the last track in upward direction and goes to the first request to the other direction (without servicing any requests on its way). Starting with test sample 1, order of the rearranged queue becomes: 65, 67, 98, 122, 124, 183, 14 & 37. Track 65 will be serviced first then it will service 67, 98, 122, 124 and 183 in upward direction starting from its initial position at track 53. After servicing track 183, as there is no request pending on this side so the head will reverse its movement and will not service request 37 on the return trip rather it goes straight to track 14 which is the pending request nearest to start of the disk. In the end, it will service track 37. The total head movement for this example is 322 tracks which is better than FCFS (640 tracks) but it is worse than SSTF (236 tracks), LOOK upward direction (299 tracks) and LOOK downward direction (208 tracks). Its large swing from track 183 to track 14 is the major cause of poor performance. For the test sample 2, reordered queue (2, 5, 7, 10, 15, 17, 19, 23, 26, 29, 35) will provide a head movement of 84 tracks. Its performance in this case is worst like FCFS due to a swing of 51 tracks (from track 53 to track 2). This suggests that the initial head position should be at the first track of the disk to obtain better results. After application of C-LOOK on test sample 3, the reordered queue becomes: 73, 76, 83, 88, 96, 103, 111, 113, 120, 123, 128, 132, 136, 156, 169, 179, 180, 210, 5, 23, 34, 35, 40, & 47. There is a swing of 205 tracks (from track 210 to track 5) which reduces its overall performance to a total head movement of 404 tracks as shown in Table 1. Due to this reason, this scheduling algorithm gives bad results compared to SSTF and LOOK (both upward and downward direction). Of course its extra head movements are substantially lesser as compared to FCFS. Test samples 4 and 5 give a total head movement of 563 tracks and 633 tracks using C-LOOK algorithm. Again, its performance remains lower than SSTF and LOOK in both cases because of a very large swing (track 285 to track 1 in test sample 4 and form track 319 to 1 for test sample 5). Figure 5 shows the order in which requests are serviced using C-LOOK for test sample 5.

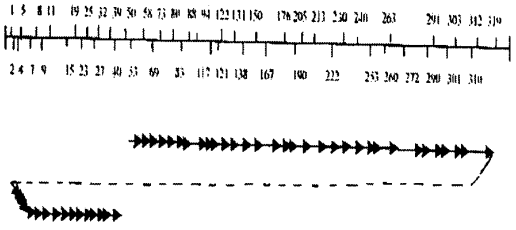

Figure 5 C-LOOK Disk Scheduling for Test Sample 5

From Table1, it transpires that FCFS algorithm has the largest head movement for all five examples. FCFS can be termed as the least efficient algorithm. However, this algorithm is easy to program and is intrinsically fair as it does not change the position of a request in the queue. SSTF algorithm can be termed as the best algorithm as far as the criteria of total head movement is concerned. LOOK (downward direction) is even better than SSTF for test samples 1 and 5. It is important to note that total head movement in LOOK algorithm depends on the preferred head movement direction and if the actual head movement is different than the supposed one then it will change the result. However the drawback of SSTF algorithm is that, in a real system, SSTF algorithm may cause starvation of some requests. To clear this point further let us assume that

Table 1 Head Movement Results of Disk Scheduling Algorithms

S/No	Total Tracks	Algorithm	Initial Head Position	Head Movement Direction	Total Head Movement (in tracks)
1.	8	FCFS	53	N.A	640
2.	8	SSTF	53	N A	236
3.	8	LOOK	53	U/WARD	299
4.	8	LOOK	53	D/WARD	208
5.	8	C-LOOK	53	N.A	322
6.	12	FCFS	53	N.A	84
7.	12	SSTF	53	N A	51
8.	12	LOOK	53	D/WARD	51
9.	12	C-LOOK	53	N.A	84
10.	24	FCFS	53	N.A	1255
11.	24	SSTF	53	N A	253
12.	24	LOOK	53	U/WARD	362
13.	24	LOOK	53	D/WARD	253
14.	24	C-LOOK	53	N.A	404
15.	24	FCFS	53	N.A	2290
16.	24	SSTF	53	N A	336
17.	24	LOOK	53	U/WARD	516
18.	24	LOOK	53	D/WARD	336
19.	24	C-LOOK	53	N.A	563
20.	50	FCFS	53	N.A	5804
21.	50	SSTF	53	N A	590
22.	50	LOOK	53	U/WARD	584
23.	50	LOOK	53	D/WARD	370
24.	50	C-LOOK	53	N.A	633

we have two requests in the queue, for 14 and 186. If a request near 14 arrives while we are servicing that request, it will be serviced next, making the request at 186 to wait. While this request is being serviced, another request close to 14 could arrive. In theory, a continual stream of requests near one another could arrive causing the request for track 186 to wait indefinitely. From table 1 we note that, for all examples, the total head movement in case of LOOK is lesser than FCFS and C-LOOK algorithms. However when we compare LOOK with SSTF, it transpires that in some cases LOOK results in lesser head movements than SSTF while in some cases the results of both are same. Drawback of the LOOK is that its results depend on the direction of head movement. Furthermore, if a request arrives in the queue just in front of the head, it will be serviced almost immediately, whereas a request arriving just behind the head will have to wait until the head moves to the end of the disk, reverses direction, and returns, before being serviced. Consider C-LOOK algorithm, it can be noted that total head movement in C-LOOK is lesser than FCFS and more than SSTF algorithm. The difference in total head movement of LOOK and C-LOOK algorithms is substantial in case of LOOK (downward direction) but in case of LOOK (upward direction) this difference is little.

5. CONCLUSION

The actual performance of disk I/O depends on computer architecture, operating system, nature of the I/O channel, disk structure and disk scheduling algorithms. On a movable-head system, total delay in an I/O operation contains seek, latency and transfer time. Major portion of delay is caused by the seek time and it must be minimized in order to enhance performance of disk scheduling. An efficient disk service requires that the head be moved as quickly as possible to the required track to perform a read/write operation. A simulator comprising of four disk scheduling algorithms (FCFS, SSTF, LOOK and C-LLOK) has been implemented in C language to reduce the seek time in this research work. Each of these algorithms is used for selecting a request for servicing from the queue of requests. The queue of requests is maintained in an appropriate order based upon the scheduling algorithm. Five test samples containing requests for varying tracks (8 to 50) were used to check performance of the simulator using four algorithms. Initial head position for each case was track 53. Head movements of each algorithm has been illustrated using test samples and figures. Tabulated simulation results indicated that FCFS provides the worst performance due to presence of wild swings. FCFS goes for a lengthy seek to service a distant waiting request even though another request may have just arrived near the track where the read/write head is currently positioned. Its total head movement is 640, 84, 1255, 2290 and 5804 tracks for test samples 1, 2, 3, 4 and 5 respectively. In SSTF algorithm, a substantial improvement has been observed

due to elimination of large swings. It reduces the head movements and provides results of 236, 51, 253, 336 and 590 tracks for five test samples. LOOK algorithm works like SSTF except that it uses either upward or downward direction to choose the requests from the rearranged queue. Performance of this algorithm was 299 (sample 1), 362 (sample 3), 516 (sample 4) and 584 (sample 5) using upward direction whereas it was 208 (sample 1), 51 (sample 2), 253 (sample 3), 336 (sample 4) and 370 (sample 5) using downward direction. Thus, it demonstrated the best results for head movement in down direction and very good results for head movement in upward direction. Results obtained for C-LOOK were 322, 84, 404, 563 and 633 for test samples 1 to 5. Its performance remains worse than LOOK and SSTF in most of the cases and provides better results than FCFS. C-LOOK is found more efficient at high loads only. These results are extremely useful for designing more efficient and effective disk scheduling algorithms to reduce seek time in multiprogramming environment.

6. REFERENCES

[1] Silberschatz, A. and Galvin, P.B., *Operating System Concepts*, 5th Edition, Addison-Wesley Publishing Company, 1998.

[2] Tanenbaum, A.S. and Woodhull, A.S., *Operating Systems*, 2nd Edition, Prentice-Hall, 1998.

[3] Tanenbaum, A.S., *Modern Operating Systems*, 2nd Edition, Prentice-Hall, 1996.

[4] Stallings, W., *Operating Systems*, Macmillan Publishing Company, 1992.

[5] Hanson, B., *Operating System Principles*, Prentice-Hall, 1992.