# Robust Quantum Based Low-power Switching Technique to improve System Performance

M. Lavanya[1], S. Saravanan[2],

[1,2]Assistant Professor, SASTRA University
Thanjavur, India
m_lavanyass@ict.sastra.edu

**ABSTRACT: Round Robin (RR) is a pre-emptive algorithm used in multiprogrammed, conventional systems to schedule all the processes which are present in ready queue for execution. It has some advantages over other algorithms i.e., it gives a chance to all process to utilize processor for equal time interval. But this technique increases average turnaround time, average waiting time and if quantum value is very less, then CPU time is wasted in switching between processes and increases overheads. If it is high, the algorithm just works like FCFS and cannot be used in time sharing systems. The algorithm performance depends on quantum value. Turnaround time and waiting time are the criteria of the system which should be maintained as less as possible. Standard RR ($S_{RR)}$ algorithm does not posses logic in fixing quantum value. In our paper we propose Low-power Switching (LS) algorithm which reduces context switching and also reduces average waiting time and average turnaround time. So throughput of system will be raised. Experimental analysis shows the feasibility of the proposed algorithm which gives better turnaround time, waiting time and context switching compared with $S_{RR}$ technique and some related works. Pseudo code has been generated to prove the work.**

## I. INTRODUCTION

Central Processing Unit (CPU) is one of the main assets of computer architecture. The efficiency of the system fully depends on processor performance, so that CPU should be utilized properly i.e., CPU should be kept busy always in executing processes. In multi programmed systems multi processes can be submitted to ready queue for processing. Scheduling is a term used to arrange the processes in proper order in ready queue to improve CPU performance. To achieve this many algorithms are used like FCFS, SJF, SRTN, Priority scheduling and RR scheduling. RR scheduling is used widely in time sharing systems because it gives a chance to all the processes in queue to utilise CPU by quantum value. Quantum is a time slice value which is the time given for each and every process. If any process is finished within a quantum value, next process can use CPU and if process can be completed within mentioned quantum level, it will be switched to ready queue. The performance of scheduling algorithms are analysed with its average waiting time, average turnaround time, response time and throughput. Here waiting time is a time where the process spends in ready queue and turnaround time is finishing time of process and throughput corresponds to number of processes executed in unit time. Average waiting time and average turnaround time should be as less as possible to improve the throughput, because throughput is inversely proportional to average waiting time and turnaround time.

## II. RELATED WORK

Number of research works and papers concentrate on improving the throughput of round robin scheduling algorithm with various ideas. This section provides an overview of existing methods. In paper [1] it allows process to utilize a processor for fixed quantum value and remaining execution time of process is compared with threshold value(which is assumed as one fourth of quantum), if it is less than threshold value currently executed process can continue its execution or else suspended to ready queue. Here quantum value is not maintained as a constant for all process which makes RR working as FCFS algorithm for most of the cases.

Assigning process to CPU based on fixed quantum value is discussed in paper [2]. After process is finished Left out Time (LOT - remaining burst time) are compared with threshold value and if it is less than threshold value it is allowed to continue. Adaptive time-slice technique arranges process in order, based on its burst time and if number of processes are even, burst time of mid process are as taken ac quantum value or else average of all process are taken as quantum value as in paper [3].

Paper [4] proposes IRR algorithm which allows process to execute for fixed quantum. After that remaining burst time of currently executed process are compared with quantum and if it is less than fixed quantum, same process is allowed in continuing its execution. In paper [5] an optimal quantum value is found by average of median and highest burst times. Quantum value degenerates RR to work as FCFS algorithm in this paper.

Priority based RR algorithm proposed in paper [6], allows process to occupy processor for fixed quantum value and second round processes are arranged in ascending order based on burst time and highest priority is given to lowest burst time process. Its burst time will be considered as new quantum value from second round of execution. In [7] the proposed self adjustment quantum algorithm compares static quantum value with 25 and if it is lesser than 25, the value is changed to 25 or else as per quantum value scheduling will be performed. In paper [8] the author proposes a standard Round Robin algorithm with random fixed time slice value for all the process.

## III.PROPOSED WORK

In proposed systems, initially all the processes which are in ready queue are arranged in ascending order based on their burst time. For first Round, Robust Quantum value is generated by taking average of minimum burst time and maximum burst time processes value. Dispatcher will dispatch the first process from ready queue to CPU for execution. If burst time of process is lesser than or equal to robust quantum value, the particular process can fully utilize the processor or else, the quantum value will be suspended to ready queue. Once all the process finish their first round, rest of the process are arranged in order based on burst time and again new robust quantum is calculated and dispatching happens until the service is provided by CPU to all the processes.

Average turnaround time and average waiting time will be calculated after all the processes are executed. Context switching count is also calculated to find the context switching taken for scheduling the given process.

## IV.PSEUDO CODE

**Input:** BT (Burst time of process), Ni (Number of process in ready queue)

**Initialize:** ST (System Time) =0, $CS_C$ (Context Switch Count) =0

$Att_i$=Average Turnaround Time, $Awt_i$=Average Waiting Time

1.While (End of ready queue!=0)

// burst time based sorting

**For** a <- 0 to Ni-1 do:

Key = BT (a)

**For** b = a + 1 to Ni-1 do:

**If** BT (b) < BT (a)

Key =BT (b)

**End-If**

**End-For**

Tempe = BT[b]

BT[b] = BT [key]

BT [min] = Tempe

**End-For**

2.Average :=( BT [0] +BT [Ni-1])/2

RQ1=Integer (Average)

3. **For** i = 0 to Ni-1

**If** (BT [i] > RQ1)

BT[i] =BT [i]-RQ1

//Suspend process to ready queue

**End if**

4. $CS_C=CS_C+1$

WT[i]) =ST

ST=BT[i] +ST

TT[i] =ST

**5. If** BT (P[i]) <=RQ1)

Repeat Step 4

**End if**

**End – for**

**6.**$(Awt_i)= \sum_{i=0}^{Ni-1} WT(i)/Ni$

**7.**$Att_i=\sum_{i=0}^{Ni-1} TT(i)/Ni$

## V.ASSUMPTIONS

**A1**: Arrival time of process to the ready queue are assumed as zero

**A2**: Only processor time is most significant. Other resources like memory and input / output devices needs are negligible

**A3**: Milliseconds are considered as a time unit

## VI.EXPERIMENTAL ANALYSIS AND COMPARISION

To prove the proposed system, an example data is taken from [1-3] and analysed using LS algorithm. For an example consider a table 1. Here process names and burst times are taken as input from [1] and according to LS algorithm processes are arranged in ascending order based on burst time. Then Robust Quantum (RQ1) is calculated by taking an average of minimum burst time process and maximum burst time process. Scheduler

arranges the Process in Ready queue based on the burst time and dispatcher unit takes the first process to the processor for burst. After first round of processing is completed, P1, P2, P5 are removed from ready queue. Second round processes are arranged again based on burst time and new Robust Quantum value (RQ2) is generated for the remaining process P3 and P4. Gantt chart fig 1 to fig 3 shows the efficiency of the proposed system in reducing system criteria like average waiting time, average turnaround time and context switching count.

**CASE 1:**

| Process | Burst Time |
|---|---|
| P1 | 12 |
| P2 | 11 |
| P3 | 22 |
| P4 | 31 |
| P5 | 21 |

Table 1. Processes specification [1]

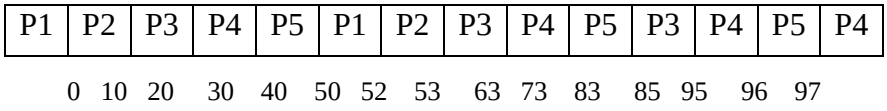


Fig 1. Gantt chart based on Standard Round Robin algorithm ($S_{RR}$)[1]

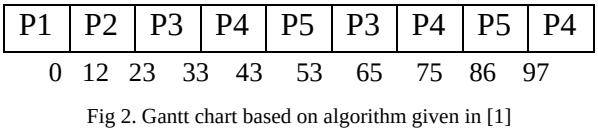


Fig 2. Gantt chart based on algorithm given in [1]

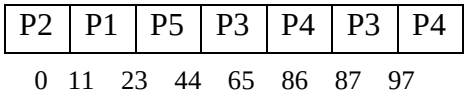


Fig 3. Gantt chart based on LS algorithm

## VII.COMPARISION ANALYSIS

To justify the proposed technique ,three cases are taken from paper [1] and user oriented criteria are calculated and compared with standard round robin algorithm ,a method proposed in paper [1]. The following table 2 shows the efficiency of proposed algorithm LS algorithm.

| | Case 1 | | | Case 2 | | | Case 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| Criteria | $S_{RR}$ | [1] | LS | SRR | [1] | LS | SRR | [1] | LS |
| (Quantum)$Q_i$ | 20 | dynamic | 21,5 | 20 | Dynamic | 52 | 10 | Dynamic | 20,6 |
| $Awt_i$ | 57.2 | 37.20 | **33** | 140.6 | 108.6 | **62.6** | 65.33 | 51.33 | **43.6** |
| $Att_i$ | 76.2 | 56.60 | **52.5** | 181.4 | 153.20 | **107.1** | 76.66 | 70.00 | **46.1** |
| CSc | 14 | 8 | **6** | 15 | 10 | **5** | 15 | 11 | **8** |

Table 2. Comparison Analysis of LS with [1]

Proposed algorithm comparison is performed on two more papers and the analysis table shows the benefits of LS algorithm in terms of its turn around, waiting and switching time

| Criteria | $S_{RR}$ | [3] | LS |
|---|---|---|---|
| $Q_i$ | 25 | 36 | 45 |
| $Awt_i$ | 70.2 | 56.8 | **49.6** |
| $Att_i$ | 109.6 | 96.2 | **89** |
| CSc | 7 | 6 | **4** |

Table 3. Comparison analysis with [3]

| Criteria | $S_{RR}$ | [2] | LS |
|---|---|---|---|
| $Q_i$ | 10 | dynamic | 22 |
| $Awt_i$ | 64.83 | 39.8 | **36.1** |
| $Att_i$ | 83.83 | 58.8 | **57.6** |
| CSc | 14 | 8 | **7** |

Table 4. Comparison analysis with [2]

## VIII. COMPARISION CHART

The proposed algorithm is also compared with the performances of paper [1-3] and comparison chart diagram is shown in fig 4 to fig 6.

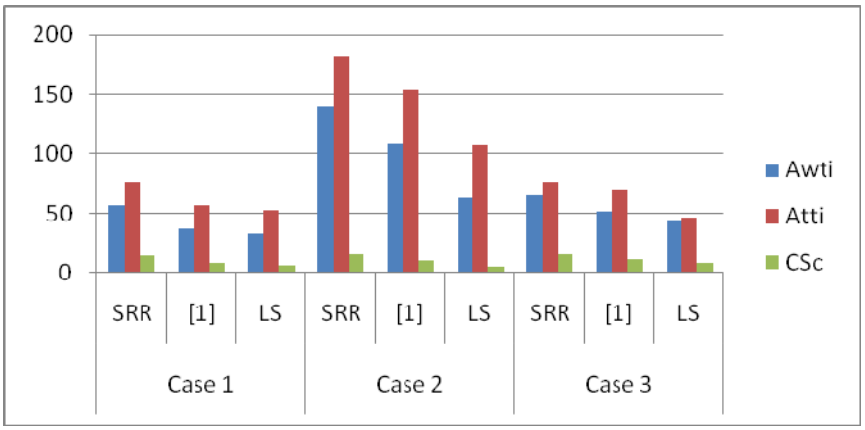


Fig 4. Analysis chart of LS with [1]

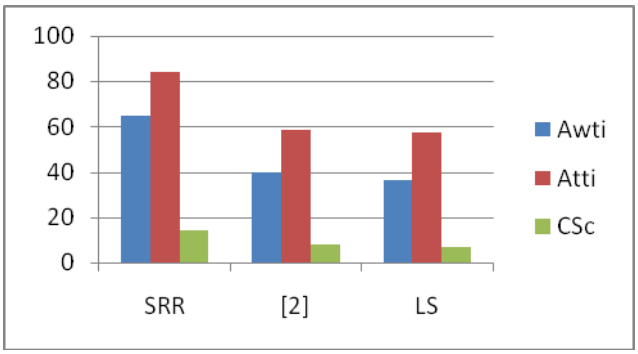


Fig 5. Analysis chart of LS with [2]

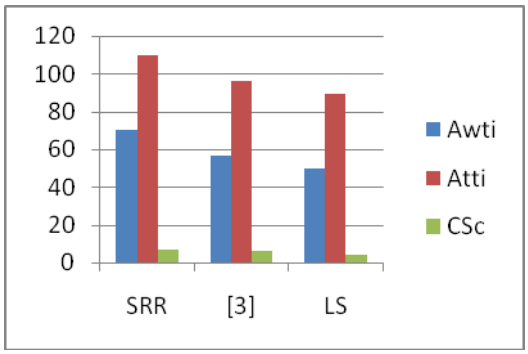


Fig 6. Analysis chart of LS with [3]

## IX.CONCLUSION

Scheduling algorithms for uni-processor systems should be very efficient to make a feasible system. As a resource manager, operating systems should maintain processor usage properly by implementing high performance scheduling algorithms. The algorithm which is proposed in this paper provides good Quality of Service by maximum reduction of context switching time, waiting time and turnaround time of all the process. Future work can be extended with specific arrival times.

## X.REFERENCES

[1] Sandeep Negi,"An Improved Round Robin Approach using Dynamic Time Quantum for Improving Average Waiting Time" International Journal of Computer Applications, 2013

[2] Mohd Abdul Ahad, "Modifying Round Robin Algorithm for Process Scheduling using Dynamic Quantum Precision", International Journal of Computer Applications, 2012

[3] Vishnu Kumar, Dhakad, Lokesh Sharma, "Performance analysis of Round Robin scheduling using adaptive approach based on smart time slice and comparison with SRR", International Journal of Advances in Engineering & Technology, 2012.

[4] Mishra, Abdul Kadir Khan, "An Improved Round Robin CPU scheduling algorithm" Journal of Global Research in Computer Science, 2012

[5] Debashree Nayak, Sanjeev Kumar Malla, Debashree Debadarshini, "Improved round robin Scheduling using dynamic time quantum", International Journal of Computer Applications, 2012

[6] Ishwari Singh Rajput, Deepa Gupta, "A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems", International Journal of Innovations in Engineering and Technology, 2012

[7] Rami J. Matarneh, "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes" American Journal of Applied Sciences, 1831-1837, 2009

[8] William Stallings, "Operating Systems Internals and Design Principles", Seventh Edition, Prentice Hall, 2011