EFFICIENT ROUND ROBIN CPU SCHEDULING ALGORITHM FOR OPERATING SYSTEMS

M.Ramakrishna
Assistant Professor, *Department of CSE*
Srinivasa Institute of Engg. and Technology,
JNTU Kakinada,India

G.Pattabhi Rama Rao
Student, *Department of CSE*
Srinivasa Institute of Engg. and Technology,
JNTU Kakinada,India

Abstract: **The main objective of this paper is to develop a new approach for round robin C P U scheduling algorithm which improves the performance of CPU in real time operating system. The proposed Priority based Round-Robin CPU Scheduling algorithm is based on the integration of round-robin and priority scheduling algorithm. It retains the advantage of round robin in reducing starvation and also integrates the advantage of priority scheduling. The proposed algorithm also implements the concept of aging by assigning new priorities to the processes. Existing round robin CPU scheduling algorithm cannot be implemented in real time operating system due to their high context switch rates, large waiting time, large response time, large turnaround time and less throughput. The proposed algorithm improves all the drawbacks of round robin CP U scheduling algorithm. The paper also presents the comparative analysis of proposed algorithm with existing round robin scheduling algorithm on the basis of varying time quantum, average waiting time, average turnaround time and number of context switches.**

Keywords – CPU scheduling, Round Robin CPU scheduling algorithm, Turnaround time, Waiting time, Response time,Context switching, Gantt chart.

I. INTRODUCTION

In computer science, scheduling is the process by which processes are given access to system resources (e.g. Processor cycles, communications bandwidth). The need for a scheduling algorithm [5] arises from the requirement of fast computer systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously).

Scheduling is a fundamental operating system function that determines which process run, when there are multiple runnable processes. CPU scheduling is important because it impacts resource utilization and other performance parameters. There exists a number of CPU scheduling algorithms [1, 2] like First Come First Serve, Shortest Job First Scheduling, Round Robin scheduling, Priority Scheduling etc, but due to a number of disadvantages these are rarely used in real time operating systems except Round Robin scheduling.

A number of assumptions are considered in CPU scheduling which are as follows [19, 20]:

1. Job pool consists of runnable processes waiting for the CPU.
2. All processes are independent and compete for resources.
3. The job of the scheduler is to distribute the limited resources of CPU to the different processes fairly and in a way that optimizes some performance criteria.

The scheduler [6] is the component of the kernel that selects which process to run next. Operating systems may feature up to three distinct types of schedulers, a long term scheduler, a mid-term or medium term scheduler and a short-term scheduler (fig1). The long term scheduler or job scheduler selects processes from the job pool and loads them into memory for execution. The short term scheduler, or CPU scheduler selects from among the processes that are ready to execute, and allocates CPU to one of them. The medium term scheduler removes processes from memory and reduces the degree of multiprogramming results in the scheme of swapping. Swapping is the scheme which is performed by dispatcher which is the module that gives control of the CPU to the process selected by the short-term scheduler [7].

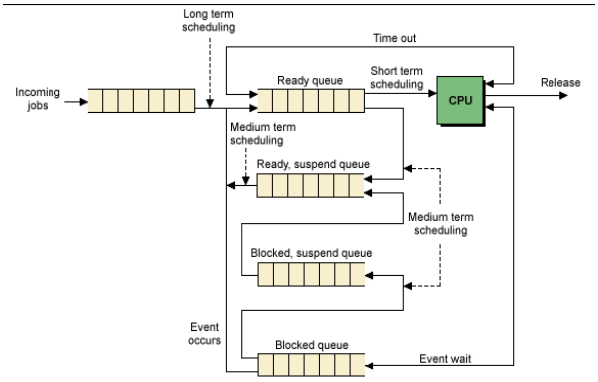


Fig. 1 Queuing diagram for scheduling

The CPU scheduling also plays an important role in the real time operating system which always has a time constraint on computations. A real time system is the one whose applications are mission-critical, where real-time tasks should be scheduled to be completed before their deadlines [8, 9]. Most real-time systems control unpredictable environments and may need operating systems that can handle unknown and changing tasks. So, not only a dynamic task scheduling is required, but both system hardware and software must adapt to unforeseen configurations [10].

There are two main types of real-time systems [23]: Hard Real-Time System, Firm or Soft Real-Time System. In Hard Real-Time System, it requires that fixed deadlines must be met otherwise disastrous situation may arise whereas in Soft Real-Time System, missing an occasional deadline is undesirable, but nevertheless tolerable. System in which performance is degraded but not destroyed by failure to meet response time constraints is called soft real time systems. In real time systems each task should be invoked after the ready time and must be completed before its deadline [12,13, 14], an attempt has been made to satisfy these constraints. Simple round robin architecture [11] is not suitable to implement in Soft real time due to more number of context switches, longer waiting and response times. This in turn leads to low throughput in the system. If a real-time process having relatively larger CPU burst it will leads to the problem of starvation [21]. Priority scheduling may be a better option for real-time scheduling but it will face the similar problem i.e. low priority processes will always starved [22].

II. SCHEDULING OBJECTIVES

A system designer must consider a variety of factors in designing a scheduling algorithm, such as type of systems us ed and what are user's needs. Depending on the system, the user and designer might expect the scheduler to [3]:

1. Maximize throughput: A scheduling algorithm should be capable of servicing the maximum number of processes per unit of time.
2. Avoid indefinite blocking or starvation: A process should not wait for unbounded time before or while process service.
3. Minimize overhead: Overhead causes wastage of resources. But when we use system resources effectively, then overall system performance improves greatly.
4. Enforcement of priorities: if system assigns priorities to processes, the scheduling mechanism should favor the higher-priority processes.
5. Achieve balance between response and utilization: The scheduling mechanism should keep resources of system busy.
6. Favor processes exhibits desirable behavior.
7. Degrade gracefully under heavy load.

A system can accomplish these goals in several ways. The scheduler can prevent indefinite blocking of processes through the concept of aging. The scheduler can increase throughput by favoring processes whose requests can be satisfied quickly, or whose completion cause other processes to run.

III. SCHEDULING CRITERIA

There are various CPU scheduling algorithms which have different properties, and the choice of a particular algorithm may favor one class of processes over another. For selection of an algorithm for a particular situation, we must consider properties of various algorithms. The scheduling criteria [2] include the following:

1 Context Switch: A context switch is process of storing and restoring context (state) of a preempted process, so that execution can be resumed from same point at a later time. Context switching is usually computationally intensive, lead to wastage of time and memory, which in turn increases the overhead of scheduler, so the design of operating system is to optimize only these switches.

2 Throughput: Throughput is defined as number of processes completed per unit time. Throughput is slow in round robin scheduling implementation. Context switching and throughput are inversely proportional to each other.

3 CPU Utilization: This is a measure of how much busy the CPU is. Usually, the goal is to maximize the CPU utilization.

4 Turnaround Time: Turnaround time refers to the total time which is spend to complete the process and is how long it takes the time to execute that process. The time interval from the time of submission of a process to the time of completion is the turnaround time. Total turnaround time is the sum of the periods spent waiting to get into memory, waiting time in the ready queue, execution time on the CPU and doing I/O.

5 Waiting Time: Waiting time is the total time a process has been waiting in ready queue. The CPU scheduling algorithm does not affect the amount of time during which a process executes or does input-output; it affects only the amount of time that a process spends waiting in ready queue.

6 Response Time: In an interactive system, turnaround time may not be best measure. Often, a process can produce some output fairly early and can continue computing new results while previous results are being produced to the user. Thus, response time is the time from the submission of a request until the first response is produced that means time

when the task is submitted until the first response is received. So the response time should be low for best scheduling.

So we can conclude that a good scheduling algorithm for real time and time sharing system must possess following characteristics [3]:

- Minimum context switches.
- Maximum CPU utilization.
- Maximum throughput.
- Minimum turnaround time.
- Minimum waiting time.
- Minimum response time.

IV. ROUND ROBIN SCHEDULING ALGORITHM

The RR scheduling algorithm [4] is given by following steps:-

1. The scheduler maintains a queue of ready processes and a list of blocked and swapped out processes.

2. The Process Control Block of newly created process is added to end of ready queue. The Process Control Block of terminating process is removed from the scheduling data structures.

3. The scheduler always selects the Process Control Block from the head of the ready queue. This is a disadvantage since all processes are basically given the same priority. Round robin also favors the process with short CPU burst and penalizes long ones [17].

4. When a running process finishes its time slice, it is moved to end of ready queue. A time slice [16] is an amount of time that each process spends on the processor per iteration of the Round Robin algorithm. All processes are executed in a first come first serve manner but are preempted after a time slice. The process will either finish in the time slice given or the process will be returned to the tail of the ready queue and return to the processor at a later time.

5. The event handler performs the following actions:

a) When a process makes an input -output request or swapped out, its Process Control Block is removed from ready queue to blocked/swapped out list.

b) When input-output operation awaited by a process finishes or process is swapped in its Process Control Block is removed from blocked/swapped list to end of ready queue.

There are some disadvantages of round robin CPU scheduling algorithm for operating system which are as follows:

- ***Larger waiting time and Response time***

In round robin architecture the time which process spends in the ready queue waiting for the processor to get executed is known as waiting time and the time when the process takes to complete its job and exit from the task is called as turnaround time. Larger waiting and response time are clearly a drawback in round robin architecture as it leads to degradation of system performance.

- ***Context Switches***

When the time slice of the task ends and the task is still executing on the processor the scheduler forcibly preempts the tasks on the processor and stores the task context in stack or registers and allocates the processor to the next task in the ready queue. This action which is performed by the scheduler is called as context switch. Context switch leads to the wastage of time, memory and leads to scheduler overhead.

- ***Low throughput***

Throughput is defined as number of process completed per time unit. If round robin is implemented in soft real time systems throughput will be low which leads to severe degradation of system performance. If the number of context switches is low then the throughput will be high. Context switch and throughput are inversely proportional to each other. With these observations it is found that the existing simple round robin architecture is not suitable for real time systems. So, its drawbacks are eliminated in the modified version of round robin described in the next section.

V. PRIORITY SCHEDULING ALGORITHM

The operating system assigns a fixed priority to every process, and the scheduler arranges the processes in the ready queue in order of their priority. Lower priority processes get interrupted by incoming higher priority processes. Overhead is not minimal, nor is it significant in this case. Waiting time and response time depend on the priority of the process. Higher priority processes have smaller waiting and response times. Deadlines can be easily met by giving higher priority to the earlier deadline processes.

Disadvantage: Starvation of lower priority processes is possible if large no of higher priority processes keep arriving continuously.

VI. PROPOSED ALGORITHM

The proposed architecture focuses on the drawbacks of simple round robin architecture which gives equal priority to all the processes (processes are scheduled in first come first serve manner). Because of this drawback round robin architecture is not efficient for processes with smaller CPU burst. This results in the increase in waiting time and response time of processes which results in the decrease in the system throughput.

The proposed architecture eliminates the defects of implementing simple round robin architecture. The proposed algorithm will be executed in two steps which will helps to minimize a number of performance parameters such as context switches, average waiting time and average turnaround time.

The algorithm performs following steps:

Step 1: Allocate CPU to every process in round robin fashion, according to the given priority, for given time quantum (say k units) only for one time.

Step 2: After completion of first step following steps are performed:

a) Processors are arranged in increasing order or their remaining CPU burst time in the ready queue. New priorities are assigned according to the remaining CPU bursts of processes; the process with shortest remaining CPU burst is assigned with highest priority.

b) The processes are executed according to the new priorities based on the remaining CPU bursts, and each process gets the control of the CPU until they finished their execution.

VII. CASE STUDIES

Five processes have been defined with CPU burst time and their priorities, these five processes are scheduled in round robin fashion and also according to the proposed algorithm. The context switch, average waiting time, average turnaround time has been calculated and the results were compared. For doing this we have implemented the priority based CPU scheduling algorithm in C and carried out number of experiments out of which only two experiments are discussed here for varying time quantum and we assure that the results analysis are remain unchanged for others**.**

Case I:

Consider five processes viz. A, B, C,D and E with given CPU burst time and associated priorities.
Let the time quantum is **5 ms.**

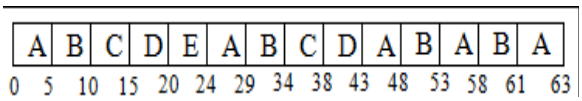
Process Name	CPU Burst Time (ms)	Priority
A	22	4
B	18	2
C	9	1
D	10	3
E	4	5

Table1. Input component for the processors

According to simple RR scheduling:

Simple Round Robin does not use priority and five processes has been scheduled using simple Round Robin architecture. The time slice of five milliseconds has been used. In round robin algorithm no process is allocated CPU for more than one time slice in a row. If the CPU process exceeds one time slice, the concern process will be preempted and put into the ready queue. The process is preempted after the first time quantum and the CPU is given to the next process which is in the ready queue (process B), similarly schedules all the process and completes the first cycle. In the second cycle same method is used to schedule the processes. The process time slicing in simple Round Robin architecture is shown in Gantt chart.

Gantt chart:



Number of context switches: 13
Average waiting Time: 33.200001 ms
Average Turnaround Time: 45.8 ms

According to proposed algorithm:

Priority based Round Robin CPU scheduling consists of two rounds:

Round 1: Process with the highest priority is executed first for the time equal to given time quantum i.e. 5 ms. In the same manner other processes are executed according to their priorities for single time quantum. Eg: The sequence of execution for above case is:

Table 2. Executed CPU burst for first round

S.No	Process	Executed Burst	Priority
1	C	5	1
2	B	5	2
3	D	5	3
4	A	5	4
5	E	5	5

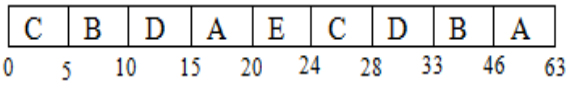
Round 2: This round includes the changing of process's priorities according to the remaining CPU Burst Time. The process with least remaining CPU Burst Time is assigned highest priority. The new assigned priorities are as follows:

Table 3. Remaining CPU burst for second round & new assigned priorities

S.No	Process	Remaining Burst	Priority
1	C	4	1
2	D	5	2
3	B	13	3
4	A	17	4

Now the processes are executed according to the new priority assigned without taking consideration of time quantum.

Gantt chart:



Number of context switches: 8
Average waiting Time: 26.200001 ms
Average Turnaround Time: 38.800000 ms

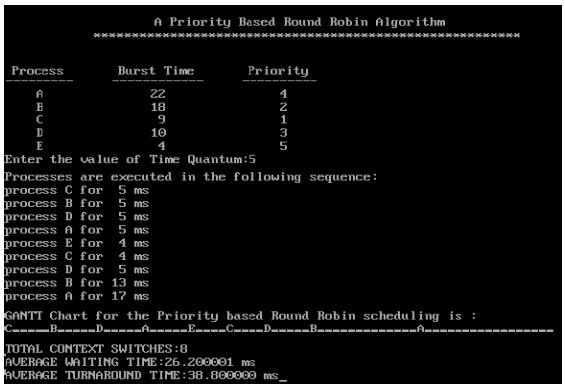


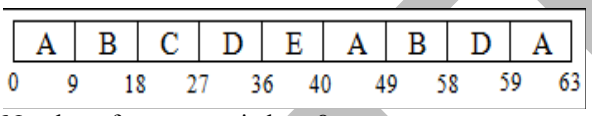
Fig. 2 Priority based round robin for time quantum=5

Case II:
Consider the same problem with varying time quantum.
Let the time quantum is 9 ms.

According to simple RR scheduling:-

Execution of processes takes place without considering priorities.

Gantt chart:



Number of context switches: 8
Average waiting Time: 38.2 ms
Average Turnaround Time: 50.8 ms

According to proposed algorithm:

Execution takes place in two rounds:
Round 1: Process with highest priority is executed first for the time equal to given time quantum i.e. 9 ms. In the same manner other processes are executed according to their priorities for single time quantum.
Ex: The sequence of execution for above case is:

Table 4. Executed CPU burst for first round

S.No	Process	Executed Burst	Priority
1	C	9	1
2	B	9	2
3	D	9	3
4	A	9	4
5	E	4	5

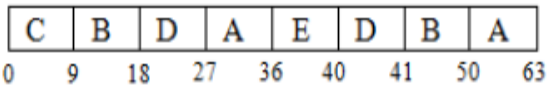
Round 2: This round includes the changing of process's priorities according to the remaining CPU Burst Time. The process with least remaining CPU Burst Time i.e. D (1 ms) is assigned highest priority. The new assigned priorities are as follows:

Table 5. Remaining CPU burst for second round & new assigned priorities

S.No	Process	Remaining Burst	Priority
1	D	1	1
2	B	9	2
3	A	13	3

In the second round the processes are executed until they finished their execution, according to the new priority assigned without taking consideration of time quantum

Gantt chart:



Number of context switches: 7
Average waiting Time: 28.000000 ms
Average Turnaround Time: 40.600000 ms

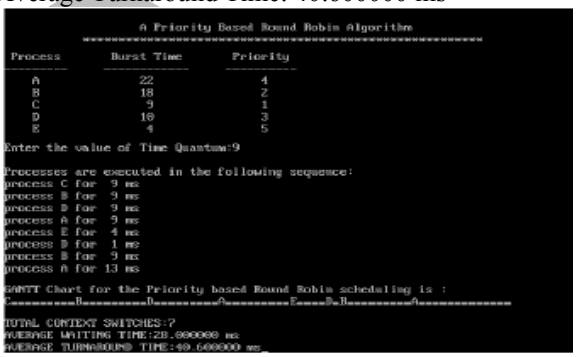


Fig. 3 Priority based round robin for time quantum=9

VIII. COMPARISON OF RESULTS AND DISCUSSION

The performance of two algorithms can be compared by considering the number of context switches, average waiting time and average turnaround time. Fig.4 shows the comparison of number of context switches performed in simple round robin and priority based round robin algorithm and can be plotted in MATLAB 7.0. It shows that the proposed algorithm performs better over simple round robin for varying time quantum. We see that priority based round robin has less number of context switches in comparison to simple round robin for same value of time quantum.

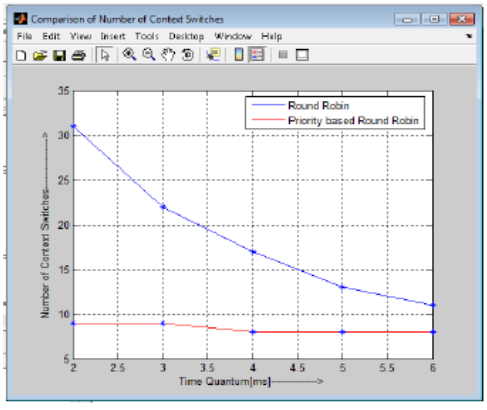


Fig. 4 Graph showing comparison of context switches in both algorithms

Fig.5 shows the comparison of average waiting time in simple round robin and priority based round robin algorithm and can be plotted in MATLAB 7.0. It shows that the proposed algorithm has less average waiting time over simple round robin for varying time quantum.

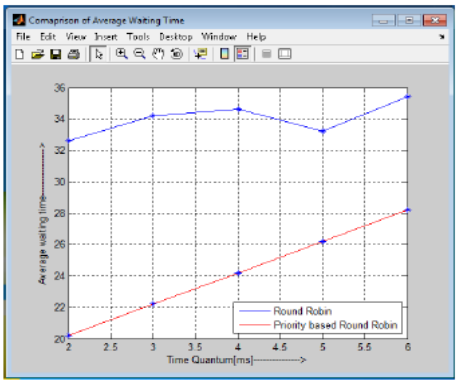


Fig.5 Graph showing comparison of average waiting time in both algorithms

Fig.6 shows the comparison of average turnaround time in simple round robin and priority based round robin algorithm and can be plotted in MATLAB 7.0. It shows that the proposed algorithm has less average turnaround time over simple round robin for varying time quantum.

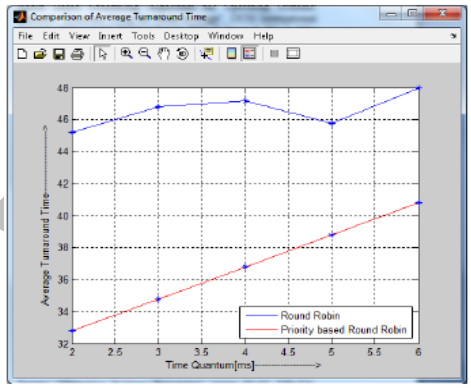


Fig. 6 Graph showing comparison of average turnaround time in both algorithms

IX. CONCLUSION

We have successfully compared both the algorithm i.e. simple round robin and the proposed one that the proposed one is more efficient because it has less average w a i t i n g time, average turnaround time and number of context switches as compared to simple round robin, in turn reducing the operating system overhead and hence dispatch latency. Also, it reduces the problem of starvation as the processes with less remaining CPU burst time are assigned with the higher priorities and are executed first in the second round of algorithm. Performance of time sharing systems can be improved with the proposed algorithm and can also be modified to enhance the performance of real time system.

REFERENCES

[1] Silberschatz, A., Peterson, J. L., and Galvin, B.,*Operating System Concepts,* Addison Wesley, 7th Edition, 2006.

[2] E.O. Oyetunji, A. E. Oluleye," Performance Assessment of Some CPU Scheduling Algorithms", Research Journal of Information Technology, *1(1): pp 22-26, 2009.*

[3] Ajit Singh, Priyanka Goyal, Sahil Batra," An Optimized Round Robin Scheduling Algorithm for CPU Scheduling", (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 07, 2383-2385, 2010.

[4] Rakesh kumar yadav, Abhishek K Mishra, Navin Prakash, Himanshu Sharma," An Improved Round Robin Scheduling Algorithm for CPUScheduling", (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 04, 1064-1066, 2010.

[5] William Stallings, Operating Systems Internal and Design Principles, 5th Edition , 2006.

[6] Saroj Hiranwal, K. C. Roy," Adaptive Round Robin Scheduling using shortest burst approach based on smart time slice", International Journal of Computer Science and Communication Vol. 2, No. 2, pp. 319-323, July-December 2011.

[7] Abbas Noon, Ali Kalakech, Seifedine Kadry, "A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 1, May 2011.

[8] Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., and Weglarz, J.: Scheduling Computer and Manufacturing Processes, Berlin, Springer, (2001).

[9] Stallings, W.: Operating Systems Internals and Design Principles, 5th edition, Prentice Hall, (2004).

[10] Swin, B. R., Tayli, M., and Benmaiza, M.: Prospects for Predictable Dynamic Scheduling in RTDOS, Journal King Saud University, Computer & Information Science, Vol. 9, pp. 57-93, (1997).

[11] Barbara Korousic –Seljak (1994) "Task scheduling policies for real-time systems" Journal on MICROPROCESSOR AND MICROSYSTEMS, VOL 18, NO. 9, pg 501-512.

[12] Enrico Bini and Giorgio C. Buttazzo (2004) "Schedulability Analysis of Periodic Fixed Priority Systems" journal on IEEE

TRANSACTIONS ON COMPUTERS VOL 53 NO.11, pg 1462-1473.

[13] Zhi Quan and Jong-Moon Chang (2003) " A Statistical Framework for EDF Scheduling" journal on IEEE COMMUNICATION LETTERS VOL 7 NO.10, pg 493-495.

[14] Houssine Chetto and Maryline Chetto (1989) " Some Results of Earliest Deadline Scheduling Algorithm" journal on IEEE TRANSACTION ON SOFTWARE ENGINEERING. VOL 15. NO.10, pg 1261-1269.

[15] Harry Katzan, Jr., "Operating Systems / A Pragmatic Approach", pages 113, 157, 350-353, Van Nostrand Reinhold Company, 1973.

[16] Stanley A.Kurzban, Thomas S. Heines, and Anthony P Sayers, "Operating Systems Principles", pages 50-52, 370-371, Van Nostrand Reinhold Company, 1984.

[17] Philipe A. Jansen, "Operating Systems / Structures and Mechanisms" pages 77-80, Academic Press, Inc., 1985.

[18] William Stallings, Ph.D., "Operating Systems / Internals and Design Principles", pages 75-76, 406-408, Prentice Hall, 2001.

[19] Englander, I., 2003. The Architecture of Computer Hardware and Systems Software; An Information Technology Approach, 3rd Edition, John Wiley & Sons, Inc.

[20] Yavatkar, R. and K . Lakshman,1995. A CPU Scheduling Algorithm for Continuous M edia Applications, In Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video, pp: 210-213.

[21] I. E. W. Giering and T. P. Baker, "A tool for the deterministic scheduling of real-time programs implemented as periodic Ada tasks," Ada Lett., vol. XIV, pp. 54-73, (1994).

[22] Shahzad, B., Afzal, M.T.: ,"Optimized Solution to Shortest Job First by Eliminating the Starvation". In: The 6th Jordanian Inr. Electrical an Jordan (2006).

[23] M.Kaladevi, M.C.A.,M.Phil., and Dr.S.Sathiyabama, M.Sc.,M.Phil.,Ph.D, "A Comparative Study of Scheduling Algorithms for Real Time Task", International Journal of Advances in Science and Technology, Vol. 1, No. 4, 2010.

