A New RR Scheduling Approach for Real Time Systems using Fuzzy Logic

Lipika Datta
Assistant Professor, CSE Dept.
CEMK ,Purba Medinipur
West Bengal, India

ABSTRACT
Round Robin scheduling algorithm is the widely used scheduling algorithm in multitasking. It ensures fairness and starvation free execution of processes. It performs optimally for time sharing systems, but because of its larger waiting time, turnaround time and greater number of context switches it is not suitable for soft real time systems. The main objective of this paper is to develop a way in which the Round Robin algorithm can be modified for implementation in real time and embedded systems by minimizing its average waiting time, average turnaround time and context switching rate. The paper discusses a fuzzy based CPU scheduling algorithm. A set of fuzzy rules is defined. Each process is assigned a new priority based on its externally defined priority, relative remaining CPU burst time and relative waiting time.

General Terms
Algorithms, CPU scheduling

Keywords
Operating System, Fuzzy logic, CPU scheduling algorithm, Priority, Average Turnaround time, Average Waiting time

1. INTRODUCTION
Modern operating systems support multitasking environment in which processes run in a concurrent manner. In a single-processor system, only one process can run in the CPU at a time. Others processes in the ready queue must wait until the CPU becomes free. The operating system must decide through the scheduler the order of execution of the processes in ready state. The objective of multiprogramming is to have some process running at all times to maximize CPU utilization. Scheduling is a fundamental operating-system function. Almost all computer resources are scheduled before using. The CPU is, of course, one of the primary computer resources. Thus, its scheduling is central to operating-system design. CPU scheduling determines which processes run when there are multiple run-able processes. CPU scheduling is important because it can have a big effect on resource utilization and the overall performance of the system. In general we want to optimize the behavior of the system. The goals of scheduling may be categorized as user based scheduling goals and system based scheduling goals [1]. User based goals are the criteria that benefit the user. Some User-based scheduling goals are:

- Turnaround Time: The time elapsed between the submission of a job and its termination is called the turnaround time.

  $t_r = wt + x$

  where t_r is turnaround time of a process

  wt is waiting time of the process in the ready queue.

  x is the execution time of the process.

  The scheduling algorithm should be designed such that turnaround time is minimized.

- Waiting Time: The time spent by the process in the ready queue is the waiting time. The scheduling algorithm should be designed such that waiting time is less.

- Response Time: It is the time period between the time of submission of a process and the first response given by the process to the user. The scheduling algorithm should be designed such that the response time is within an acceptable range.

- Predictability: The algorithm should take care that a process does not take too long in processing as compared to the predictable behavior of the process.

- Deadlines: The scheduling algorithm should be designed such that real-time processes will execute within their deadlines.

Some system-based scheduling goals are:

- Throughput: Throughput is the number of processes completed in a unit time. The scheduling algorithm should be designed in such a way that throughput in a system is maximized.

- CPU Utilization: It is the percentage of time that the CPU is busy in executing a process. The fundamental goal of scheduling is to keep the processor busy all the time.

- Fairness: All processes in the system should be treated in the same way unless there is some preference or priority for a specific process. In that case also processes with lower priority should not be ignored to avoid starvation.

- Context Switch**:** Context switching is the procedure of storing the state of an active process and restoring the state of another process for the CPU when it has to start executing the later process. Context switch is total overhead to the system and leads to wastage of CPU time. The scheduling algorithm should be designed such that the context switch be minimum.

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

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

Real time system applications are mission-critical. The real-time tasks should be scheduled to be completed before their deadlines. Most real-time systems handle unpredictable environments. So, the real time operating system should handle unknown and changing task populations. In this case, not only a dynamic task scheduling is required, but both the system hardware and software must adapt to unforeseen configurations [2].

2. RELATED WORK DONE

RR algorithm performs optimally in timeshared systems, but it is not suitable for soft real time systems because of its higher context switching rate, larger waiting time and larger turnaround time. Some researchers have already introduced some variations of RR scheduling algorithm. But these algorithms have some limitations. In [3] authors have proposed an algorithm in which according to the given priority the CPU is allocated to the processes only once in RR fashion for a given time quantum. Then, processes are arranged in increasing order of their remaining CPU burst time in the ready queue. New priority is assigned to each process following the rule that lesser the remaining burst time higher the priority. Then, processes are allocated CPU according to non-preemptive priority scheduling algorithm. If this algorithm is used after first response from the system user may have to wait long for next response. So, a fairness criterion is not held. In [4] different time slices are calculated for different processes based on three aspects: user defined priority, average CPU burst, context switch avoidance time. An assumption is made on average CPU burst. In [5] also different time slices are calculated for different rounds of RR scheduling algorithm based on remaining CPU burst time. In [6] the authors have introduced a concept called intelligent time slicing which depends on priority, next CPU burst and original time slice. The time slice is static. This algorithm is modified to get different time slice values in different rounds for different processes in [7] Algorithm with Intelligent Time Slice for Soft Real Time Systems). It calculates the initial time slice for each process as the previous algorithm [6] and in each round the time slices for each of the processes is modified depending on the priority of the processes and the original time slice. In [8] the authors have made the priority and time slice for a process dynamic by calculating the weighted mean values of time quantum and priorities of the processes and considering the burst time of the processes. The algorithm introduced in [9] calculates the time slice of each process in each round dynamically considering the priority of a process and the average and shortest burst time of all the currently running processes. Fuzzy logic is applied in the design and implementation of a rule-based scheduling algorithm to solve the shortcoming of well-known scheduling algorithms in [10]. This algorithm is modified for better result in [11]. In [12] authors have introduced another parameter-waiting time, while designing the fuzzy inference system.

2.1 My contribution

In my work, an improved RR algorithm is proposed, which calculates dynamic time slices for different rounds of RR scheduling algorithm considering the remaining CPU bursts of the currently running processes. In each round priority of each process is calculated depending on the remaining burst time, waiting time and the static priority of the process. Then, according to the new priority the processes are scheduled in that round. Experimental result shows that my algorithm is better than existing algorithms in terms of average turnaround time, average waiting time and number of context switches.

2.2 Organization of paper

Section 3 presents the illustration of my proposed algorithm. In section 4, Experimental results and its comparison with existing algorithms is presented. Section 5 contains the conclusion.

3. PROPOSED APPROACH

The proposed algorithm eliminates the drawbacks of implementing simple round robin architecture in real time system by introducing a concept of assigning different time slices to in different rounds of RR scheduling algorithm. The static priority of a process is assigned by user externally. In the proposed architecture in each round the priority of each process is reassigned depending on its remaining CPU burst, waiting time until that round and the value of its static priority. It is assumed that lesser number implies higher priority. A small dedicated processor is used to calculate the priority of each process at the beginning of each round using Fuzzy inference engine and the time quantum of that round. It arranges the processes in ascending order of their newly assigned priorities and then creates the ready queue for the main processor. This small dedicated processor is used to reduce the burden of the main processor. The processes then execute in the main processor according to round robin scheduling algorithm. At the beginning of each round of the RR algorithm the following matrices are calculated for each process at the dedicated processor:

a) $$RR = \frac{Remaining\ CPU\ burst\ of\ the\ process}{Total\ remaining\ CPU\ burst\ of\ current\ processes}$$

b) $$RW = \frac{Waiting\ time\ of\ the\ process\ till\ now}{Total\ waiting\ time\ of\ all\ currently\ running\ processes,\ till\ now}$$

c) Priority Ratio (PR):

$$= \frac{Processes's\ static\ priority}{Highest\ static\ priority\ among\ the\ current\ processes}$$

In this paper suitable linguistic variables are used as input and output for compute a crisp value for new priority. Relative Remaining Burst (RR) measured as Small, Medium and Large. Relative waiting time (RW) measured as Small, Medium and Large. Priority Ratio (PR) measured as Small, Medium and Large. New Priority (NP) measured as Very small, Small, Medium, Large and Very Large. The proposed scheduling algorithm is a collection of linguistic fuzzy rules which describe the relationship between defined input variables (RR, RW and PR) and output (NP).

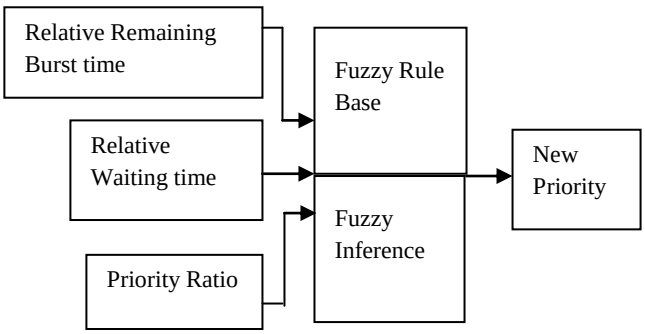

Fig 1: Block diagram of Fuzzy Inference System

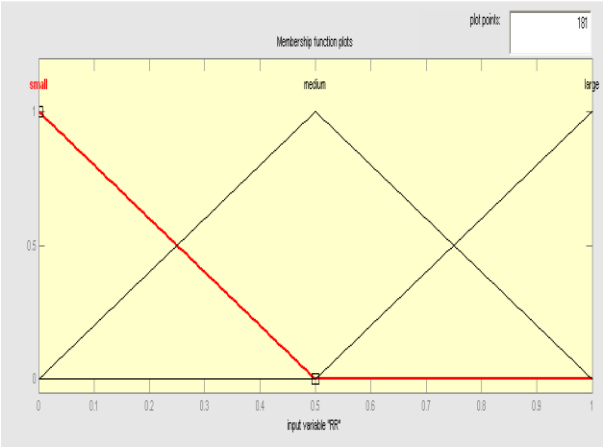

Fig 2: Membership Function for Relative Remaining Burst Time

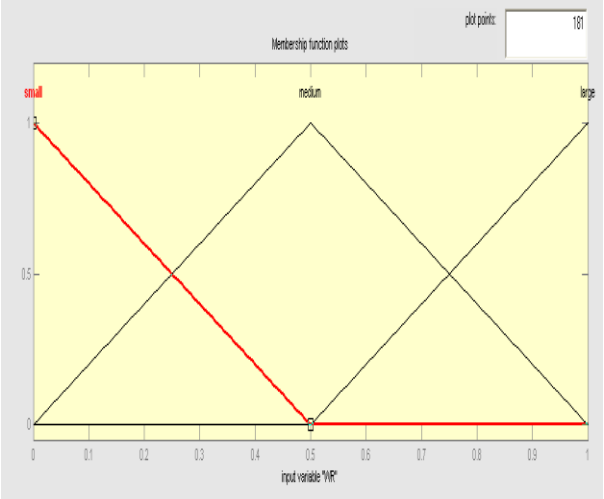

Fig 3: Membership Function for Relative Waiting Time

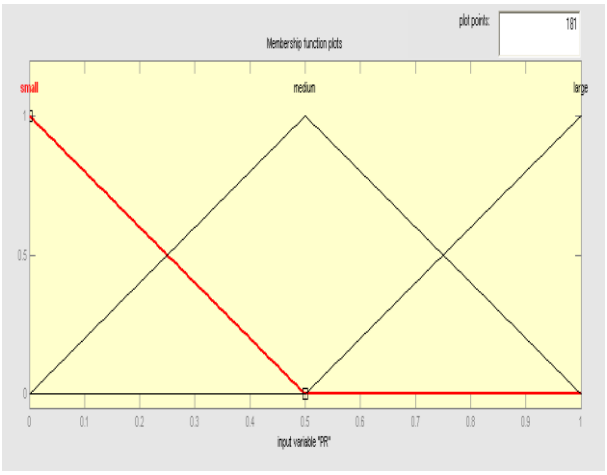

Fig 4 : Membership Function for Relative Waiting Time

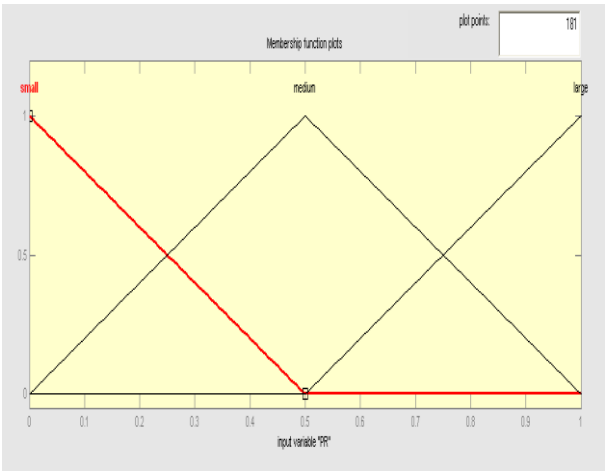

Fig 5 : Membership Function for Static Priority Ratio

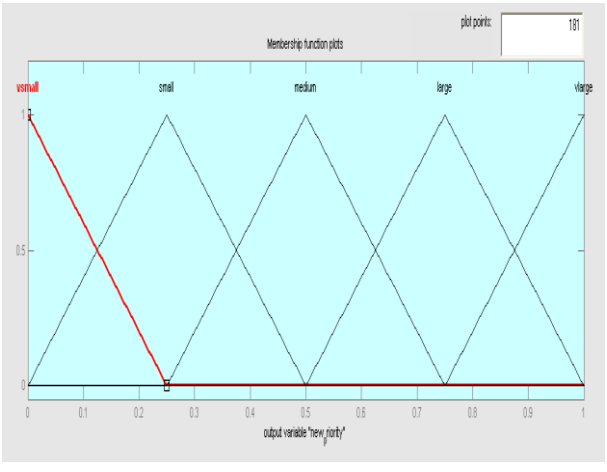

Fig 6 : Membership Function for New Priority

The following table contains 27 rules which are based on IF THEN statement such as: -

If RR is Small, WR is Small and PR is Small then NP is Very Small.

Table 1. Fuzzy Rules for proposed design

RR	WR	PR	NP
Small	Large	Small	Very Small
Small	Large	Medium	Very Small
Small	Large	Large	Small
Small	Medium	small	Very Small
Small	Medium	Medium	Very Small
Small	Medium	Large	Small
Small	Small	Small	Very Small
Small	Small	Medium	Small
Small	Small	Large	Small
Medium	Large	Small	Small
Medium	Large	Medium	Small
Medium	Large	Large	Medium
Medium	Medium	Small	Small
Medium	Medium	Medium	Medium
Medium	Medium	Large	Medium
Medium	Small	Small	Medium
Medium	Small	Medium	Large
Medium	Small	Large	Large
Large	Large	Small	Medium
Large	Large	Medium	Large
Large	Large	Large	Very Large
Large	Medium	Small	Large
Large	Medium	Medium	Very Large
Large	Medium	Large	Very Large
Large	Small	Small	Very Large
Large	Small	Medium	Very Large
Large	Small	Large	Very Large

These rules compute the crisp value using Centroid Defuzzification method of Mamdani inference in MATLAB that represents the NP of each task.

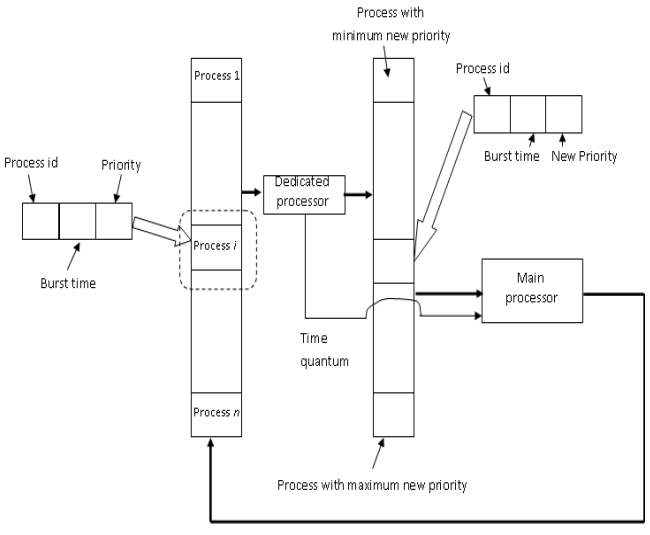

Fig 7 : Proposed Architecture

4.1 Proposed Algorithm

1. Initialize n processes with their burst time and priority.
2. Let RBT_i be the remaining CPU burst of P_i
   WT_i be the waiting time of P_i till that time
   TWT_i be the total waiting time of all the currently running processes till that time.
3. TQ = $\sum_{i=1}^{n} \frac{RBTi}{n}$
4. For each process calculate the following parameters:
   $$RR = \frac{RBTi}{\sum_{i=1}^{n} RBTi}$$
   $$WR = \frac{WRi}{\sum_{i=1}^{n} WRi}$$
   $$PR = \frac{PRi}{Maximum\ Priority}$$
5. Calculate new priority of each process using Fuzzy Rules.
6. Sort the processes in ready queue in the ascending order of new priority.
7. Schedule processes from ready queue according to RR algorithm with time quantum TQ.
8. For each process calculate $RBT_i = RBT_i – TQ$
9. If RBT_i <=0 remove the process from the ready queue.
10. If new process arrives wait for current time slice to expire or completion of execution of current process whichever is earlier and goto step 4.
11. Repeat steps 3 to 10 until ready queue is empty.
12. Calculate Average Waiting time, Average Turnaround time, number of Context Switch.

4. EXPERIMENTAL RESULTS

4.1 Assumptions:

Experiments are performed in single processor environment and on independent processes. All the parameters like number of processes, and burst time of all the processes are known before submitting the processes to the processor. All processes are CPU bound and none I/O bound. Context switching overhead and time taken for calculating the time slices are ignored while calculating average turnaround time and average waiting time.

4.2 Data set:

To compare the performance of the algorithm with the algorithms introduced in [7] (PBDRR), [9] (DTQ), three cases of the data set are considered: the processes with burst time in increasing, decreasing and random order respectively.

Same data set applied to PBDRRR, DTQ and Proposed Algorithm:

TABLE 2. Inputs for case 1

Process id	Priority	Burst time
P1	2	5
P2	3	12
P3	1	16
P4	4	21
P5	5	23

TABLE 3. Calculation of New Priority for proposed algorithm for case 1

			Round 1	Round 2
Pid	Priority	Burst time	NP	NP
P1	2	5	0.34	-
P2	3	12	0.434	-
P3	1	16	0.449	-
P4	4	21	0.515	0.501
P5	5	23	0.533	0.554

TABLE 4. Comparison between algorithms for case 1

Algorithm	Average Turnaround Time	Average Waiting Time	No. of Context Switch
PBDRR	46.4	31	17
DTQ	47.2	31.8	10
Proposed Algorithm	40.4	25	6

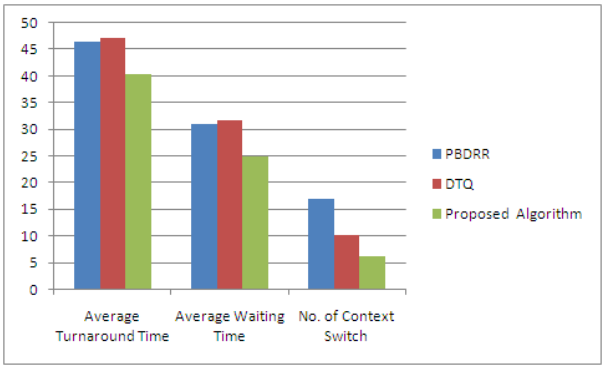

Fig 8: Analysis of performance among algorithms (case 1)

TABLE 5. Inputs for case 2

Process id	Priority	Burst time
P1	2	31
P2	1	23
P3	4	16
P4	5	9
P5	3	1

TABLE 6. Calculation of New Priority for proposed algorithm for case 2

			Round 1	Round 2
Pid	Priority	Burst time	NP	NP
P1	2	31	0.589	0.603
P2	1	23	0.472	0.463
P3	4	16	0.466	-
P4	5	9	0.398	-
P5	3	1	0.273	-

TABLE 7. Comparison between algorithms for case 2

Algorithm	Average Turnaround Time	Average Waiting Time	No. of Context Switch
PBDRR	50.4	34.4	12
DTQ	54.8	38.8	15
Proposed Algorithm	36.4	20.4	6

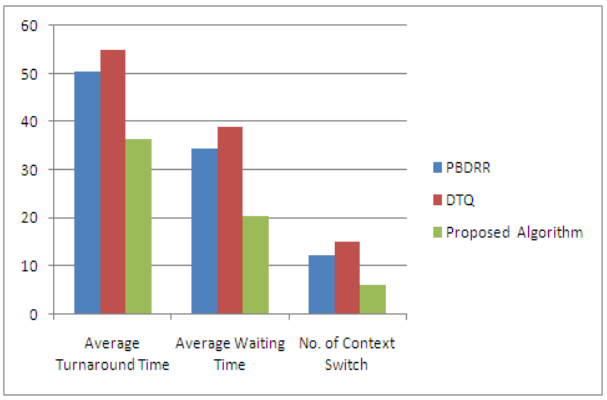

Fig 9: Analysis of performance among algorithms (case 2)

TABLE 8. Inputs for case 3

Process id	Priority	Burst time
P1	3	11
P2	1	53
P3	2	8
P4	4	41
P5	5	20

TABLE 9. Calculation of New Priority for proposed algorithm for case 3

			Round1	Round2
P id	Priority	Burst time	NP	NP
P1	3	11	0.37	-
P2	1	53	0.532	0.481
P3	2	8	0.34	-
P4	4	41	0.538	0.367
P5	5	20	0.43	-

TABLE 10. Comparison between algorithms for case 3

Algorithm	Average Turnaround Time	Average Waiting Time	No. of Context Switch
PBDRR	76	49.4	18
DTQ	79.8	53.2	11
Proposed Algorithm	61.2	30.8	6

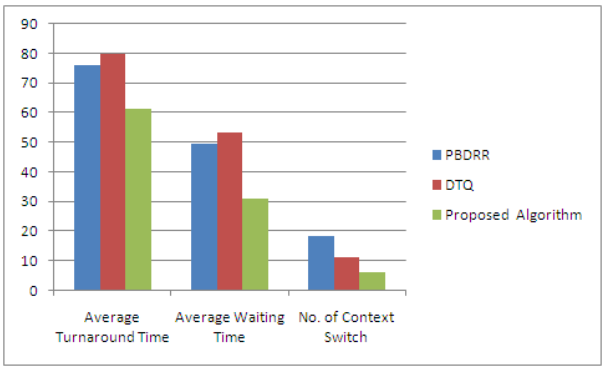

Fig 10: Analysis of performance among algorithms (case 3)

5. CONCLUSION

From the above comparisons, it is observed that the proposed algorithm is performing better than the algorithm PBDRR proposed in paper [7] and DQT proposed in paper [9] in terms of average waiting time, average turnaround time and number of context switches. If the CPU burst times of the processes vary very widely the algorithm doesn't produce good result. In the other cases the quality of service can be improved and overhead can be reduced. Thus, memory space which is an important constraint for embedded system applications can be saved. Real time systems must meet their deadlines. Deadlines of tasks can be considered in future work as a new input parameter. So, some new rules may be added to the fuzzy rule set while calculating the new priority of processes in each round.

6. REFERENCES

[1] Principles of Operating System, Naresh Chauhan, Oxford University Press, 2014

[2] 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)

[3] 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 (IJIET) Vol. 1 Issue 3 Oct 2012

[4] C.Yaashuwanth, Dr.R.Ramesh "A New Scheduling Algorithms for Real Time Tasks", International Journal of Computer Science and Information Security, Vol. 6, No.2, 2009

[5] Rakesh Mohanty , H. S. Behera , Debashree Nayak "A New Proposed Dynamic Quantum with Readjusted Round Robin Scheduling Algorithm and its performance Analysis", International Journal of Computer Applications (0975-8887),Volume 5-No.5, 2010

[6] Yaashuwanth .C, Dr.R. Ramesh, "A new scheduling algorithm for real time tasks", International Journal of Computer Science and Security, Vol 6, No 2, 2009

[7] Rakesh Mohanty , H. S. Behera , Khusbu Patwari, Monisha Dash , M. Lakshmi Prasanna "Priority Based Dynamic Round Robin (PBDRR) Algorithm with Intelligent Time Slice for Soft Real Time Systems", International Journal of Advanced Computer Science and Applications, Vol. 2, No.2, February,2011

[8] H.S.Behera, Sabyasachi Sahu and Sourav Kumar Bhoi, "Weighted mean priority based scheduling for interactive systems", Journal of global reaearch in computer science,2011

[9] Lipika Datta, "Modified RR Algorithm with Dynamic Time Quantum for Externally Prioritized Tasks", International Journal of Recent and Innovation Trends in Computing and Communication, Volume 3, Issue 1,January 2015

[10] Shatha J. Kadhim and Kasim M. Al-Aubidy," Design and Evaluation of a Fuzzy-Based CPU Scheduling Algorithm", Information Processing and Management (2010): 45-52.

[11] Rajani Kumari, Vivek Kumar Sharma, Sandeep Kumar," Design and Implementation of Modified Fuzzy based CPU Scheduling Algorithm", International Journal of Computer Applications (0975 – 8887) Volume 77 – No.17, September 2013

[12] Bashir Alam, M.N. Doja1, R. Biswas, M. Alam," Fuzzy Priority CPU Scheduling Algorithm", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 6, No 1, November 2011 ISSN (Online): 1694-0814