ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

Real Time System Scheduling Algorithms & Fault Tolerance

Charu Rani[1], Mrs. Manju Godara[2]

Student, Cse, Jan nayak Choudhary Devi lal Memorial College of Engineering, Sirsa, India[1]

Lecturer-cum-Guide, Cse, Jan nayak Choudhary Devi lal Memorial College of Engineering, Sirsa, India[2]

Abstract: The main objective of this paper is to implement the real time scheduling algorithms and to minimize the average waiting time so that given set of tasks may be completed in a minimal time with an efficient output. We will also discuss the advantages and disadvantages of the same. Task within the real time system are designed to accomplish certain service(s) upon execution, and thus, each task has a particular significance to overall functionality of the system. Scheduling algorithms in non real time system not considering any type of dead line but in real time system deadline is main criteria for scheduling the task.

Keywords: CPU Scheduling, FCFS, Non Preemptive, Preemptive, Round Robin, Round Robin with SRTF, SJF.

I. INTRODUCTION

A. Real time system

The Oxford Dictionary of Computing defines a real-time system as:

Any system in which the time at which output is produced is significant. This is usually because the input corresponds to some movement in the physical world, and the output has to relate to that same movement. The lag from input time to output time must be sufficiently small for acceptable timeliness.

The correct behavior of a real-time system depends as much on the timing of computations as it does on the results produced by those computations. Results delivered too late may be useless, or even harmful.

According to Silberchatz, Galvin and Gagne[1]; The aim of operating system to allow a number of processes concurrently in order to maximize the CPU utilization. The most important attribute of a task in a real-time system is its timing constraints. Such timing constraints must be expressed precisely. A deadline is the most widely used form of a timing constraint. It offers a dual view of the usefulness of a task's completion with respect to a single point in time. The completion of a task is of no value beyond the deadline(timing constraints). Improper use of CPU can reduce the efficiency of the system in multiprogramming computing systems. In multiprogramming systems, multiple processes are being kept in memory for maximum utilization of CPU [2]. CPU utilization can be maximized by switching CPU among waiting processes in the memory and running some process all the time [1] and Waiting time is the amount of time of a process has been waiting in the ready queue [9]. The main aim of the CPU scheduling algorithms is to minimize waiting time, turnaround time, response time and context switching and maximizing CPU utilization. This study focuses on improving the effectiveness of Round Robin CPU scheduling algorithm.

As per "Office of Aviation Research and Development Washington, D.C. 20591. Real-Time Scheduling Analysis. November 2005"[3] A real-time scheduling System is composed of the scheduler, clock and the processing hardware elements. In a real-time system, a process or task has schedule ability; tasks are accepted by a real-time system and completed as specified by the task deadline depending on the characteristic of the scheduling algorithm.

1) Hard real time system:

Hard real time tasks are periodic activities. In hard real time system a hard dead line is a completion time constraint, such that if the deadline is satisfied, i.e., the task's execution point reaches the end of the deadline scope before the deadline time occurs, then the time constrained portion of the task's execution is timely, otherwise that portion is not timely. E.g. in Autopilot system microprocessor must control the airbags etc in case of out of control, nuclear plant control etc.

2) Soft real time systems:

SRT-tasks are activities that are not subject to strict deadlines. Usually, they are aperiodic tasks.

A Soft deadline is a completion time constraint, such that if the deadline is satisfied i.e., the task execution point reaches the end of the deadline scope before the deadline time occurs then the time constrained portion of the task's execution is more timely. Thus hard dead line is the special case of soft deadline. E.g. telephone switching it makes the connection before process execution, Image processing applications etc where utility of result decreases over time after deadline expires.

There are a large variety of real time systems but all have common characteristics, which differentiate them from non-real time systems.

2.1) Time Constraint: One very common form of time constraint is deadline associated with tasks. A task deadline specifies the time before which the task must complete and produce results. It is the responsibility of the RTOS, schedulers particularly, to ensure that all tasks meet their respective deadline.

DOI 10.17148/IJARCCE.2015.4762

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

2.2) Safety-Criticality: For traditional non-real time systems safety and reliability are independent issues. However, in many real time systems these two issues are intricately bound together making them safety-critical.

II. APPLICATIONS OF REAL TIME SYSTEM

Commercial transaction systems, transportation systems, and military/space systems - to name a few. The supporting research includes system architecture, design techniques, coding theory, testing, and validation, proof of correctness, modeling, software reliability, operating systems, parallel processing, and real-time processing. These areas often involve widely diverse core expertise ranging from formal logic, mathematics of stochastic modeling, graph theory, hardware design and software engineering. Redundancy has long been used in fault-tolerant and adaptive systems. However, redundancy does not inherently make a system fault-tolerant and adaptive; it is necessary to employ fault-tolerant methods by which the system can tolerate hardware component failures, avoid or predict timing failures, and be reconfigured with little or graceful degradation in terms of reliability and functionality. Early error detection is clearly important for real-time systems; error is an abbreviation for erroneous system state, the observable result of a failure.

The ability to deliver service is called dependability. The schema of dependability computing Means to attain dependability has been grouped by researchers into four major categories:

A. Fault prevention/avoidance

Fault prevention aims at reducing the creation or occurrence of faults during the computing system life cycle. Means are used during the system design phase. Some of them have an impact on the created system. Others prevent faults occurring during its useful life. These means concern the system modeling tools (including implementation technologies), the system models and the processes used to obtain these models.

B. Fault tolerance

Fault tolerance aims at guaranteeing the services delivered by the system despite the presence or appearance of faults. Fault tolerance approaches are divided into two classes:

- Compensation techniques for which the structural redundancy of the system masks the fault presence.
- Error detection and recovery techniques, that is, detection and then resumption of the execution either from a safe state or after the operational structure modification (reconfiguration).

C. Fault removal

Fault removal aims at detecting and eliminating existing faults. Fault removal are older that those on fault prevention. Fault removal techniques are often considered at the end of the model definition, particularly when an operational model of the system is complete.

Scheduling Algorithms are used for fault tolerance as well as fault avoidance which may be classified as

1) First Come First Serve
2) Shortest Job First
3) Preemptive
4) Non-Preemptive
5) Round-Robin Technique

Round Robin is considered the most widely used scheduling algorithm in CPU scheduling [5]-[6], also used for flow passing scheduling through a network device [7]-[8].

III. CPU SCHEDULING

In multi-programmed operating systems CPU scheduling plays a fundamental role by switching the CPU among various processes [1]. CPU scheduling algorithms are used to allocate the CPU to the processes waiting in the ready queue. Some of the popular CPU scheduling algorithms are First-Come-First-Served (FCFS), Shortest Job First (SJF), Priority Scheduling and Round Robin (RR). FCFS is the simplest form of CPU scheduling algorithm. In this scheduling algorithm, the process that arrives first in ready queue served first, so the name First-Come-First-Served. The average waiting time in FCFS is quite long [1]. In Shortest Job First (SJF) algorithm, process from the ready queue that has shortest CPU burst time will execute first. If two processes are having same CPU burst time and arrival time, then FCFS procedure is followed. In SJF average waiting time decreases. Priority scheduling algorithm allocates the CPU to the higher priority process from the ready queue. In Round Robin (RR), a small unit of time quantum is given to each process present in the ready queue which maintains the fairness factor. In this paper we have proposed an improvement in RR to reduce the average waiting time.

A. Scheduling Criteria

1) CPU Utilization:
This is a measure of how much busy the CPU is.

2) Throughput:
Throughput is the no. of processes completed per time period.

3) Waiting time:
Waiting time[9] is the sum of periods spends waiting in the ready queue. CPU executes only one job at a time. The rest of process wait for the CPU.

4) Turnaround time:
It is the total time taken to execute a job. The interval from the time of submission of a process to the time of completion is the turnaround time.

5) Response time:
Response time is the time from the submission of a request until the first response is produes.

IV. ASSUMPTIONS

- Set of tasks **Ti** every task has attribute arrival time, Deadline, Worst case execution time.
- **Priorities** of the task, priorities are set according to the scheduling algorithm.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

- **System_clock** is the clock set for the task i.e. deadline time to every task, system clock is set according to the task requirement.
- **arrival_queue** is used to store the arrival task, i.e. all the arrival tasks are inserted into the **arrival_queue** .
- **ready_queue** is the queue for inputting the ready task, i.e all the ready to execute tasks are store in ready queue.
- **remove** method is used to take the task from queue.
- **execute** method is used to process the task, i.e. according to the turn of the task they are get executed.
- **Compare method** function is used to compare the task for their deadline if deadline of the task **Ti** is greater than the task **Tk** then task **Ti** is executed otherwise **Tk** is executed using executed method if both having equal deadline then task with first arrival is executed.

V. ALGORITHMS

A. First Come First Serve (FCFS)

Algorithm 1) FCFS:

1. Input set of periodic tasks set S=T1, T2, T3……., Tn, Duration, Activation Time, Deadline in arrival_queue ;
2. Calculate priorities of the task according to their deadline.// Highest priority is given to critical delay i.e. smaller delay higher its priority.
3. for time 1,2,3……system_clock, system_clock do
4. remove task from arrival_queue and put all tasks Ti in ready_queue;
5. start with highest priority task Ti according to their deadline;
6. while(ready_queue is not empty)

```
{
   remove task Ti from ready_queue;
   execute task Ti on single CPU;
   compare(Ti,Tk)
    {
  if Deadline(Ti) > Deadline(Tk)
        execute Ti;
else
execute Tk on single CPU;
if Deadline(Ti) = = Deadline(Tk)
use first input of ready_queue;
              }
          }
end while
end for
```

7. end Algorithm

In FCFS CPU scheduling algorithm, the process that arrives first in the ready queue is served first. The average waiting time in this scheduling is quite long [1].

B. Shortest Job First Algorithm(SJF)

Algorithm 1) SJF:

1. Input set of periodic tasks set S=T1, T2, T3……., Tn, Duration, Activation Time, Deadline ;
2. Calculate priorities of the task according to their deadline.// Highest priority is given to critical delay i.e. smaller delay higher its priority.
3. for time 1,2,3……system_clock, system_clock do
4. remove task from arrival_queue and put all tasks Ti in ready_queue;
5. start with highest priority task Ti according to their deadline;

```
while(ready_queue is not empty)
{
   remove task Ti from ready_queue;
   execute task Ti on Multiple CPUs;
   compare(Ti,Tk)
    {
  if Deadline(Ti) > Deadline(Tk)
        execute Ti on Multiple CPUs;
else
execute Tk on Multiple CPUs;
if Deadline(Ti) = = Deadline(Tk)
use  first input of ready_queue;
  for T=1 To Ti //Loop to check for preemption
  if newly arrival task Tk  > currently executing task Tk
          then
preempt();
    }
}           end while
end for
preempt ()
{
  Temp=Ti;
  execute Tk on Multiple CPUs;
  assign priority to newly arrival task;
}
```

8. end Algorithm

C. Round robin Quantum Non-Preemptive (RRQ_NP) algorithm

Algorithm 1) RRQ_NP:

1. Input set of periodic tasks set S=T1, T2, T3……., Tn, in arrival_queue Duration, Activation Time, Deadline,Quantum;
2. Calculate priorities of the task according to their deadline.// Highest priority is given to critical delay i.e. smaller delay higher its priority.
3. Set Quantum=n;
4. for time 1,2,3……system_clock, system_clock do
5. remove task from arrival_queue and put all tasks Ti in ready_queue;
6. start with highest priority task Ti according to their deadline;
7. while(ready_queue is not empty)

```
{
   remove task Ti from ready_queue;
   execute task Ti on Singlee CPU;
 if Quantum is complete
then
take another task Tk according to priority on single CPU;
end if
end while
end for
```

8. end Algorithm

DOI 10.17148/IJARCCE.2015.4762

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

D. Round Robin Quantum Preemptive(RRQ_P) Algorithm

Algorithm 1) RRQ_P:

1. Input set of periodic tasks set S=T1, T2, T3……., Tn, in arrival_queue Duration, Activation Time, Deadline,Quantum;
2. Calculate priorities of the task according to their deadline.// Highest priority is given to critical delay i.e. smaller delay higher its priority.
3. Set Quantum=n;
4. for time 1,2,3……system_clock, system_clock do
5. remove task from arrival_queue and put all tasks Ti in ready_queue;
6. start with highest priority task Ti according to their deadline;
7. while(ready_queue is not empty)

{

remove task Ti from ready_queue;
execute task Ti on Multiple CPU;
if Quantum is complete
then
take another task Tk according to priority on Multiple CPU;
If task with higher priority than preempt currently executing task
preempt ();
end if
end while
end for

8. end Algorithm

E. Improved Round Robin Quantum Preemptive (IRRQ) Algorithm

Algorithm 1) IRRQ:

1. Input set of periodic tasks set S=T1, T2, T3……., Tn, in arrival_queue Duration, Activation Time, Deadline,Quantum;
2. Calculate priorities of the task according to their deadline.// Highest priority is given to critical delay i.e. smaller delay higher its priority.
3. Set Quantum=n;
4. for time 1,2,3……system_clock, system_clock do
5. remove task from arrival_queue and put all tasks Ti in ready_queue;
6. start with the first process in the ready queue and sort acc. To least remaining execution time, highest priority task Ti according to their deadline;
7. while(ready_queue is not empty)

{

remove task Ti from ready_queue;
execute task Ti on Multiple CPU;
if Quantum is complete
then
take another task Tk from ready queue;
If task with lower remaining time after every quantum then preempt currently executing task
preempt ();
end if
end while
end for

8. end Algorithm

VI. PROBLEM FORMULATION

In the above mentioned algorithms Round Robin algorithm is better. However the average waiting time of SRTF is lesser than Round Robin but SRTF may lead to starvation leaving the CPU in a deadlock position. So in order to reduce the average waiting time and starvation problem we have to developed a new algorithm.

VI. PROPOSED RESEARCH WORK

In the recent years, a number of CPU scheduling mechanisms have been developed for predictable allocation of processor.

Mixed Scheduling (A New Scheduling Policy) [4], uses the job mix order for non preemptive scheduling FCFS and SJF. According to job mix order, from a list of N processes, the process which needs minimum CPU time is executed first and then the highest from the list and so on till the nth process.

Time quantum is continuously adjusted according to the burst time of the processes in Self-Adjustment Time Quantum in Round Robin Algorithm[10].

Robust quantum time value has been proposed in [11] after arranging the process in the ascending order and taking the average of minimum and maximum burst time of the processes in the ready queue.

A. Proposed Improved Round Robin(IRR) Scheduling Algorithms

An Improved Round Robin Scheduling Algorithm for CPU Scheduling [1]. We have combined the features of SRTF and Round Robin. The improved Round Robin will never lead to starvation and will have lesser average waiting time.

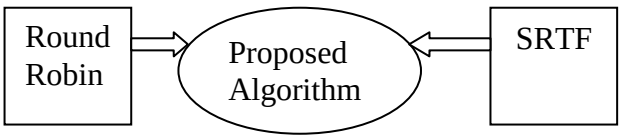


Fig.(IRR=RR+SRTF)

Following is the proposed IRR(Improved Round Robin) CPU scheduling algorithm:

1. Make a ready queue RQUEUE of the Processes submitted for execution in order of their arrival time.

2 Execute the first process from the queue for the quantum or CPU brust time, whichever is lesser.

3. DO steps 4 to 6 WHILE queue RQUEUE becomes empty.

4. Arrange the processes in the ready queue REQUEST in the ascending order of their remaining burst time.

DOI 10.17148/IJARCCE.2015.4762

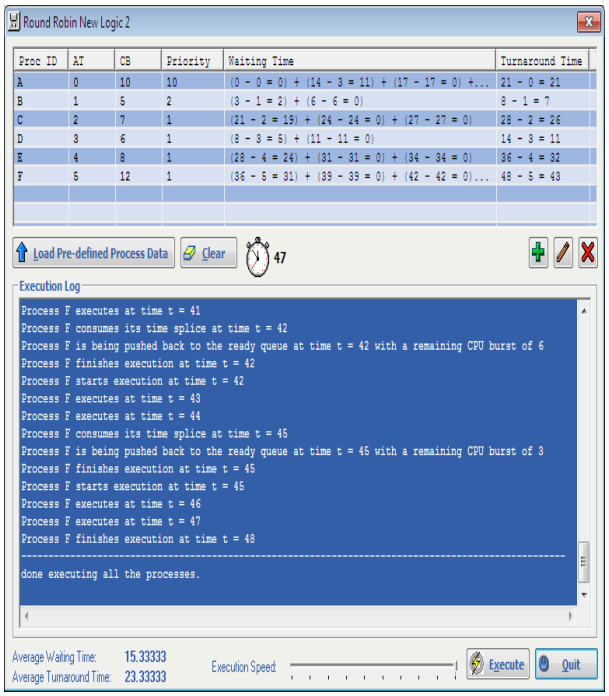
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

5. Execute the first process from the queue for the quantum or CPU brust time, whichever is lesser.

6. Remove the currently running process from the ready queue RQUEUE, if it has finished execution time and Go To step 4.

7 END Algorithm.



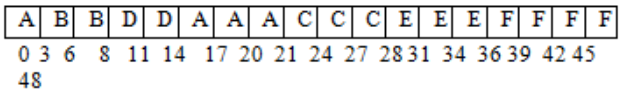
VI. ILLUSTRATION

Suppose we have six processes as per the details given below:

Sr no.	Proc ID	Arrival Time	Brust Time
1.	A	0	10
2.	B	1	5
3.	C	2	7
4.	D	3	6
5.	E	4	8
6.	F	5	12

1) **In IRR** ready queue with six processes A,B,C,D.E and F has been considered for illustration purpose. The processes are arranged in the ascending order of their arrival time in the ready queue which gives the sequence A,B,C,D,E and F. The time quantum value is set equal to 3. CPU is allocated to the process A at Clock 0 from the ready queue for a time quantum of 3 milliseconds (ms). After first quantum, the remaining burst time for A is 10-3=7.
2) Now, Clock time is 3 there are four processes A,B,C,D in the ready queue and having remaining Brust time 7,5,7,6 resp.The processes A,B,C and D are arranged in the ascending order of their remaining burst time in the ready queue which gives the sequence B,D,A and C having new brust time 5,6,7,7 resp. CPU is allocated to the processes B from the ready queue for a time quantum of 3ms. After second quantum, the remaining burst time for B is 5-3=2.
3) Now, Clock time is 6 there are six processes A,B,C,D,E and F having remaining Brust time 7,2,7,6,8,12 resp.The processes A,B,C,D,E and F are arranged in the ascending order of their remaining burst time in the ready queue which gives the sequence B,D,A,C,E,F having new brust time 2,6,7,7,8,12 resp. CPU is allocated to the processes B from the ready queue for a time quantum of 2ms which is less than allocated quantum.After quantum, the remaining burst time for B is 0. The process B has finished execution, so it is removed from the ready queue.
4) Now, Clock time is 8 there are five processes A,C,D,E and F having remaining Brust time 7,7,6,8,12 resp.The processes A,B,C,D,E and F are arranged in the ascending order of their remaining burst time in the ready queue which gives the sequence D,A,C,E,F having new brust time 6,7,7,8,12 resp.Again it repeats the same procedure until Ready queue is not empty.
5) The average waiting time is 15.33 ms in IRR. while average waiting time *is 26.33 in RR.*
6) Average waiting time of all scheduligs with the same set of processes and the values are given below:

Algorithm	Avg. Waiting Time	Avg. Turn around Time
FCFS	16	24
SJF	15.83	23.83
Priority(np)	18.16	26.16
Priority(preempt)	17.66	25.66
SRTF	14.16	22.16
RR	26.33	34.33
IRR	15.33	23.33



VII. CONCLUSION

From the above comparisons it is clear that average waiting time of IRR is lesser than all the algorithms except for SRTF.
But SRTF may result in starvation so IRR is better than SRTF also as it'll never result in deadlock.

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

REFERENCES

[1] A. Silberschatz, P. B. Galvin & G. Gagne, [John Wiley and Sons] *"Operating System Concepts,"* [7th edition 2005] Inc,157-167.

[2] Rakesh Kumar Yadav, Abhishek K Mishra, Navin Prakash & Himanshu Sharma, An Improved Round Robin Schedduling Algorithm for CPU Scheduling *International Journal on Computer Science and Engineering, Vol. 2, No. 4, 2010,1064-1066.*

[3] http://www.tc.faa.gov/its/worldpac/techrpt/ar05-27.pdf

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

[5] Silberschatz ,Galvin and Gagne, *Operating systems concepts*, (8th edition, Wiley, 2009).

[6] Lingyun Yang, Jennifer M. Schopf and Ian Foster, *Conservative Scheduling: Using predictive variance to improve scheduling decisions in Dynamic Environments", Super Computing* 2003, USA, Phoenix, AZ, November, 15-21,

[7] Weiming Tong, Jing Zhao, *Quantum Varying Deficit Round Robin Scheduling over Priority Queues, international Conference on Computational Intelligence and Security.*(China, 2007) 252- 256.

[8] Abbas Noon1, Ali Kalakech2, Seifedine Kadry1, A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average, *IJCSI International Journal of Computer ScienceIssues, Vol. 8, Issue 3, No. 1, May 2011.*

[9] Saroj Hiranwal and K. C. Roy, Adaptive Round Robin Scheduling Using Shortest Burst ApproachBased On Smart Time Slice, *International Journal of Computer Science and Communication Vol. 2, No. 2, July-December 2011,319-323*

[10] Rami J., *Matarneh,Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of Now Running Processes", American J. of Applied Sciences, Vol. 6, No. 10* (2009) 1831-1837.

[11] M Lavanya & S. Saravanan, Robust Quantum Based Low-power Switching Technique to improve System Performance, *International Journal of Engineering and Technology, Vol. 5, No. 4, 2013,3634-3638.*