Assesment of Round Robin with Priority Scheduling Algorithm

Maria Beatriz E. Ang, Cindy Mae C. Chua,
Jacinto Maurico A. Fausto
Ateneo De Manila University
Quezon City, Philippines

***Abstract*-The purpose of this study is to assess the Priority based Round Robin CPU scheduling Algorithm for Real Time systems by Ishwari Singh Rajput and Deepa Gupta. The Algorithm proved to be more efficient than that of the standard round robin, despite difficulties in its implementation lying in updating and assigning priorities per Round Robin, and an issue regarding the sequence of the final step. The efficient was tested by running it through matlab and a console-based software that tests its runtime. The results returned less response time and faster turnaround time making it more ideal for real-time systems.**

I. INTRODUCTION

The proposed CPU scheduling algorithm proposed by Ishwari Singh Rajput and Deepa Gupta is entitled "A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems." In a sense, it is not a new discovery or invention of an algorithm, but more of a novel way to utilize and combine currently existing algorithms, as the title implies, to make them more efficient. Essentially, it's the improvement of the Round Robin algorithm, through combining it with the priority scheduling algorithm to make scheduling more efficient by reducing the waiting and response times, usually from short CPU bursts, and making it more suitable for hard real time systems. This was created due to the limitations of each algorithm alone. The Round Robin algorithm's primary limitation was that of constant context switching making it inefficient with short bursts, and the tendency of Priority Scheduling to cause starvation of less prioritize processes.

II. DEMONSTRATION OF PROPOSED ALGORITHM

There are two (2) phases to the proposed algorithm, as stated in the article. It will behave exactly like a round robin implementation, except for the addition of a priority that dictates the turn of each quantum time given. The following are steps that demonstrate the implementation of this algorithm, with guiding illustrations to show the step-by-step progression.

1. The system runs a round robin in according to a given priority and quantum time, instead of the default First Come First Serve algorithms commonly used in standard Round Robin algorithms.

1st Round Robin		
Process	**CPU Burst (ms)**	**Priority**
P1	22	4
P2	18	2
P3	9	1
P4	10	3
P5	4	5
Quantum Time = 5 ms		

Following the authors of the algorithm, the first set of priorities are predetermined, and are not based on any of the remaining CPU bursts yet.

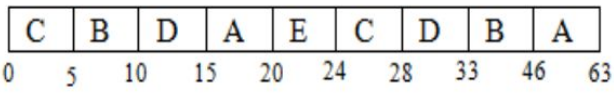
2. After the first round robin has been executed, the priorities will change. The highest priority will be given to the process with the shortest remaining CPU burst (much like the Shortest Job First algorithm).

2nd Round Robin		
Process	**Remaining CPU Burst (ms)**	**New Priority**
P1	17	4
P2	13	3
P3	4	1
P4	5	2
P5	0	x
Quantum Time = 5 ms		

3. The third round shows that there are only two processes left. The authors did not schedule this in round robin form. Instead, the authors finished each process (starting with P2, then P1) without following the given quantum time.

3rd Round Robin		
Process	**Remaining CPU Burst (ms)**	**New Priority**
P1	13	2
P2	8	1
P3	0	x
P4	0	x
P5	0	x
Quantum Time = 5 ms		

The authors of the algorithm summarized in the runtimes in a simple Gantt chart (Gantt Chart taken from their version of the implementation):



Where A is P1, B is P2, C is P3, D is P4, and E is P5. This illustrates that P1 and P2 (A and B) are the last two processes, and did not follow the given quantum time in Round Robin Implementation.

The second phase, as the authors stated, no longer followed the quantum time. This will be discussed further in the paper.

In summary, there two (2) steps in the implementation. First is to implement the standard round-robin using priority as order, instead of FCFS (First Come First Serve). Second, after the first round, reassign the priorities, in ascending order, depending on the remaining CPU burst. The second round will no longer follow the assigned quantum time of the Round Robin implementation.

III. DIFFICULTIES AND ISSUES IN PROPOSED ALGORITHM

There are two major difficulties in implementing this algorithm.

The first problem that we encounter is that of the priority reassignment. Given that the case study uses a deterministic model, it could not be said for sure what the remaining CPU burst of each process will be. The algorithm proved to be faster, and more efficient, given a set of given processes and bursts, but given the unpredictability of a real time system (that which the algorithm was tailored for), the second round of reassigned priorities could not be determined accurately.

The second issue is the second round. Specifically, the abandoning of the quantum times. The quantum time was not followed in the test case given by the authors, and it showed a significant improvement in turnaround and response time. However, once again, this was based on the test case of a deterministic model, and was tested only twice (on the same data). The main issue is that falling back into priority without regard to the round robin algorithm in the second round, could once again cause starvation resulting from the lack of turns.

In the second round of the first test (as seen in the picture of the implementation) The two of the four remaining processes were conveniently within the quantum time (P3 and P4), and the other two finished their remaining long rounds (P1 and P2). However, if P3 and P4 had roughly the same remaining CPU burst as P1 and P2, the lack of the round robin implementation in the second round will no longer have a more efficient waiting time, and will cause starvation on the processes that will run last.

The group recommends that the test be implemented more than once, in multiple simulations and deterministic scenarios. Given that this, as said, will have to deal with the demanding and unpredictable nature of real-time systems (more importantly that of hard-real time), the algorithm must be conducted in multiple environments, especially to test the efficiency of foregoing the quantum time in the second round. The average turnaround times and waiting times must be measured then. In addition to testing the second portion, testing it in a simulation (or even in implementation) would also give insight to the first issue of predicting the reassignment of priorities.

Furthermore, the algorithm must also be tested without the rule of the second round and the continuation of the first. If the algorithm was continued after the first (where the priorities will keep being reassigned) - the problem of starvation brought up in the second issue which is seen in longer running processes could potentially be solved.

From the first Gantt Chart provided by the authors, the group presents the case of their recommendation in this new Gantt Chart (based off the data of the first):

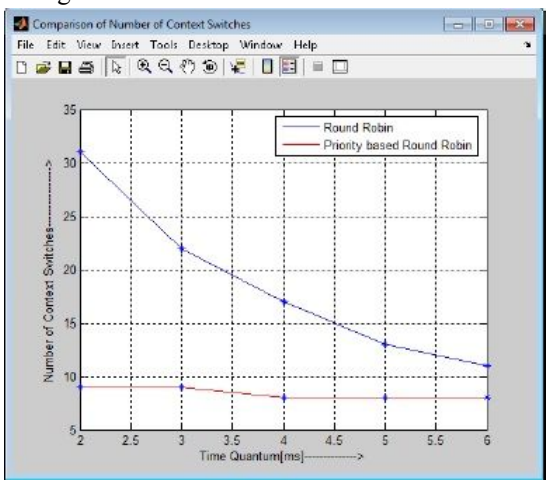
Process	P3	P2	P4	P1	P5	P3	P4	P2	P1	P2	P1	P2	P1
CPU Burst	0-4	5-9	10-14	15-19	20-23	24-27	28-32	33-37	38-42	43-47	48-52	53-55	56-62

IV. APPLICATION IN UNI- AND MULTIPROCESSOR THREADS

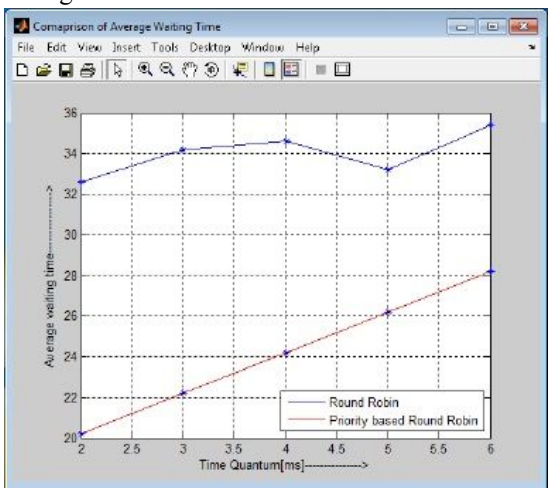
V. APPROACHES USED TO TEST PERFORMANCE OF ALGORITHM

To test the efficiency of the proposed algorithm, the researchers took the following into account: number of context switches, average waiting time, and average turnaround time. The following graphs were plotted using the MATLAB 7.0.

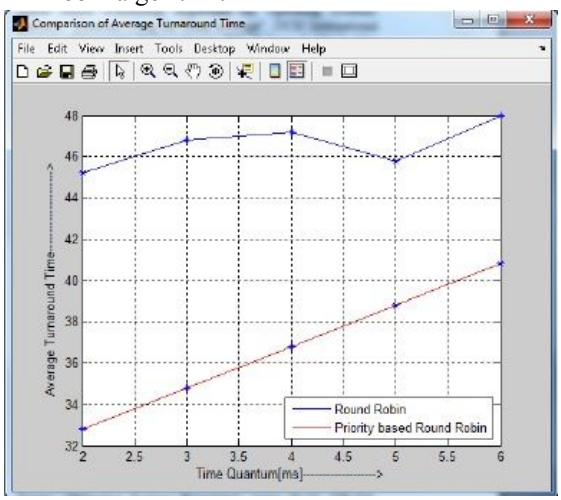
1. The number of context switches was proven to be lesser compared to the simple Round Robin algorithm.



2. The average waiting time for the proposed algorithm is lesser compared to the simple Round Robin algorithm.



3. The average turnaround time for the proposed algorithm is lesser compared to the simple Round Robin algorithm.



VI. PERFORMANCE OF ALGORITHM IN COMPARISON TO CLASSICAL ONES

In comparison to the proposed algorithm, the simple Round Robin algorithm lacks in the three aspects. There seems to be a larger waiting time and response time as the processes spend more time in the ready queue, thus affecting its total waiting, response, and turnaround time. As much as possible, we would like to spend lesser time executing these processes in order to avoid starvation. The number of context switches were also found to be larger compared to the proposed algorithm, and this could be a disadvantage as it results to more time and memory being wasted, leading to a scheduler overhead. Lastly, the throughput of the simple Round Robin is lower compared to the proposed algorithm. The fewer the context switches, the higher the throughput will be therefore having a larger amount of context switches puts the simple Round Robin algorithm into a disadvantage compared to the proposed algorithm.

When it comes to the Priority algorithm, time wasn't much of a problem as to compared to the simple Round Robin algorithm. Instead, starvation might occur to the lower priority processes.

If taken individually, these two classical algorithms may come off as inefficient when working with real time systems. The proposed algorithm, however, combines the strengths of both and eliminates their disadvantages. By taking the simple Round Robin algorithm's ability to prevent starvation and the Priority algorithm's time management, the researchers were able to propose a new algorithm that performs better and is a more effective algorithm for real time systems.

Approaches Used to Test Performance of Algorithm (CINDY - The paper's last few pages contains their matlab graphs that proves its efficiency. You can just refer to that)

Performance of Algorithm in Comparison to Classical Ones (CINDY - Just compare it to Round Robin ONLY and Priority ONLY - which, luckily was also shown in the first few pages of the paper)

Advantages and Disadvantages (BEA - You need to refer to CIndy's second question to outline the advantages, but you will have to deduce its disadvantages. Just PM me if you're stuck regarding this :))

Application (BEA- Just name any one system or application that is MIS related that this could be implemented in, and how it could possible improve it)