INTELLIGENT TIME SLICE FOR ROUND ROBIN IN REAL TIME OPERATING SYSTEMS

Yaashuwanth .C & R. Ramesh

Department of Electrical and Electronics Engineering,
Anna University Chennai, Chennai 600 025

ABSTRACT

The main objective of this paper is to develop a new scheduling algorithm for scheduling tasks in real time operating systems. Scheduling algorithms play a significant role in the design of real time embedded systems. Simple round robin architecture cannot be implemented in real time operating systems because of high context switch rate, large waiting time and larger response time. Missing deadlines will degrade the system performance in real time embedded systems. The proposed algorithm modifies all the drawbacks of simple round robin architecture, the proposed architecture calculates the time slice for tasks and exclusively allocates the time slice for every individual tasks. A comparison with round robin architecture to the proposed architecture has been made. It is observed that the proposed architecture solves the problems encountered in simple round robin architecture in real time operating systems by decreasing the number of context switches waiting time and response time thereby increasing the system throughput.

Key words: *Round robin architecture, time slice, deadlines, Real time operating systems.*

1. INTRODUCTION

A real-time operating system (RTOS) is a multitasking operating system intended for real-time embedded applications. RTOS facilitates the creation of a real-time system, but does not guarantee the final result will be real-time; this requires correct development of the software. An RTOS does not necessarily have high throughput; rather, an RTOS provides facilities which, if used properly, guarantee deadlines can be met generally or deterministically. An RTOS will typically use specialized scheduling algorithms in order to provide the real-time developer with the tools necessary to produce deterministic behavior in the final system. An RTOS is valued more for how quickly and/or predictably it can respond to a particular event than for the amount of work it can perform over a given period of time. Key factors in an RTOS are therefore a minimal interrupt latency and a minimal thread switching latency

The intelligent time slice for round robin architecture for real time operating systems is a modified version of simple round robin architecture. Simple round robin architecture cannot be implemented in real time operating systems because of high context switch rate, larger waiting time and larger response time. Because of these performance criteria of the round robin architecture is not suitable to implement in real time systems. Real time operating systems have no hard deadlines for tasks but missing of deadlines in real time operating systems will degrade the system performance. The proposed algorithm covers all the drawbacks of round robin architecture by reducing the number of context switches, reducing the waiting time, reducing the response time thereby increasing the system throughput .

Real time systems always have a time constraint on computation. Each task should be invoked after the ready time and must complete before its deadline (Enrico Bini and Giorgio C. Buttazzo 2004; Zhi Quan and Jong-Moon Chang 2003; Houssine Chetto and Maryline Chetto 1989). An attempt has been made to satisfy the constraints in many scheduling algorithms both non preemptive and preemptive (Barbara Korousic –Seljak 1994).. Richard roehi proposed a new way of scheduling which implements a new priority queue in the round robin architecture that gives priority to tasks with short central processing unit(CPU) burst thereby improving the performance of the tasks with less CPU burst (Richard roehi 1995). Fair scheduling with tunable latency (Klaus H. Ecker 1996) is a round robin approach that proposes an alternative and lower complexity approach to packet scheduling, based on modifications of the classical round robin scheduler. The authors showed that appropriate modifications of the Weighted Round Robin service discipline can, in fact, provide tight fairness properties and

efficient delay guarantees to multiple sessions. Round robin approach has its applications on networks by allowing the network devices to have a free share of network resources. Various types of scheduling algorithms such as Dificit Round Robin (Shreedhar. M , George Varghese 1996), Dificit Round Robin Alternated (Marco.J, et.al). and credit round robin(Viet.L.Do and Kenneth Y.Yun 2003) have been implemented.

Real-time scheduling theory has shown a transition from cyclical executive based infrastructure to more flexible scheduling models such as fixed-priority scheduling, dynamic-priority scheduling, feedback scheduling or extended scheduling (Sha, L.et al 2004). In fact, recent studies shows that almost every existing real-time operating system provides only POSIX-compliant fixed-priority scheduling (Ripoll I. et.al. 2003] since it can be easily implemented in commercial kernels. The author(Barbara Korousic –Seljak 1994) has discussed the basic architecture of fixed priority scheduling which implies that there will be single server and the jobs will be allocated based on the priority given by the user. The aurthor (Burns.A 1994) proved that standard analysis of fixed priority assumes that all the computations of each task must be completed within task deadlines but in practice this is not the case, deadlines is most currently associated with last observed event of the task. The internal events and kernel overheads can occur after the deadlines. Reindir and Pieter revealed the worst case response time analysis of real time tasks using hierchial fixed priority scheduling (Reindir J. bril and Pieter J.L. Cuijpers 2006) .Using an example which consist of single server the author portrays that the existing worst case response time analysis can be improved

With these observations it is found that these existing architecture are not suitable to be implemented with a round robin manner in Real Time Operating system . The proposed algorithm is a modified version of round robin algorithm with intelligent time slicing concept.

2. ROUND ROBIN ARCHITECTURE

Round robin architecture is a pre emptive version of first come first serve scheduling algorithm. The tasks are arranged in the ready queue in first come first serve manner and the processor executes the task from the ready queue based on time slice. If the time slice ends and the task is still executing on the processor the scheduler will forcibly pre-empt the executing task and keeps it at the end of ready queue then the scheduler will allocate the processor to the next task in the ready queue. The preempted 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 time management function to implement the round robin architecture and requires a tick timer. The time slice is proportional to the period of clock ticks. The time slice length is critical issue in real time operating systems. The time slice must not be too small which results in frequent context switches and should be slightly greater than average task computation time

2.1 Round Robin Drawbacks in Real Time Systems

Round robin when implemented in real time operating systems faces two drawbacks they are high rate of context switch and low throughput. These two problems of round robin architecture are interrelated.

Context switch :When the time slice of the task ends and the task is still executing on the processor the scheduler forcibly pre empts the tasks on the processor and stores the task context in stack or registers and allocates the processor to the next task in the ready queue. This action which is performed by the scheduler is called as context switch. Context switch leads to the wastage of time, memory and leads to scheduler overhead

Larger waiting time and Response time : In round robin architecture the time the process spends in the ready queue waiting for the processor to get executed is known as waiting time and the time the process completes its job and exits from the taskset is called as turn around time. Larger waiting and response times are clearly a drawback in round robin architecture as it leads to degradation of system performance

Low throughput: Throughput is defined as number of process completed per time unit. If round robin is implemented in real time operating systems throughput will be low which leads to severe degradation of system performance. If the number of context switches is low then the throughput will be high. Context switch and throughput are inversely proportional to each other

3. INTELLIGENT TIME SLICE FOR ROUND ROBIN IN REAL TIME OPERATING SYSTEMS SCHEDULING

The proposed architecture focuses on the drawbacks of simple round robin architecture which are context switch, equal priority to all the tasks Because of these drawbacks round robin architecture is not suitable for real time operating systems.

The proposed architecture eliminates the defects of implementing a simple round robin architecture in real time operating system by introducing a concept called intelligent time slicing which depends on three aspects they are priority, average CPU burst, context switch avoidance time. The proposed architecture allows the user is allowed to assign priority to the system . An assumption is made on average CPU burst which are reasonable to the system. A dedicated small processor used to reduce the burden of the main processor is assigned for calculating the time slice. The calculated time slice will be different and independent for each tasks and the tasks are fed into the ready queue and these tasks execute in the main processor with their individual time slices. The proposed architecture can be implemented in real time operating systems because of greater response time and throughput and the users can allocate priority to every individual task. The intelligent time slice are calculated by the dedicated small processor as shown in figure 3.1

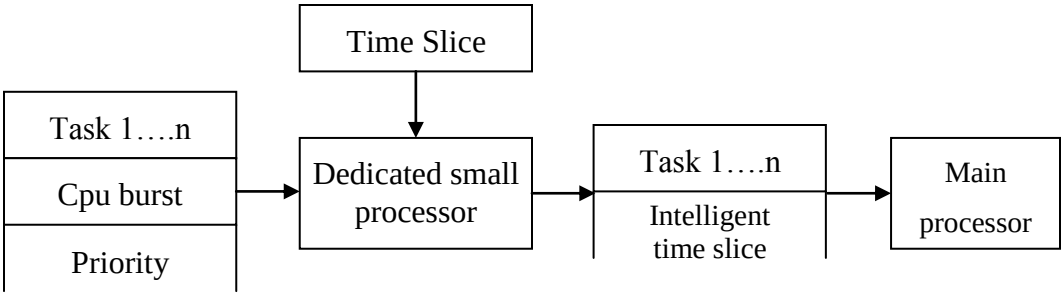

Figure 3.1 Intelligent time slice generation.

The priority, average CPU burst and context switch avoidance time along with original time slice is given to the small dedicated processor which calculates the new time slice dedicated to the corresponding tasks. The output of small dedicated processor will be the task id no, CPU burst and the calculated time slice these criteria are given to the main processor. These dedicated time slice are different and are exclusively allocated for each tasks and the tasks execute on the processor based on these time slicing. Thus the proposed architecture is superior to existing simple round robin architecture and can be implemented in real time operating systems.

3.1 Intelligent Time Slice Calculation

A new way of intelligent time slice calculation has been proposed which allocates the frame exclusively for each task based on priority, shortest CPU burst time and context switch avoidance time.

Let the original time slice (OTS) is the time slice to be given to any process if it deserves no special consideration.

Intelligent time slice = Original Time Slice(OTS)+ Priority Component (PC)+ Shortness Component for CPU burst time (SC)+ Context Switch Component(CSC)

----------→equation 1

Priority component (PC) is assigned by the user depending upon the priority which is inversely proportional to the priority number (higher the priority, greater the PC).

Shortness component (SC) is assigned inversely proportional to the length of the next CPU burst for the process. Shortest component should be lesser than assumed CPU burst (ATS).

Context Switch Component(CSC) is calculated as follows.

- Calculate the Computed Component (CC): Priority Component (PC) and Shortness Component (SC) are added to the Original Time Slice (OTS).

Computed component (CC) =Priority Component(PC) + Shortness Component (SC) + Original Time Slice (OTS)

- CC is deducted from the Assumed CPU burst (ATS). Let the result be called balance CPU burst.

Balanced CPU burst = Assumed Time Slice(ATS) – Computed Component(CC)

If this balance CPU burst is less than OTS, it will be considered as Context Switch Component (CSC).

4. CASE STUDY

Five processes has been defined (with its priority), these five processes are scheduled in round robin and also in the proposed architecture. The context switch, waiting time, turn around time has been calculated and the results were compared. The process id, burst time and priority are defined as shown in table 4.1

Table 4.1 Input component for the processor

Process ID	CPU burst time (milliseconds)	Priority
1	25	2
2	5	3
3	15	1
4	8	2
5	10	1

4.1 Round robin architecture

The above five processes has been scheduled using simple Round Robin architecture. The time slice of four milliseconds has been used. In round robin algorithm no process is allocated CPU for more than one time slice in a row. If the CPU process exceeds one time slice, the concern process will be preempted and put into the ready queue. The process is preempted after the first time quantum and the CPU is given to the next process which is in the ready queue (process P2), similarly schedules all the process and completes the first cycle In the second cycle process P2 requires one millisecond time slice (doesn't require four milliseconds time slice), so it quits before its time quantum expires. The CPU is given to next process P3. In the same way all other processes in the system are scheduled once each process have received one time slice the CPU is returned to process P1 for additional time slice. The process time slicing in simple Round Robin architecture is shown in figure.4.1

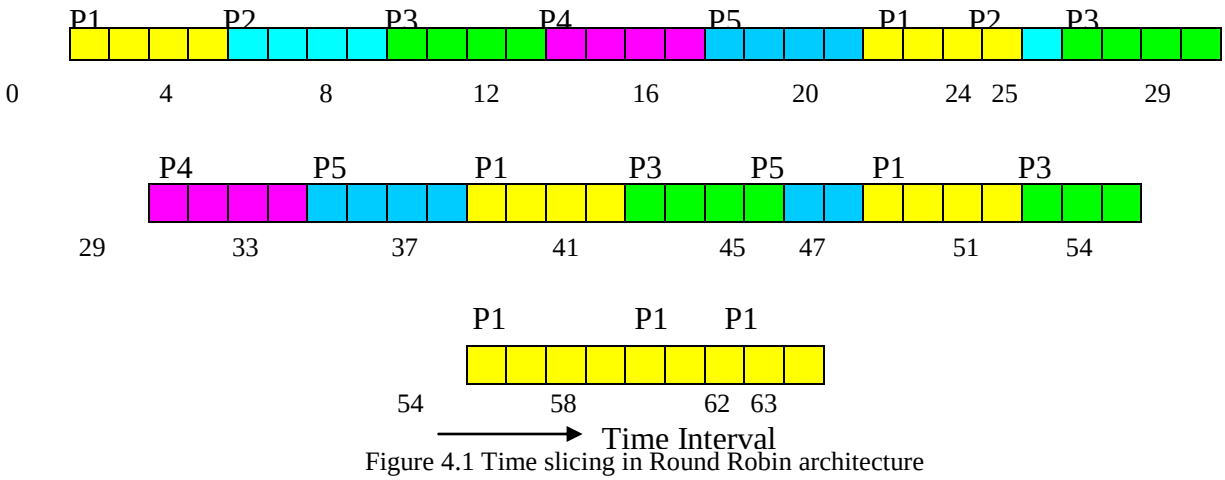

Figure 4.1 Time slicing in Round Robin architecture

4.2 Intelligent time slice for round robin architecture

The above five processes has been scheduled using proposed architecture. Intelligent time slice has been calculated using the equation-1 and the calculated intelligent time slice is shown in table.4.2

Table 4.2 Calculation of intelligent time slice for round robin architecture

Process ID	CPU burst	Calculated time slice (milliseconds)				
		OTS	PC	SC	CSC	Intelligent time slice
1	25	4	0	0	0	4
2	5	4	0	1	0	5
3	15	4	1	0	0	5
4	8	4	0	1	3	8
5	10	4	1	0	0	5

The intelligent time slice of process P1 is same as the original time slice of four milliseconds and time slice of four milliseconds is assigned to process P1. After the execution of four milliseconds time slice the CPU is allocated to process P2. Sine the CPU burst of process P2 is lesser than the assumed CPU burst (ATS), one milliseconds of SC has been included. The process P3 has the highest priority, so priority component is added and the total of five milliseconds is allocated to process P3. The Balanced CPU burst for process P4 is leaser than OTS, context switch component is added and a total of eight millisecond time slice is given to process P4. Process P5 is given a total of five milliseconds with one millisecond of priority component is added to original time slice. After executing a cycle the processor will again be allocated to process P1 for the next cycle and continuously schedules in the same manner. The process time slicing in proposed architecture is shown in figure 4.2.

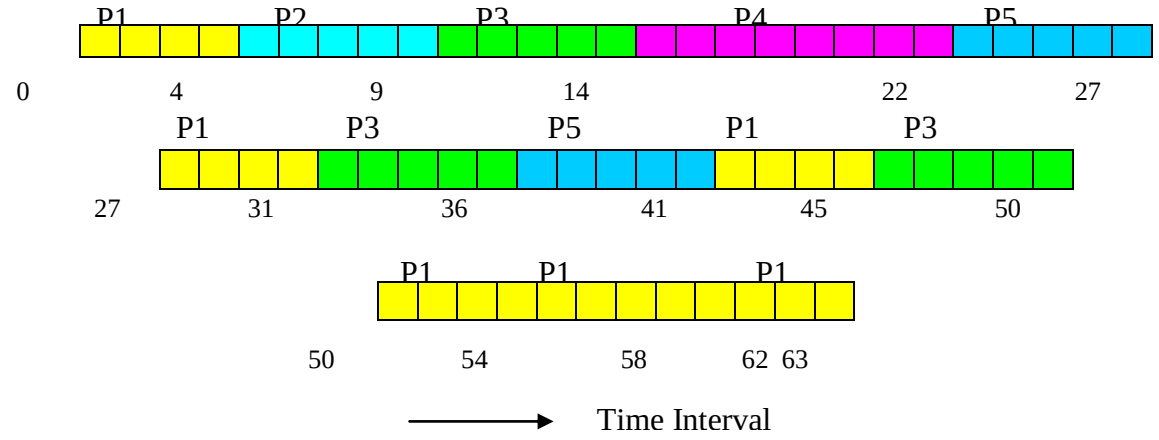

Figure 4.2 Time slicing in Proposed architecture

The context switch, waiting time and turn around time has been calculated using the formula given below and the results were compared.
Waiting time is calculated by

$$\sum_{i=1}^{n} \frac{wt_i}{n}$$

wt → waiting time of process.
n → No. of process.

Turn around time is calculated by

$$\sum_{i=1}^{n} \frac{tt_i}{n}$$

tt → turned around time of process.
n → No. of process.
The computed results are tabulated and tabulated in table 4.3 .

Table 4.3 Round robin and intelligent time slice for round robin comparison

Algorithm	Number of Context switch	Waiting time in milliseconds	Turn around time in milliseconds
Simple round robin	16	31	44
Intelligent time slice for round robin	13	25	37

It is observed that the proposed architecture is superior as it involves less waiting time, less turn around time and lower no of context switches. Therefore it is concluded that proposed intelligent time slice for round robin architecture has the potential to be considered in future research

5. CONCLUSION AND FUTURE WORK

A comparative study of round robin architecture and intelligent time slice for round robin architecture is made. It is concluded that the proposed architecture is superior as it has less waiting, response times, usually less preemption and context switching thereby reducing the overhead and saving of memory space. Future work can be based on this architecture modified and implemented for hard real time system where hard deadline systems require partial outputs to prevent catastrophic events.

REFERENCES

1. Barbara Korousic –Seljak (1994) "Task scheduling policies for real-time systems" Journal on MICROPROCESSOR AND MICROSYSTEMS, VOL 18, NO. 9, pg501-512.
2. Reindir J. bril and Pieter J.L. Cuijpers (2006) "Analysis of hierchial fixed priority preemptive scheduling "TU/e, CS-Report 06-36, www.win.tue.nl/~pcuijper/docs/CSR-06-36.pdf.
3. Richard roehi (1995) "Designing a fairer round robin scheduling algorithm" SIGCOMM '95 Cambridge, MA USA 0 ACM.
www.cmg.org/proceedings/2005/5056.pdf.
4. Marco.J, . Meziat.D, . Alarcos.B "Dificit round robin alternated : a new scheduling algorithm"_ E.P. 28871 Alcalá de Henares, Spain.
https://portal.uah.es/portal/page/portal/epd2.../proms00.pdf
5. Burns.A (1994) "Fixed priority scheduling with deadline prior to completion" Real-Time systems Research Group Department of computer science university of york, UK.
6. Shreedhar. M , George Varghese (1996) "Efficient fair queuing using deficit round robin" IEEE/ACM TRANSACTIONS ON NETWORKING VOL 4, NO3, pg375 -385.
7. Viet.L.Do and Kenneth Y.Yun (2003) "An Efficient Frame based scheduling Algorithm: Credit round robin " San Diego, IEEE.
8. Klaus H. Ecker , (1996) "Solving Hard Real-Time Scheduling Problems on a Single Processor: IEEE Proceedings of the 4th WPDRTS.
9. Sara. R ,Biyabani , John A. Stankovic, Krithi Ramamitham (1988) "The Integration of Deadline and Criticalness in Hard Real-Time Scheduling'Sara R" IEEE.
10. Sha, L., Abdelzaher, T, Arzen, K., Cervin, A., Baker, T.P., Burns, A., Buttazzo, G., Caccamo, M., Lehoczky, J., and Mok, (2004) "A. Real time scheduling theory: A historical perspective". Real-Time Systems, 28(2):46-61.
www.springerlink.com/index/W2312110GL0243K8.pdf
11. Ripoll I. et al. (2003) "RTOS state of the art analysis. Technical report", OCERA Project.
12. Enrico Bini and Giorgio C. Buttazzo (2004) "Schedulability Analysis of Periodic Fixed Priority Systems" journal on IEEE TRANSACTIONS ON COMPUTERS VOL 53 NO.11, pg 1462-1473.
13. Zhi Quan and Jong-Moon Chang (2003) " A Statistical Framework for EDF Scheduling" journal on IEEE COMMUNICATION LETTERS VOL 7 NO.10, pg 493-495.
14. Houssine Chetto and Maryline Chetto (1989) " Some Results of Earliest Deadline Scheduling Algorithm" journal on IEEE TRANSACTION ON SOFTWARE ENGINEERING. VOL 15. NO.10, pg 1261-1269.