An Improved Priority Dynamic Quantum Time Round-Robin Scheduling Algorithm

Nirali A. Patel
PG Student, Information Technology, L.D. College Of Engineering,Ahmedabad,India

ABSTRACT

In real-time embedded systems, scheduling policy is considered one of the main factors that affect their performance. It helps to choose which task should be selected first from ready queue to run. Round Robin (RR) scheduling algorithm is widely used and its performance highly depends on a Quantum size Qt, which is a predefined amount of time assigned by CPU to every task to be executed. However, the performance degrades with respect to an average waiting time (AWT), an average Turn- Around time (ATT) and a number of Context Switches (NCS). This paper presents a new improved dynamic Round Robin scheduling algorithm to reduce the average waiting time, turn- around time and the number of context switches in order to improve the system overall performance. The queue size distribution and average waiting time for a time-shared system using round-robin (RR) scheduling, with and without overhead, are determined. In this study, the incoming processes are prioritized, and dynamic quantum times are assigned depending on the level of priority. With these parameters, RR versus priority dynamic quantum time round robin scheduling algorithm is analyzed to explore the effect of changing the quantum time of processes and determine the optimum context switches, turnaround time, and waiting time.

Keywords—*Round Robin; Priority Dynamic Quantum Time; scheduling algorithms. average waiting time, system performance, turn- around time, context switch, real-time embedded systems*

Introduction

Operating system is the interface between a user and a machine and it has many features to deliver an excellent service to the user. Scheduling is one of that fundamental features and it is responsible about deciding which job is selected and run from ready queue[1]. Scheduling method affects CPU performance since it determines the CPU and resources utilizations[1,2]. The main purpose of scheduling policy is to ensure completely fairness between different tasks in the ready queue, maximizing the throughput, minimizing the average waiting and turn-around times and the overhead occurs from context switches, and makes sure no starvation happens at all. Two schemes of scheduling algorithms exist today which are l. Preemptive algorithms: where a task is blocked by a higher priority process; and 2. Non preemptive algorithms: where the task completes its execution time even if a higher priority process has arrived[1]. Several factors are used to determine whether a scheduling policy is good or not and can be summarized as follows: A. Waiting time: the time between the task becomes available until the first time of its execution; B. CPU Utilization: the percentage of the CPU being busy; C. Turn-around time: the summation of waiting and execution time for each task and D. Fairness: which is dividing the CPU time equally among all available jobs[1,2,3]. Multiple algorithms exist which can be summarized as follows:

1) First-Come-First-Serve (FCFS): a process arrives first is immediately allocated to the CPU. The major disadvantage of this algorithm is that a process with small burst time takes long time waiting to be executed if another process with long burst is chosen first. 2) Shortest Job First (SJF): this approach allocates processes with short bursts first from their ready queue. It is more efficient than FCFS since it minimizes the average waiting time for small jobs. However, processes with long burst time wait longer which cause a starvation for CPU resources. The drawback of this method is that it requires advance knowledge about CPU burst time which is impractical and difficult in most cases. 3) Round Robin (RR): each process gets its tum to be executed in a fairly time slicing; this concept is known as Time Quantum and it is fixed for all processes. When the quantum time for any process expires, it is temporarily blocked and placed at the end of the ready queue. This procedure is applied on all available tasks until no more tasks exist in the ready queue. This algorithm's efficiency depends totally on the quantum time; if it is small amount then frequent context switch occurs which causes too much of overhead. On the other hand, too long quantum time increases the average waiting and tum-around times. 4) Earliest Deadline First (EDF): a process with shortest deadline time gets its tum first since it has a highest priority among all other processes. This algorithm is considered optimal can be used for both types of tasks (periodic and aperiodic). 5) Multilevel Feedback Queues (MFQ): a process moves between different queues; this action is characterized by the CPU burst time (8t). If the process requires too much time then it moves to a

queue where lower priorities processes are placed. However, if it waits long time then it moves to the queue of the higher priorities processes to prevent starvation from occurring[3,4,5,6].
In an operating system, many processes compete for the services provided by a central processing unit (CPU). The scheduling algorithm of a computer should distribute "bursts" of computer time among these processes such that the cycle time of a process is inversely related to its level of priority or importance while a reasonable cycle time is maintained for all machine processes. This type of systems may be studied using classical queueing theory. The processes of the system are the customers, and the CPU is the server. The processing time is the service length in the queueing system. Few studies have intensively analyzed round-robin (RR) scheduling with another algorithm, and no recent investigations in this area have been conducted. Several related studies will be discussed in this section. Reference [7] presented an expression using RR queue size and average waiting time for M/M/1 to assume a fixed switching time overhead for every slice time. This expression determined the quantum size by studying a cost measure based on specific priorities of processes to decrease the process service time. Meanwhile, a sharing queueing model of multithreading web server was demonstrated by [8]. Multiple users were used in this experiment, and a single server, including a group of economic and flexible servers (Apache), and a single-speed router were
used to verify and accomplish the model. Reference [9] presented an early analysis of the join-shortest-queue for farms with shared processor by using the single queue that was isolated from other queues. The arrival rate of the single queue was dependent on the number of processes at the same queue. Then, the impact of the other queues was described using the conditional arrival rates. In [10] the performance of RR was investigated by considering the process switching overhead such that an incoming process waited for all precedent processes to arrive to obtain time slices before it was assigned its queue time.

I. ROUND-ROBIN SCHEDULING (RR)

The RR architecture is a pre-emptive version of the firstcome first-served scheduling algorithm. The tasks are arranged in the ready queue in a first-come first-served manner, and the processor executes the task from the ready queue on the basis of time slice. If the time slice ends and tasks are still being executed by the processor, the scheduler will forcibly pre-empt the executing task and keep it at the end of the ready queue. Then, the scheduler will allocate the processor to the next task in the ready queue. The pre-empted task will make its way to the beginning of the ready list and will be executed by the processor from the point of interruption.
A scheduler requires a tick timer and a time management function to implement the RR architecture . The time slice is proportional to the period of clock ticks . The time slice length is critical in a real-time operating system (ROTS). The time slice should not be too small, which results in frequent context switches, and should be slightly greater than the average task computation time.

II. PRIORITY DYNAMIC QUANTUM TIME ROUND-ROBIN (PDQT)

The RR scheduling algorithm does not prioritize and has fixed quantum time, which makes this algorithm unsuitable for an RTOS. The RR is limited by high context switch, high waiting and turnaround times, and low throughput. Thus, the RR scheduling algorithm is not the optimal choice for an RTOS. Priority RR scheduling is limited by starvation, in which the least priority thread with fixed quantum time will be starved and pre-empted by the highest priority thread. Thus, we propose an algorithm called priority dynamic quantum time round robin scheduling algorithm (PDQT) that depends on the existing RR. The basic principle of PDQT involves different priority levels and quantum times . PDQT is performed as follows:
a) Priorities are set for the processes that enter the ready queue.
b) The new quantum time is calculated depending on the old one by using a simple formula, q=k+n-1, where q is the new quantum time, k is the old quantum time, and n is the priority of the processes in the ready queue.
c) Different quantum times are set for the processes on the basis of the level of priority. The highest priority process will obtain the largest quantum time, q, and the lowest priority process will be assigned the smallest quantum time, k.
d) The process in between will be assigned a quantum time that is 1 lower than the time of the process before it.
e) The original RR is applied with the levels of priority and new different quantum times.
f) The context switches, average turnaround time, and average waiting time are calculated. The existing RR is improved by reducing the context switches, as well as by reducing the waiting and turnaround times, thereby increasing throughput. The next sections present case studies to show the differences between RR and PDQT.

III. ANALYSIS WITH QUEUEING THEORY

We built a mathematical model for our case study to illustrate the need for the processes to join the ready queue. The model also shows the rules by which the processes are allowed into the processor and the time required for execution. Queueing theory covers all these aspects to construct the model that is suitable to build all observable models that include all features of a queue. In our case study, the unit requesting services are the processes, the queue is the line of processes, and the server is the CPU. Queue indicates a waiting line or the formation of a line while waiting for something. This process involves arriving items that wait to be served at the facility that provides the service being sought .

IV. Case study

Five processes have been defined with CPU burst time, arrival times, and their priorities. These five processes are scheduled in RR technique as well as according to the PDQT algorithm. The context switch, average waiting time, and average turnaround time are calculated, the results are compared. We consider five processes (A, B, C, D, and E) with different arrival times, but two processes, namely, B and C, are scheduled to arrive at the same time. Thus, the priority of the process plays an important role in the execution of the processes.

Task	Arrival Time	Burst Time	Priority
A	0	12	2
B	5	8	4
C	5	4	5
D	10	10	3
E	15	6	1

Table-1:The Inputs For the threads of case study

Here, the process with the highest priority should proceed first whenever it arrives at the same time with another process with lower priority. Table 1 shows the inputs of the processes, and Fig.1 shows the diagram of the case study. Figs. 2 and 3 illustrate the Gantt chart of the process time slicing in RR and PDQT, respectively. The quantum time is 3millisecond.

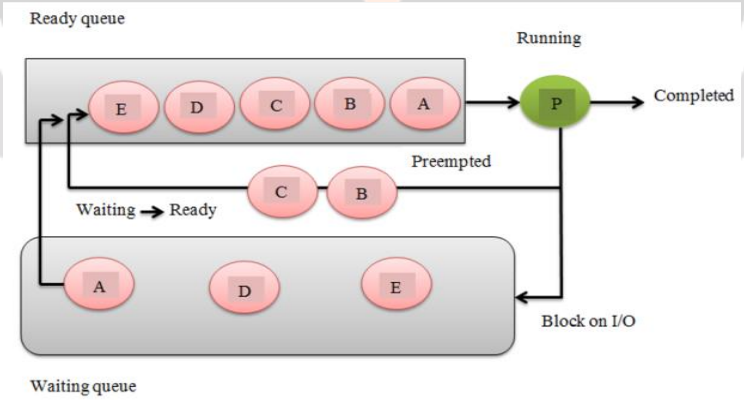

Fig-1:Diagram Of Case Study

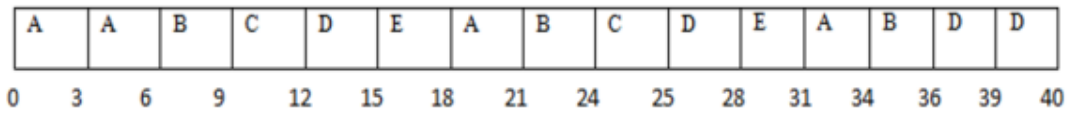

Fig-2:Gantt chart of process time slicing in simple Round-Robin Architecture

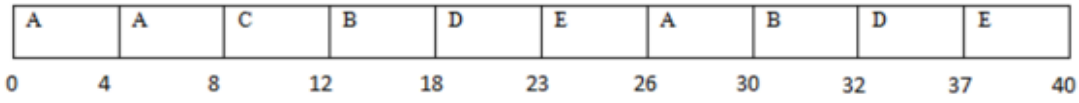

Fig-3:Gantt chart of process time slicing in simple PDQT Architecture

In accordance with the original RR, simple RR does not prioritize. Hence, five processes have been scheduled using simple RR architecture, with a time slice of 3 millisecond. In RR algorithm, no process is allocated in the CPU for more than one time slice in a row. If the CPU process exceeds onetime slice, the concerned process will be pre-empted and placed into the ready queue. The process is pre-empted after the first quantum time, and the CPU is given the next process that is in the ready queue (process B). A similar process is conducted for the schedule until the first cycle is completed. In the second cycle, the same method is used to schedule the processes. The PDQT algorithm uses process priorities and different quantum times depending on the level of priority; thus, each process obtains a unique quantum time. The process with the highest priority obtains the largest quantum time, the lowest priority process obtains the smallest quantum time, and the processes in between obtain quantum time 1 less than the one before. By using RR, we obtained 26.2 milliseconds for average turnaround time and 18.2 milliseconds for average waiting time, and the context switch is 14. By applying PDQT, we got 23.8 and 15.2 milliseconds for average turnaround time and average waiting time, respectively, and the context switch is 9. The two algorithms are applied with the given arrival times, burst times, and priorities, where the processes enter the ready queue until the execution is completed before they leave the system. We analyzed the times of these processes in the ready queue and the system. Tables 2 and 3 illustrate the analysis of the RR and PDQT processing times, respectively. The tables contain 9 or 10 columns. The first one is the Task that enters to the ready queue, second, third and fourth are arrival, burst and quantum times respectively. The fifth column is the moment that the process start processing inside the CPU, followed by the end of time service (but not the processing). Next column contains one value either 1 or 0, where 1 for the process which out of quantum time but still in service, however, 0 for the processes complete execution. Waiting time column to calculate the waiting time for each process in every round. The last column to announce that the processing for the process has been completed. An additional column added to table 3 for PDQT which is the fourth one is the priority of the process.

Task	A.T.	B.T.	Q.T.	Start	End	Time in Q	W.T.	Finishing
A	0	12	3	0	3	0	0	
A	-	9	3	3	6	1	0	
B	5	8	3	6	9	1	1	
C	5	4	3	9	12	1	4	
D	10	10	3	12	15	1	2	
E	15	6	3	15	18	1	0	
A	-	6	3	18	21	1	12	
B	-	5	3	21	24	1	12	
C	-	1	3	24	25	0	12	Comp.
D	-	7	3	25	28	1	10	
E	-	3	3	28	31	0	10	Comp.
A	-	3	3	31	34	0	10	Comp.
B	-	2	3	34	36	0	10	Comp.
D	-	4	3	36	39	0	8	
D	-	1	3	39	40	0	0	Comp.
Total	35		40			8	91	

Table-2:Analysis times of simple Round-Robin

Task	A. T.	B. T.	Priority	Q. T.	Start	End	Time in Q	W .T.	Finishing
A	0	12	4	4	0	4	0	0	
A	-	8	4	4	4	8	1	0	
C	5	4	1	7	8	12	0	3	Comp.
B	5	8	2	6	12	18	1	7	
D	10	10	3	5	18	23	1	8	
E	15	6	5	3	23	26	1	8	
A	-	4	4	4	26	30	0	18	Comp.
B	-	2	2	6	30	32	0	12	Comp.
D	-	5	3	5	32	37	0	9	Comp.
E	-	3	5	3	37	40	0	11	Comp.
Total	35						4	76	

Table-3:Analysis Times Of PDQT

Factors	RR	PDQT
n	5	5
T	131	114
Avg. T	26.2	23.8
W	91	76
Avg. W	18.2	15.2
nw	8	4
P	1.6	0.8
I	0	0
To	40	40
C_{active}	100%	100%
ג	0.4	0.4
μ	1.5	1.1
$C_{capacity}$	0.3	0.4
S	157	151
Avg. S	31.4	30.2
A	35	35
Avg. A	7	7
Avg. $_{stay}$	8	8
P1	0.96n	0.16n
M	2.3	4

Table-4:Result of Analysis for two RR vs PDQT

In the table above, nw for RR is 8 while in PDQT is 4. That mean the number of processes in RR that had to repeat their cycle to enter again to the ready queue and waiting for processing is double than in PDQT. Thus, the probability of the processes to wait in the ready queue with PDQT less than in RR. Because the total time for service the two algorithms is the same and there is no idle time in CPU (i.e. CPU always bust), so the activity of CPU is 100% for the both algorithms. Moreover, the capacity of CPU with PDQT more than RR by about 0.1. Processes in RR take more time for service and average time for service than in PDQT where it is 157, 31.4 and 151, 30.2 for RR and PDQT respectively. The probability the n number of processes to stay in CPU is 0.96 in RR and 0.16 in PDQT while the mean number of processes in CPU with PDQT is greater than with RR. From

all these results we conclude that by setting priorities and changing the quantum time from fixed to dynamic will give RR more flexibility to execute more processes with less time and high throughput.

V. COMPARISON WITH EXISTING RR

The performance of the two algorithms are compared by considering some of the results with different quantum times. Figs. 3 to 8 show the curves the equations indicating six factors (average turnaround time, average waiting time, probability, CPU capacity, probability of n processes in the CPU, and mean number of processes in CPU) and four quantum times (2, 3, 5, and 6). Results show that the proposed algorithm performs better than existing RR for dynamic quantum time. The PDQT behaves better than RR in most cases under similar values of quantum time.

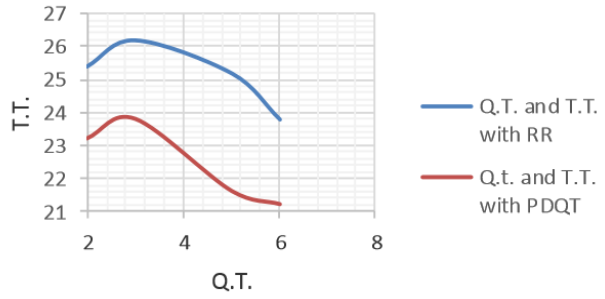

Chart-1:Performance of RR and PDQT for quantum time and turnaround time

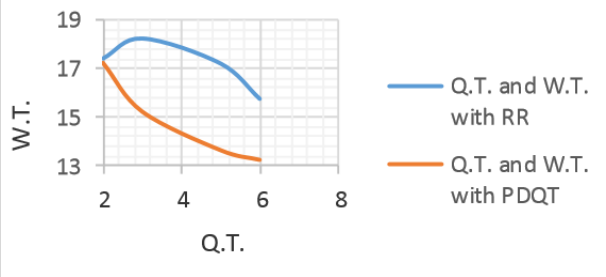

Chart-2:Performance of RR and PDQT for quantum time and waiting time

We can see from the curves above that our algorithm behave better than the existing RR in most of cases in term of all the factors mentioned earlier. The peak of the two algorithms arrived to top with quantum time 3 according to turnaround time but the lowest value with quantum time 6. Moreover, the highest value of average waiting time with RR when the quantum is 3 but 2 according to PDQT.

VI. CONCLUSION

We mathematically analyze the results of simple RR and proposed PDQT on the basis of queueing theory. PDQT performs more efficiently than RR in most cases under similar quantum times. This algorithm also has lower turnaround and waiting times. Thus, the operating system overhead is reduced, and throughput is increased. Starvation is also reduced because the processes with the highest priorities are assigned with the largest quantum time and are executed prior to the lower priority processes. The performance of time-shared systems can be improved with the proposed algorithm and can also be modified to enhance the performance of a real-time system.

REFERENCES

[1] A. R. S. K. Sahu and S. K. Samantra, "An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum", International Journal of Computer Science, Engineering and Information Technology (lJCSEIT),Vol. 5,No. I, February 2015.

[2] M. S. Iraji, "Time Sharing Algorithm with Dynamic Weighted Harmonic Round Robin", Journal of Asian Scientific Research,
Vol. 5,No. 3,pp. 131-142,2015

[3] D. Maste,L. Ragha and N. Marathe,"Intelligent Dynamic Time Quantum Allocation in MLFQ Scheduling",International Journal of Information and Computation Technology, ISSN 0974-2239, Vol. 3, No. 4, pp. 311-322, 2013. International Research Publications House.
[4] A. Noon,A. Kalakech and S. Kadry,"A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average", International Journal of Computer Science Issues (T.TCSI),Vol. 3,Issue 3,No. 1,pp. 224-229,May 2011.
[5] 1. S. Rajput and D. Gupta,"A Priority Based Round Robin CPU Scheduling Algorithm for Real Time Systems", International Journal of Innovations in Engineering and Technology (TJIET), Vol. 1,Issue 3,pp. 1-11,October 2012.
[6] H. S. Behera, R. Mohanty and D. Nayek, "A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis", International Journal of Computer Applications (0097 - 8887),Vol. 5,No. 5, pp. lO IS, August 201O.
[7] P. J. Rasch, "A queueing theory study of round-robin scheduling of time-shared computer systems," Journal of the ACM (JACM), vol. 17, pp. 131-145, 1970.
[8] J. Cao, M. Andersson, C. Nyberg, and M. Kihl, "Web server performance modeling using an M/G/1/K* PS queue," in Telecommunications, 2003. ICT 2003. 10th International Conference on, 2003, pp. 15011506.
[9] V. Gupta, M. H. Balter, K. Sigman, and W. Whitt, "Analysis of join-the-shortest-queue routing for web server farms," Performance Evaluation, vol. 64, pp. 1062-1081, 2007.
[10] V. Gupta, "Finding the optimal quantum size: Sensitivity analysis of the M/G/1 round-robin queue," ACM SIGMETRICS Performance Evaluation Review, vol. 36, pp. 104-106, 2008.