Volume 3, No. 3, March 2012
Journal of Global Research in Computer Science
ISSN-2229-371X
REVIEW ARTICLE
Available Online at www.jgrcs.info

DESIGN AND PERFORMANCE EVALUATION OF THE ADVANCE MIX JOB WITH DYNAMIC QUANTUM ROUND ROBIN SCHEDULING ALGORITHM FOR REAL TIME SYSTEMS

H.S.Behera[*1], Bijayalaxmi Panda[2], Sreelipa Curtis[3]
Department of Computer Science and Engineering,
Veer Surendra Sai University of Technology, Burla, Odisha, India
hsbehera_india@gmail.com[1]
bijayafouru@gmail.com[2]
csrilipa@gmail.com[3]

Abstract: The purpose of an operating system is to provide an interface in which a program can execute in a convenient and efficient manner. While designing an operating system, a programmer must consider which scheduling algorithm will perform best for the system. There is no universal "best" scheduling algorithm, and many operating systems are using extended or combinations of the scheduling algorithms. The efficiency of scheduling depends on optimal time quantum and proper distribution of system resources. So, we have proposed a new algorithm known as advanced mix job with dynamic quantum round robin (AMDRR) scheduling algorithms. We have experimentally shown that the efficiency of AMDRR is better than conventional RR(round robin) and DQRRR (dynamic quantum readjusted round robin) by reducing its context switching, average turnaround time and average waiting time.

Keywords: Scheduling, round robin, burst time, waiting time, turnaround time, context switching, priority

INTRODUCTION

Scheduling is a key concept in computer multitasking, multiprocessing operating system and real time operating system designs. Scheduling refers to the way processes are assigned to run on the available CPU. There are typically many more processes running than they are available on CPU. This assignment is carried out by a software know as scheduler and dispatcher. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute. Scheduler can be long term scheduler, short term scheduler. Long term scheduler determines which jobs are admitted to the system for processing. Long term scheduler executes less frequently than the short-term scheduler and controls the degree of multiprogramming. Medium term scheduler swaps jobs in and out of memory to reduce contention for the CPU. Short term scheduler selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them. In time sharing system, the CPU will execute multiple jobs by switching among them, but the switches occur so frequently that the users can interact with each program while it's running.

Dispatcher is a module that gives control of CPU to the process selected by short term schedulers. Time taken for the dispatcher to stop one process and start running another is known as dispatcher latency. It must be short because every context switch invokes dispatcher. In some real-time scheduling algorithms, a task can be preempted if another task of higher priority becomes ready. In contrast, the execution of a non-preemptive task should be completed without interruption, once it is started. A schedule is called *preemptive* if each task may be preempted at any time and restarted later at no cost, perhaps on another processor. If preemption is not allowed then the schedule is *non-preemptive*.

Scheduling Algorithm:

In first-in first- serve (FCFS) algorithm, the process that requests CPU first is allocated to CPU first. The implementation of this algorithm is easily done with FIFO queue. Shortest-Job-First (SJF) is non-preemptive discipline in which waiting job (or process) with the smallest estimated run-time to completion is run next. In other words, when CPU is available, it is assigned to the process that has smallest next CPU burst. In case of priority scheduling, priority is assigned to each process and CPU is allocated to the process with highest priority. Equal priority processes are scheduled in FCFS order. Round robin (RR) algorithm is used for time sharing systems. It is similar to FCFS with preemption.

Motivation:

In RR scheduling a fixed time quantum is given to all process that are submitted in ready queue. So there is frequent switching between processes by which efficiency of CPU decreases. If the time slice is a large one then waiting time and turnaround time increases. So to overcome these above situations, we have proposed an algorithm that uses priority, modified shortest job first and dynamic time quantum concept.

Related Work:

SARR algorithm uses a new approach that it is using dynamic time quantum in which time quantum is repeatedly changing with respect to their burst time. Mixed scheduling uses two non-preemptive type scheduling i.e. FCFS and SJF. According to mixed job first scheduling the process with minimum time will be executed first then the process with maximum burst time and so on. DQRRR algorithm uses dynamic quantum concept. Quantum is chosen by finding the median of burst time and it changes when all the process execute once.

Our Contribution:

In this paper, the main objective is to reduce average waiting time and turnaround time occur in RR and DQRRR scheduling. For this purpose, we have developed a method that drastically reduces average waiting time and turnaround time and switching between processes.

Organization of Paper:

This paper represents a method for reducing context switching, average waiting time and average turnaround time using random sorting and dynamic quantum with burst task component and priority task component. Section 2 describes all preliminary work. Section 3 presents our proposed approach, algorithm and flow-chart. Section 4 shows experimental analysis and comparison of result. In Section 5 conclusion is given.

BACKGROUND WORK

Terminology:

A program in execution is known as process. To schedule process, processor should know its arrival time, burst time and time-slice assigned for each process. Burst time is the amount of time a process uses the CPU for a single time. To calculate the efficiency of scheduling waiting time, turnaround time and context switching plays an important role. Turnaround time is the total amount of time to execute a particular process. It is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU. Waiting time is the sum of the periods a process spent waiting in the ready queue .Response time is the amount of time process takes from when a request was submitted until the first response is produced. The number of times CPU switches from one process to another is called as context switching.

RR Scheduling Algorithm:

In RR, each ready task runs turn by in turn in a cyclic queue for limited time slices. It is widely used in traditional OS. RR is a hybrid model i.e. clock driven model (e.g. cyclic model) as well as event driven (e.g. Preemption). The performance of RR algorithm is highly dependent on time slice. For low time-slice context switching is more and for high time-slice response time is more. So the time quantum plays most determining factor for the performance of RR algorithm...

DQRRRR Scheduling Algorithm:

Dynamic quantum re-adjusted round robin (DQRRRR) is a method for scheduling which uses dynamic time quantum. Time quantum is calculated by finding the median of burst time. The process with shortest burst time will execute first then the process with next shortest burst time and so on. Time quantum changes when all processes execute once. The method is continued till there is no process in ready queue.

PROPOSED APPROACH

In this approach, the processes are arranged in ready queue in ascending order according to their scheduling time. To get optimised result for time quantum, median is calculated. The median is calculated as follows:

$$\text{Median} = \begin{cases} 1/2\left(Y_{(N/2)} + Y_{)(1+N/2)}\right) & \text{if N is even} \\ Y_{(N+1)/2} & \text{if N is odd} \end{cases}$$

The scheduling time is calculated as follows:
Scheduling time= (burst time* burst task component) + (priority * priority task component);
The determinant factor is calculated as follows:

Determinant factor = (max of scheduling time of processes + min of scheduling time of processes)/2;
Now time quantum is calculated as follows:
Time quantum = (median + determinant factor)/2;

Time quantum is assigned to each process. Median is recalculated with remaining scheduling time after each execution of each cycle. In the next step, the processes which need lowest scheduling time will be replaced as first processes then with next lowest scheduling time from queue will be replaced as second process and so on.

Pseudo code of our Proposed Algorithm:

1. I.P: process(Pi), Burst time(Bt), arrival time(At),priority(P),burst task components(btc), priority task components(Ptc)
O.P: context switch (CS), average turnaround time (atat), average waiting time (awt).

2. Initialize the ready queue =0, CS=0, awt=0, atat=0, St=0, Detfact=0.

3. St=(Bt*btc + P*ptc)

4. Detfact= ((max of St+ min of St) of processes)/2.

5. While (ready queue== NULL)
    Sort the process in ascending order in
    ready queue according to scheduling time.
    // Find median;
    Qt=(median+Detfact)/2;

6. for each process i=1 to n
    do
    {
    if(i%2==0)
        put minimum amount in ready queue.
    else
        put the maximum in ready queue;
    }//end of for

7. //Assign Qt to each process
    for each process i=1 to n
        p[i]->Qt;

7. //if a new process arrives
    update the counter n and goto step 2;
    end while.
    awt, atat,CS is calculated.

8. stop and exit.

Flowchart:

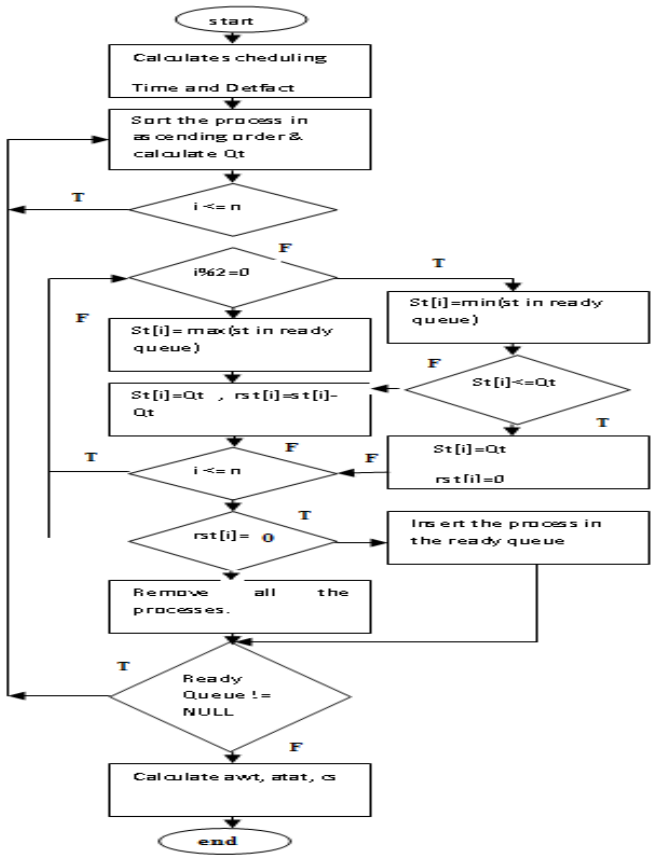

Figure: 1

Illustration:

Let us explain above algorithm with an example. Here arrival time is considered to be zero. The processes are P1, P2, P3, and P4 with their burst time 19, 92, 107, 72 and priority of these processes are 10, 2,15,7 respectively. Burst task component and priority task component is assumed to be 60% and 40% respectively. So in first step we need to calculate scheduling time as described in step-3 like P1, P2, P3, and P4 as 15,56,70 and 46 respectively. Then arrangement occurs in ready queue in ascending order as per scheduling time. Now median is calculated. Determinatfactor is calculated as described in step-4. Now time quantum is calculated by taking the half of addition of median and determinant factor. Time quantum is assigned to processes. Here the time quantum is calculated as Qt=47.In next step scheduling of the processes are to be done as described in step-6 i.e. P1 with St=15, P4 with St=46, P3 with St=70, P2 with St=56. After assignment of time quantum to each process, remaining scheduling time will be P1:0, P4:0, P2:9, P3:23 and in this case time quantum is calculated as Qt=30. When executation of a process completed, it is automatically deleted from ready queue and scheduling is done by step-6.

EXPERIMENTAL ANALYSIS:

***Assumption*:**

All experiments are performed in a uni-processor environment. All processes are independent of each other. Here P1, P2 ...Pn n-processes are taken. Burst time and priority of corresponding process are known before submitting the task to the processor. Burst task component and priority task component can be taken constant for n-number of processes.

Experimental Frame Work:

Pn is the number of processes. The input parameters for the processes are burst time, arrival time, priority which is BT, AT and P respectively. The output parameters are Context switch (CS), average waiting time (awt), average turnaround time (atat).

Data Set:

We have considered two cases here. Case-1 is for process with zero arrival time.Case-2 is for process with certain arrival time. In both case-1 and case-2, there are 3-subcases i.e. processes are taken in ascending, descending and random order.

Performance Metrics:

The significance of our performance metrics for experiment analysis is as follows:

a. Turnaround time (TAT): For the better performance of scheduling algorithm, turnaround time should be less.
b. Waiting time (WT): For better performance of scheduling algorithm, waiting time for processes should be less.
c. Context switch (CS): The number of context switches should be low for better result of proposed algorithm.

Experiments Performed:

To evaluate the efficiency of our proposed algorithm (AMDRR), the output parameters are compared with round robin (RR) and Dynamic quantum with re-adjusted round robin (DQRRR). This algorithm can work effectively with large number of processes. For simplicity we have taken five processes with ascending, descending and random order to illustrate our proposed algorithm.

Results Obtained:

Here we are considering two cases i.e. processes are with zero arrival time in case -1 and processes are with certain arrival time in case-2. For RR scheduling algorithm we have taken 25 as the fixed time quantum. Burst task component and priority task component are taken as 60% and 40% respectively.

CASE 1: With Zero Arrival Time

Increasing Order:

We have considered five processes P1, P2, P3, P4, P5 arriving at time 0 with burst time 30, 42,50,85,97 respectively and priority of each process shown in table 4.6.1. Table 4.6.2 shows the comparing result of RR, DQRRR and our proposed algorithm (AMDRR).

Table 4.6.1.Data in increasing order

No. of process	At	Bt	P	St
P1	0	30	10	22
P2	0	42	15	31
P3	0	50	5	32
P4	0	85	7	54
P5	0	97	4	60

Table 4.6.2 comparison among RR, DQRRR and AMDRR

Algorithms	RR	DQRRR	AMDRR
Qt	25	50,41,6	37,31
CS	13	7	5
Awt	146.2	134.4	74.6
Atat	207	195.2	135.4

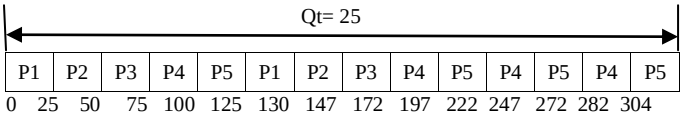

Figure. 4.6.1: Gantt chart for RR in Table 4.6.1

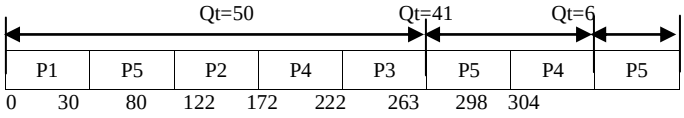

Figure. 4.6.2: Gantt chart for DQRRR in table 4.6.1

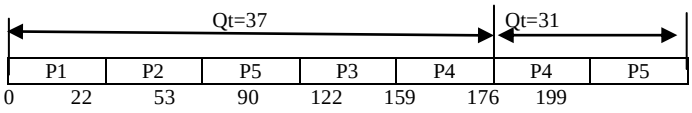

Figure. 4.6.3: Gantt chart for AMDRR in Table 4.6.1

Decreasing Order:

We have considered five processes P1, P2, P3, P4, P5 arriving at time 0 with burst time 105, 90, 60,45,7 respectively and priority of each process shown in table 4.6.3. Table 4.6.4 shows the comparing result of RR, DQRRR and our proposed algorithm (AMDRR).

Table 4.6.3.Data in decreasing order

No. of process	At	Bt	P	St
P1	0	105	20	71
P2	0	90	10	58
P3	0	60	3	37
P4	0	45	15	33
P5	0	7	7	24

Table 4.6.4 comparison among RR, DQRRR and AMDRR

Algorithms	RR	DQRRR	AMDRR
Qt	25	60,37,8	42,46
CS	15	7	5
awt	214	152.4	82.2
atat	281	219.4	143.6

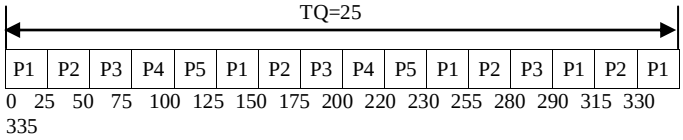

Figure. 4.6.4: Gantt chart for RR in Table 4.6.3

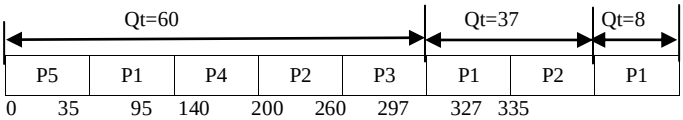

Figure. 4.6.5: Gantt chart for DQRRR in table 4.6.3

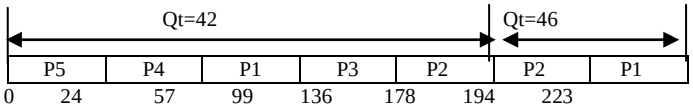

Figure. 4.6.6: Gantt chart for AMDRR in Table 4.6.3

Random Order:

We have considered five processes P1, P2, P3, P4, P5 arriving at time 0 with burst time 92,70,35,40,80 respectively and priority of each process shown in table 4.6.5. Table 4.6.6 shows the comparing result of RR, DQRRR and our proposed algorithm (AMDRR).

Table 4.6.5 .Data in random order

No. of process	At	Bt	P	St
P1	0	92	10	53
P2	0	70	2	43
P3	0	35	5	23
P4	0	40	35	38
P5	0	80	4	47

Table 4.6.6 comparison among RR, DQRRR and AMDRR

Algorithms	RR	DQRRR	AMDRR
Qt	25	60,37,8	41,22
CS	15	7	7
Awt	214	152.4	92.4
Atat	281	219.4	155.8

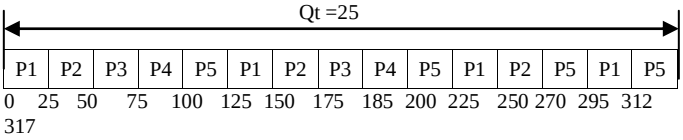

Figure.4.6.7: Gantt chart for RR in Table 4.6.5

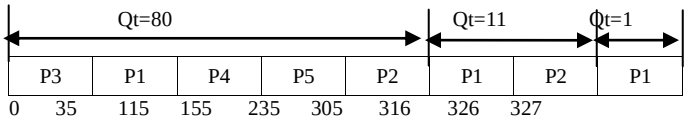

Figure. 4.6.8: Gantt chart for DQRRR in table 4.6.5

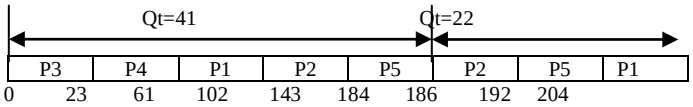

Figure. 4.6.9: Gantt chart for AMDRR in Table 4.6.5

CASE 2: without zero arrival time

Increasing Order:

We have considered five processes P1, P2, P3, P4, P5 arriving at time 0, 2,6,6,8 with burst time 28,35,50,82,110 respectively and priority of each process shown in table 4.6.7. Table 4.6.8 shows the comparing result of RR, DQRRR and our proposed algorithm (AMDRR).

Table 4.6.7 .Data in increasing order

No. of process	At	Bt	P	St
P1	0	28	10	21
P2	2	35	2	22
P3	6	50	7	33
P4	6	82	15	55
P5	8	110	5	68

Table 4.6.8 comparison among RR, DQRRR and AMDRR

Algorithms	RR	DQRRR	AMDRR
Qt	25	28, 66, 30,14	21,45,31
CS	14	7	5
awt	139.8	112.2	72.2
atat	199.4	173.2	133.2

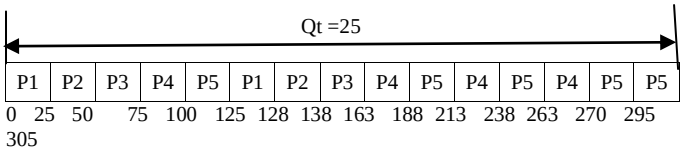

Figure. 4.6.10: Gantt chart for RR in Table 4.6.7

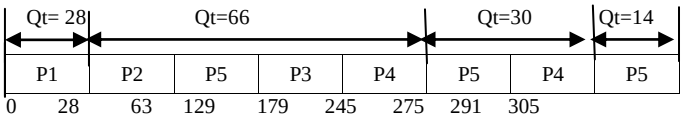

Figure. 4.6.11: Gantt chart for DQRRR in Table 4.6.7

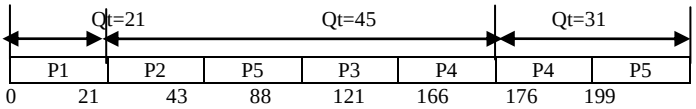

Figure. 4.6.12: Gantt chart for AMDRR in Table 4.6.7

Decreasing Order:

We have considered five processes P1,P2,P3,P4,P5 arriving at time 0,2,3,4,5 with burst time 80,72,65,50,43 respectively and priority of each process shown in table 4.6.9. Table 4.6.10 shows the comparing result of RR, DQRRR and our proposed algorithm (AMDRR).

Table 4.6.9.Data in decreasing order

No. of process	At	Bt	P	St
P1	0	80	12	53
P2	2	72	7	46
P3	3	65	2	40
P4	4	50	15	36
P5	5	43	6	26

Table 4.6.10 comparison among RR, DQRRR and AMDRR

Algorithms	RR	DQRRR	AMDRR
Qt	25	80,57,11,4	53,39,22
CS	13	7	5
awt	216.8	147.8	96
atat	280.2	209.2	158

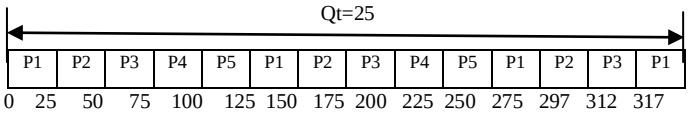

Figure. 4.6.13: Gantt chart for RR in Table 4.6.9

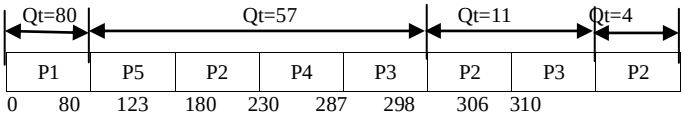

Figure. 4.6.14: Gantt chart for DQRRR in Table 4.6.9

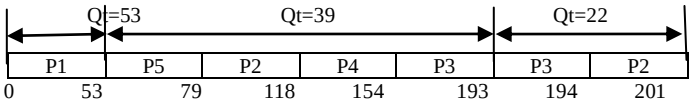

Figure. 4.6.15: Gantt chart for AMDRR in Table 4.6.9

Random Order:

We have considered five processes P1,P2,P3,P4,P5 arriving at time 0,1,2,5,7 with burst time 26,82,70,31,40 respectively and priority of each process shown in table 4.6.11. Table 4.6.12 shows the comparing result of RR, DQRRR and our proposed algorithm (AMDRR).

Table 4.6.11 .Data in random order

No. of process	At	Bt	P	St
P1	0	26	2	16
P2	1	82	7	52
P3	2	70	5	44
P4	5	31	4	20
P5	7	40	11	28

Table 4.6.12 comparison among RR, DQRRR and AMDRR

Algorithms	RR	DQRRR	AMDRR
Qt	25	26,55,21,6	16,35,24
CS	12	7	5
awt	149.4	95.6	58.8
atat	199.2	145.4	108.6

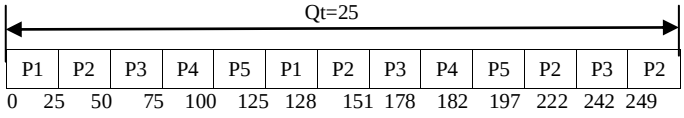

Figure. 4.6.16: Gantt chart for RR in Table 4.6.11

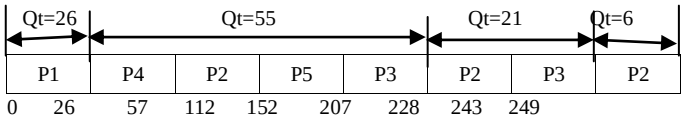

Figure. 4.6.17: Gantt chart for DQRRR in Table 4.6.11

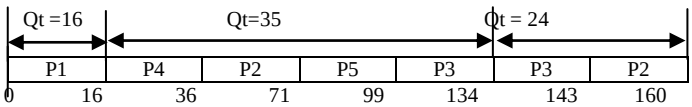

Figure. 4.6.18: Gantt chart for AMDRR in Table 4.6.11

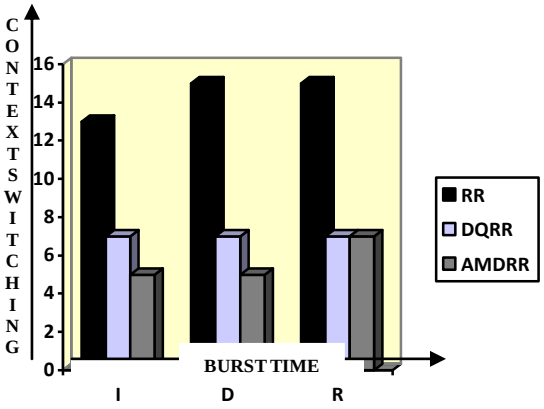

Figure 4.6.19 Context Switching (RR vs. DQRRR vs. AMDRR) with arrival time=0

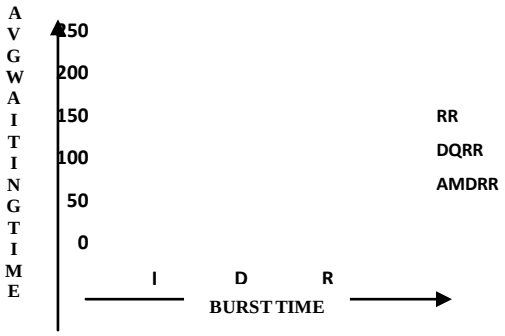

Figure 4.6.20 Avg. waiting time (RR vs. DQRRR vs. AMDRR) with arrival time= 0

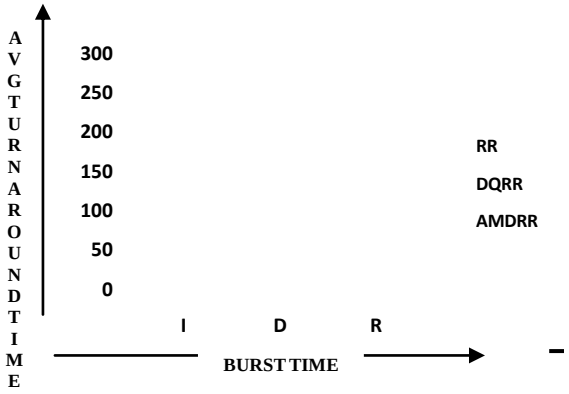

Figure 4.6.21 Average Turnaround time (RR vs. DQRRR vs. AMDRR) with arrival time=0

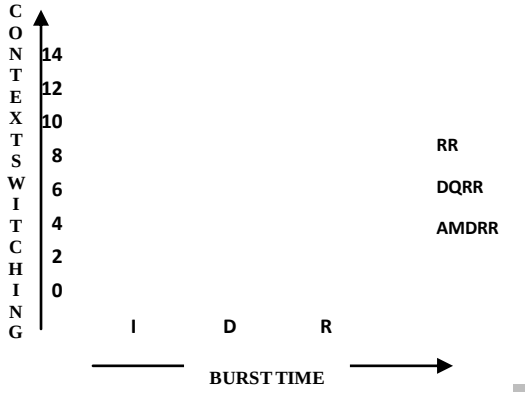

Figure 4.6.22 Context switching (RR vs. DQRRR vs. AMDRR) with arrival time

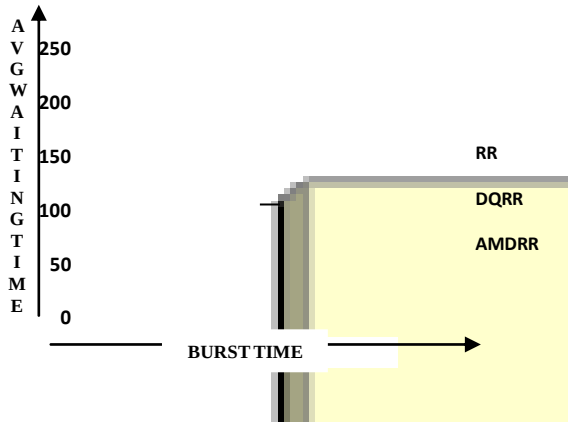

Figure 4.6.23 Average waiting time (RR vs. DQRRR vs. AMDRR) with arrival time

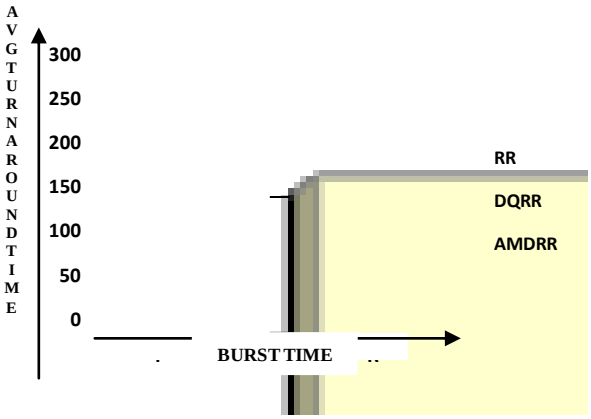

Figure 4.6.19 Average turnaround time (RR vs. DQRRR vs. AMDRR) with arrival time.

CONCLUSION

We have explored the nature of real-time systems in the context of scheduling and it implies on the quality of computation and the behavior of the system. Average turnaround time, average waiting time decreases drastically in the above proposed algorithm. Our proposed algorithm can be further investigated to be useful in providing more and more task oriented result in future.

REFERENCES

[1]. H.S Behera, R.Mohanty, Debashree Nayak "A New Proposed Dynamic Quantum with readjusted Round Robin Scheduling Algorithm and its Performance Analysis", International Journal of Computer Application(0975-8887) volume 5-No.5,August 2010.

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

[3]. Rami J. Matarneh."Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of Now Running Processes", American J. of Applied Sciences 6(10): 1831-1837, 2009.

[4]. Sunita Mohan. "Mixed Scheduling (A New Scheduling Policy)". Proceedings of Insight'09, 25-26 November 2009.

[5]. Helmy, T. and A. Dekdouk, 2007. "Burst Round Robin as a Proportional-share Scheduling Algorithm", IEEEGCC, http://eprints. kfupm.edu. sa/1462/.

[6]. Rashid, M.M. and Z.N. Akhtar, 2006. "A New Multilevel CPU Scheduling Algorithm". J.AppliedSci, 6:2036- 2039. DOI: 10.3923/jas. 2006. 2036. 2039.
[7]. Silberschatz, A., P.B.GalvinandG.Gagne, 2004. "Operating Systems Concepts". 7th Edn., John Wiley and Sons, USA. , ISBN: 13:978-0471694663, pp: 944.
[8]. Tanebun, A.S., 2008, "Modern Operating Systems".3rd Edn. Prentice Hall, ISBN: 13:9780136006633, pp: 1104.
[9]. BogdanCaprita, Wong Chun Chan, Jason Nieth, Clifford Stein and Haoqiang Zheng. "Group Ratio Round-Robin: O(1) Proportional share Scheduling for Uni-processor and Multiprocessor Systems". In USENIX Annual Technical Conference, 2005.
[10]. Biju K Raveendran, Sundar Bala Subramaniam, S.Gurunarayanan, "Evaluation of Priority Based Realtime Scheduling Algorithms: Choices and Tradeoffs", SAC'08, March 16- 20, 2008, copyright 2008 ACM 978-1-59593-753-7/08/003.

Short Bio Data for the Author

 Prof H.S Behera is currently working as a Senior Lecturer in Dept. of Computer Science and Engineering is Veer Surendra Sai University of Technology (VSSUT), Burla, Orissa, India. His research areas of interest include Operating Systems, Data Mining and Distributed Systems.

 Bijayalaxmi Panda is a Final year B. Tech. student in Dept. of Computer Science and Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Orissa, India.

 Sreelipa Curtis is a Final year B. Tech. student in Dept. of Computer Science and Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Orissa, India.