Simulation based Metrics Performance of Real time Scheduling Algorithms

Kirti Patel, Ramesh Prajapati
P.G.Student, Asst.Prof.
CE Department,
Department of Computer Engineering,
Faculty of Engineering & Technology,Rajpur-382715

***Abstract* - Real time systems are systems in which there is a commitment for timely response by the computer to external stimuli. Real Time applications have to function correctly even in presence of faults. Scheduling algorithms are a governing part of real time systems and there exists many different scheduling algorithms due to varying needs and requirements of different real time systems. The choice of algorithm is important in every real time system and is greatly influenced by what kind of system the algorithm will serve. The scheduler must meet the timing constraints of the physical system. The basic requirement of real time system is predictability of the various metrics of the system. Unless the behavior of a real-time system is predictable, the scheduler cannot guarantee that the computation deadlines of the system will be met. This metrics of scheduling algorithms can be predicted by using various simulation tools. Here we are discussing the various real time scheduling algorithms and simulators to analysis the metrics performance of these real time scheduling algorithms.**

***Keywords* - SRS;UML;ASSERT;PERTS;SEW;AFTER**

I. INTRODUCTION

Real time systems are systems intended to serve real time applications and process data as it comes in without any time delay. Such systems are used to govern physical applications that differ in complexity from vehicle ignition structures to controllers for flight structures and nuclear energy plants. Such structures are called hard real time systems due to the fact that the timing necessities of system computations ought to usually be met or the system will fail. The scheduler for such systems must coordinate resources to satisfy the temporal order constraints of the physical system. This means that the scheduler should be able to predict the execution behavior of all tasks among the system. The primary requirement of real-time structures is predictability. Unless the conduct of a time period system is predictable, the scheduler cannot guarantee that the computation deadlines of the system are met.

It is the need of prediction of the scheduled task that differentiates actual real-time systems from standard computing environments and makes the scheduling solutions for standard systems inappropriate for real-time systems. In standard computing environment like that in time sharing system timing parameters for the task and resources are not known in advance. As such, no effort is formed in these systems to work out, a priori, the execution behavior of the tasks. It's moreover the case in these systems that an unknown waiting time for computation results is acceptable. None of those assumptions hold for a real-time system. To predict scheduling in real time systems, all the tasks and their resource and computation necessities should be best-known in advance. Also, for hard real-time systems, unknown delays anticipating computation results are unacceptable. A scheduling resolution that captures the temporal arrangement behavior of the system is required for a real-time system

The general scheduling answer for the task predicting needs of real-time systems is to apply cyclical executions. Cyclical executions create a static timeline upon which task and resources are assigned in specific time intervals that provide a deterministic schedule for all task and resources. Timing requirements of many real time systems has met using this technique. However, the use of cyclical executions is the general scheduling solution for prediction demand in most of real time systems. This technique has been ready to meet the timing needs of the many real-time systems. However, the cyclical execution approach has many serious drawbacks. Though a Cyclical execution provides predictability through the utilization of a deterministic schedule, it's the need of determinism that is supply of those drawbacks. A period of time system usually has many various timing constraints that have to be met and no formal algorithm exists for making timelines to fulfill these constraints. Therefore, every real-time system needs a unique, handcrafted cyclical execution that needs intensive testing to verify its correctness. Also, due to the numerous ad hoc selections made to form the timelines, changes to the system have unpredictable effects upon the correctness of the timelines and so, an entire and new set of testing is needed. Thus, the alternating executive approach is usually expensive to form, verify, and update. As real-time systems become more complex, the issues related to alternating executives only become harder. A much better scheduling solution is required that has sure thing while not the need of determinism.

II. REAL TIME TASK DESCRIPTION

A real time task is usually divided into three categories basically based upon its arrival pattern and its deadline. All real time tasks are placed into one of these three categories. If it is essential for a system operation to meet a deadline for given task, then such task's deadline is considered as hard deadline. Usually it is desirable that given task deadline is met but sometimes missing the task deadline is tolerated, then deadline is known as soft. Tasks with regular arrival times are known as periodic tasks. A common

use of periodic tasks is to process device knowledge and update this state of the real-time system on a regular basis. Periodic tasks, typically used in control and signal process applications, have hard deadlines. Tasks with irregular arrival times are non periodic or aperiodic tasks. Aperiodic tasks are used to handle the processing needs of random events like operator requests. An aperiodic task generally has a soft deadline. Aperiodic tasks that have hard deadlines are known as sporadic tasks. We assume that every task contains a best-known worst case execution time. In summary, we have

A. PERIODIC TASKS

Periodic tasks are real-time tasks that are activated frequently at fixed periods. Normally, periodic tasks have a constraint which indicates that instances of them should execute once per period.

B. Aperiodic Tasks

An aperiodic task is a stream of jobs arriving at irregular intervals. The time constraint is usually a deadline.

C. Sporadic Tasks

Sporadic tasks are real-time tasks that are activated on an irregular basis with some best-known finite rate. The bounded rate is characterized by a minimum inter-arrival period, that is, a minimum interval of time between two successive activations. The time constraint is usually a deadline. It is impossible to ensure that a sporadic task's deadline would always be met.

To meet the timing constraints of the system, a scheduler should coordinate the use of all system resources employing a set of well understood real-time scheduling algorithms that meet the subsequent objectives:

• Guarantee that tasks with hard temporal arrangement constraints can always meet their deadlines.

• Attain a high degree of schedulable utilization for hard deadline tasks (periodic and sporadic tasks). Schedulable utilization is the degree of resource utilization at or below which all hard deadlines is secured. The schedulable utilization attainable by an algorithm is a measure of the algorithm's utility: the higher the schedulable utilization, the more applicable the algorithm is for a range of real-time systems.

• Provide quick average response times for tasks with soft deadlines (aperiodic tasks).

• Ensure scheduling stability beneath transient overload.

In the offline setting, there is no distinction between the design of scheduling algorithm and checking feasibleness of a task system. The scheduling algorithm provides a possible sequence of jobs wherever tasks are always completed by the deadlines. In online setting the real time scheduling theory focuses on two totally different scheduling issues.

1. The feasibility analysis problem given task system and scheduling environment determines whether or not there exists a schedule that can meet all the deadlines.
2. The run time scheduling problem of a given task system that is best-known to be possible, determine scheduling algorithm deadline, according to the supported scheduling environment.

III. COMPARISON OF PERFORMANCE METRICS OF SCHEDULING ALGORITHMS

Task scheduler	Implementation	Processor Utilization	Context Switches	No. of Preemptions	Predictability
RM	Multi-level queue	Not guaranteed	Many	Few	Good
EDF	Heap	full utilization	few	Many	Bad
Polling Server	FIFO	Low utilization	Many	Many	Bad
Deferrable Server	FIFO	High utilization	Few	Few	Good
Sporadic Server	FIFO	Better than PS and DS	Very few	Very Few	Good

Table – 3.1

IV. IMPLEMENTATION DETAILS AND PARAMETER

*A. **Module 1: Implementation of Basic Algorithms.***

This is basically a simulator based approach where all algorithms are implemented based on dynamic approach. By single change, anyone can add new algorithm in this simulator and will get comparative results compare to existing algorithms.

Accept Task Set From the User

Class Name: Public Class input
Method1: Private Sub tn_add_periodic_task_set_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnadd.Click
Method 2: Private Sub btn_ok_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn_ok.Click

Class Name: Public Class ip_server
Method 1: Private Sub btnadd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnadd.Click
Method 2: Private Sub btn_ok_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn_ok.Click

Execution Environment for Rate-Monotonic Scheduling Algorithm

Class Name: Public Class frmrm
Method 1: Private Sub btnschedule_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnschedule.Click

Execution Environment for Earliest Deadline First (EDF) Scheduling Algorithm

Class Name: Public Class frmedf
Method 1: Private Sub btnschedule_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnschedule.Click

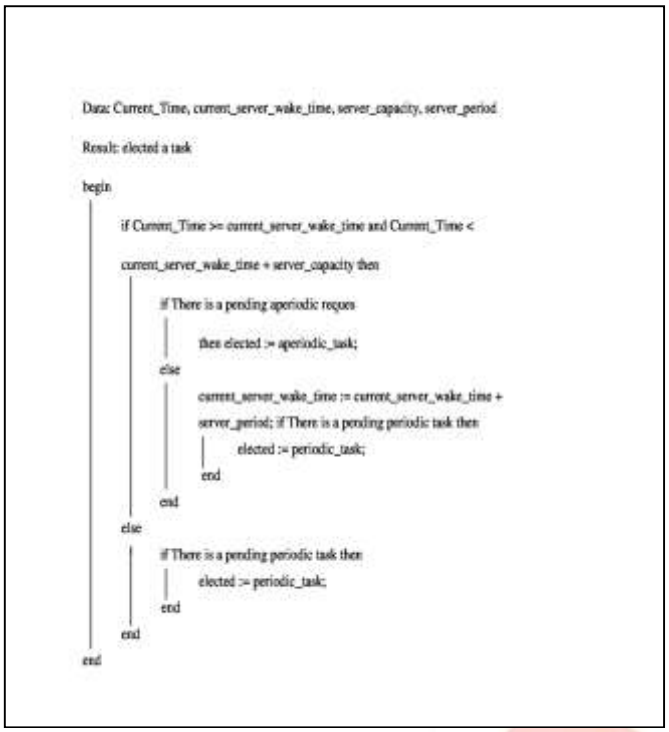
Execution Environment for Polling Server Scheduling Algorithm

Class Name: Public Class poll
Method 1: Private Sub btnschedule_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnschedule.Click

Execution Environment for Deferrable Server Scheduling Algorithm

```
Data: Curremet_Time, Current_Server_Capacity, Current_Server_wake_Time,
      Server_Capacity, Server_Period
Result: elected a task
begin
    /* Refill the Server capacity and compute the next wake time */
    if Current_Time = current_server_wake_time then
        current_server_capacity := server_capacity;
        current_server_wake_time := current_server_wake_time + server_period;
    end
    /* Election                                                  */
    if current_server_capacity > 0 and there is a pending aperiodic task then
        elected := aperiodic_task;
        current_server_capacity := current_server_capacity - 1;
    else
        if There is a pending periodic task then
            elected := periodic_task;
        end
    end
end
```

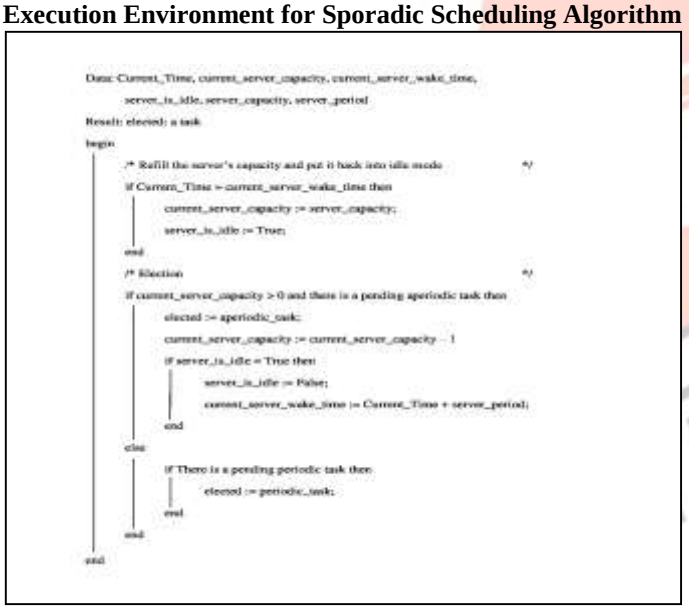
Class Name: Public Class deff



```
Data: Current_Time, current_server_wake_time, server_capacity, server_period
Result: elected a task
begin
    if Current_Time >= current_server_wake_time and Current_Time <
    current_server_wake_time + server_capacity then
        if There is a pending aperiodic reques
            then elected := aperiodic_task;
        else
            current_server_wake_time := current_server_wake_time +
            server_period; if There is a pending periodic task then
                elected := periodic_task;
            end
        end
    else
        if There is a pending periodic task then
            elected := periodic_task;
        end
    end
end
```

Method 2: Private Sub btnschedule_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnschedule.Click

Execution Environment for Sporadic Scheduling Algorithm



```
Data: Current_Time, current_server_capacity, current_server_wake_time,
      server_is_idle, server_capacity, server_period
Result: elected: a task
begin
    /* Refill the server's capacity and put it back into idle mode        */
    if Current_Time = current_server_wake_time then
        current_server_capacity := server_capacity;
        server_is_idle := True;
    end
    /* Election                                                           */
    if current_server_capacity > 0 and there is a pending aperiodic task then
        elected := aperiodic_task;
        current_server_capacity := current_server_capacity - 1
        if server_is_idle = True then
            server_is_idle := False;
            current_server_wake_time := Current_Time + server_period;
        end
    else
        if There is a pending periodic task then
            elected := periodic_task;
        end
    end
end
```

Class Name: *Public Class sporadic*

Method 2: Private Sub btnschedule_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnschedule.Click

B. Module 2: Calculation of Performance Matrices

This module is included in respective algorithm class and their methods and functions and it is used to calculate performance matrices for each scheduler after every quantum. Various class mentioned above contain function that returns currently executing task id along with conditions check when task arrives for its next execution. Function for selecting the task mentioned below returns new selected task and pre task execution returns previously selected task. Function also returns total quantum that has already executed. Function in various classes also returns total utilization of tasks at any instance.

C. Module 3: Simulated Analysis of Scheduling Algorithms

Function1: Public Function rm(ByVal no As Integer) in Class Name: Public Class frmrm
Function 2: Public Function edf(ByVal no As Integer) in Class Name: Public Class frmedf
Function 3: Public Function poll(ByVal no As Integer) in Class Name: Public Class poll
Function 4: Public Function deff(ByVal no As Integer) in Class Name: Public Class deff
Function 5: Public Function spor(ByVal no As Integer) in Class Name: Public Class sporadic

Function 6: Private Sub btnschedule_Click (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnschedule.Click in Class Name: Public Class sporadic

Above mentioned function in respected class code gives the simulated based analysis in graphical manner. It shows clearly that at what quantum which task is executing, what was the sequence of task execution, whether there is context switching or not at particular quantum etc.

Sr. No.	Test Case Name	Test Case Description	Steps			Test Status (P/F)
			Steps	Expected Result	Actual Result	
1.	Type of Scheduling Algorithm	To select type of the Scheduling Algo.	Click on "Rate Monotonic"	"Input Form For Rate Monotonic" get open	"Input Form For Rate Monotonic" get open	**P**
			Click on "Earliest Deadline First"	"Input Form For Earliest Deadline First" get open	"Input Form For Earliest Deadline First" get open	**P**
			Click on "Polling Server"	"Input Form For Polling Server" get open	"Input Form For Polling Server" get open	**P**
			Click on "Deferrable Server"	"Input Form For Deferrable Server" get open	"Input Form For Deferrable Server" get open	**P**
			Click on "Sporadic Server"	"Input Form For Sporadic Server" get open	"Input Form For Sporadic Server" get open	**P**
			Click on "OK" and then on "Home"	Main form gets opened	Main form gets opened	**P**
2.	Task Information for EDF Scheduling	Here we Accept information of task and schedule by periodic scheduling algorithms- EDF	Enter the tasks name, arrival time, computation time, period & click on "Ok" and then on "Schedule Task"	Scheduling of shown on EDF output form	Scheduling of tasks shown on EDF output form	**P**
			Click on "OK" and then on "Home"	Main form gets opened	Main form gets opened	**P**
3.	Task Information for Polling Server Scheduling	Here we Accept information of task and schedule by aperiodic algorithms- Polling Server	Enter the tasks name, arrival time, computation time, period & click on "Ok" and then on "Schedule Task"	Scheduling of tasks shown on Polling Server output form	Scheduling of tasks shown on Polling Server output form	**P**
			Click on "OK" and then on "Home"	Main form gets opened	Main form gets opened	**P**
4.	Task Information for Deferrable Server Scheduling	Here we Accept information of task and schedule by aperiodic algorithms- Deferrable Server	Enter the tasks name, arrival time, computation time, period & click on "Ok" and then on "Schedule Task"	Scheduling of tasks shown on Deferrable Server output form	Scheduling of tasks shown on Deferrable Server output form	**P**
			Click on "OK" and then on "Home"	Main form gets opened	Main form gets opened	**P**
5.	Task Information for Sporadic Server Scheduling	Here we Accept information of task and schedule by aperiodic algorithms- Sporadic Server	Enter the tasks name, arrival time, computation time, period & click on "Ok" and then on "Schedule Task"	Scheduling of tasks shown on Sporadic Server output form	Scheduling of tasks shownon Sporadic Server output form	**P**

V. RESULT ANALYSIS

Fig-5.1

Analysis Result of Rate Monotonic Scheduling Algorithm

TASK SET Ti (Ai,Pi,Ei)	No. of Preemptions	No. of Context Switch	Utilization	Schedulability Test	Task Missing Deadline
T1(0,10,2) T2(2,6,3) T3(3,3,1)	7	57	1.03333	Fail	T1 misses deadline at 40
T1(0,20,2) T2(1,30,3) T3(3,40,4)	0	33	0.3	Success	None

Table-5.1

Fig-5.2

Here as shown in table 5.2 we have performance result for rate monotonic scheduling algorithm. Ai, Pi and Ei is the arrival time, Period time and execution time respectively of the given task. The same result is captured as shown in figure 5.2 for the given periodic tasks.

Analysis Result of Earliest Deadline First Scheduling Algorithm

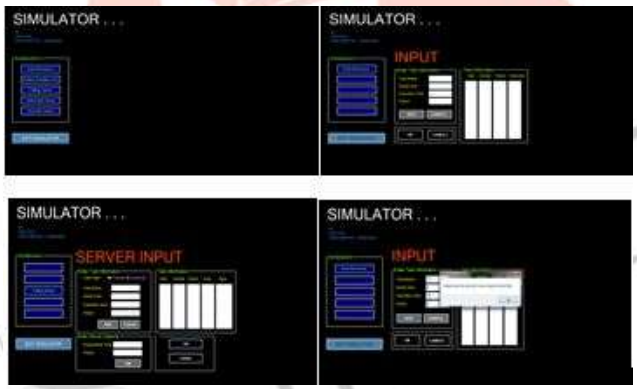


Table-5.2

Fig-5.3

Here as shown in table 5.2 we have performance result for Earliest Deadline First scheduling algorithm. Ai, Pi and Ei is the arrival time, Period time and execution time respectively of the given task. The same result is captured as shown in figure 5.3 for the given periodic tasks.

Analysis Result of Polling Server Scheduling Algorithm (Periodic)

TASK SET Ti (Ai,Pi,Ei)	No. of Preemptions	No. of Context Switch	Utilization	Schedulability Test	Task Missing Deadline
T1(0,20,10) T2(5,20,5) T3(3,30,6)	49	149	1.01	Fail	T3 misses deadline at 30
T1(0,20,2) T2(2,30,5) T3(3,20,10)	1	134	0.96666666	Success	none

Table-5.3

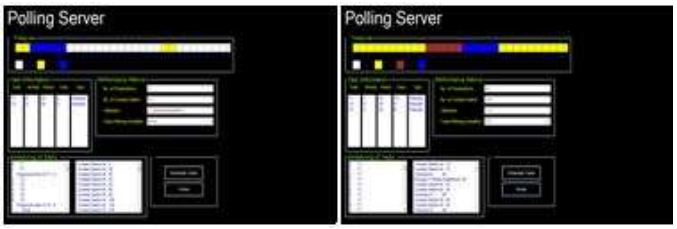


Fig-5.4

TASK SET Ti (Ai,Pi,Ei)	No. of Preemptions	No. of Context Switch	Utilization	Schedulability Test	Task Missing Deadline
T1(0,5,1) T2(2,12,6) T3(5,6,2)	7	21	1.03333	Fail	T3 misses deadline at 6
T1(0,20,10) T2(0,50,5) T3(0,35,10)	1	5	0.8857142	Success	None

Here as shown in table 7.4 we have performance result for Polling Server scheduling algorithm for periodic tasks. Ai, Pi and Ei is the arrival time, Period time and execution time respectively of the given task. Server Capacity for server algorithm must be given to get Schedulability results. The same result is captured as shown in figure 7.3 for the given periodic tasks.

Analysis Result of Polling Server Scheduling Algorithm (Aperiodic)

TASK SET Ti (Ai,Pi,Ei)	No. of Preemptions	No. of Context Switch	Utilization	Schedulability Test	Task Missing Deadline
A1(0,5,2) A2(0,5,2) T1(5,10,5)	1	1	0.7	Success	none

Table-5.3

Fig-5.4

Here as shown in table 7.3 we have performance result for Polling Server scheduling algorithm for aperiodic task. Ai, Pi and Ei is the arrival time, Period time and execution time respectively of the given task. For aperiodic task period time is same as server period. Every server's activation, it checks if there are any unfinished non periodic tasks, if there are, the server uses its capability to service them till either the task is finished or the server's capacity is depleted.

Analysis Result of Deferrable Server Scheduling Algorithm (Periodic)

TASK SET Ti (Ai,Pi,Ei)	No. of Preemptions	No. of Context Switch	Utilization	Schedulability Test	Task Missing Deadline
T1(0,20,2) T2(3,10,4) T3(3,12,5)	2	9	1.0166666	Fail	T3 misses deadline at 12
T1(01,50,10) T2(2,401,15) T3(5,30,6)	3	6	0.825	Success	none

Table-5.4

Fig-5.5

Here as shown in table 5.4 we have performance result for Deferrable Server scheduling algorithm for aperiodic tasks. Ai, Pi and Ei is the arrival time, Period time and execution time respectively of the given task. For Server Capacity for server algorithm must be given to get Schedulability results. The same result is captured as shown in figure 5.5for the given periodic tasks.

Analysis Result of Deferrable Server Scheduling Algorithm (Aperiodic)

TASK SET Ti (Ai,Pi,Ei)	No. of Preemptions	No. of Context Switch	Utilization	Schedulability Test	Task Missing Deadline
T1(0,5,2) T2(1,5,3) A1(2,3,1)	0	16	1.3333333	Fail	T2 misses deadline at 5

Table-5.5

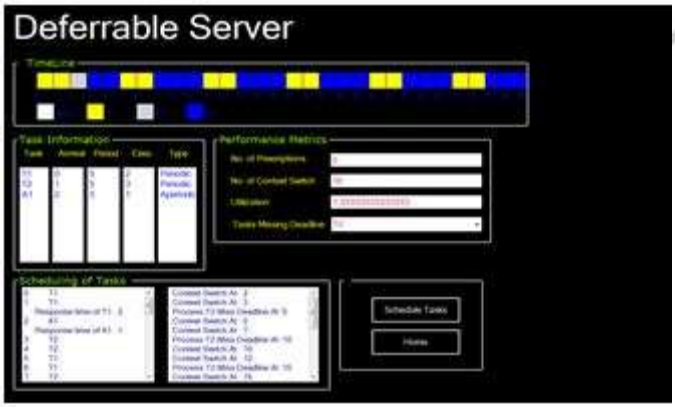


Fig-5.6

Here as shown in table 5.5 we have performance result for Deferrable Server scheduling algorithm for periodic tasks. Ai, Pi and Ei is the arrival time, Period time and execution time respectively of the given task. For aperiodic task period time is same as server period. As the polling server, the deferrable server has a capacity and a period. However, the deferrable server is not periodically activated to service aperiodic tasks however once an aperiodic task is pending and the server has capability superior to zero. The server then services the task until depletion of its capability or the aperiodic task is finished. The server's capacity is restored periodically every period time. If an aperiodic task happens when the server has depleted its capability, this task can be serviced once the server will have replenished its capability. Server Capacity for server algorithm must be given to get Schedulability results. The same result is captured as shown in figure 5.6 for the given periodic and aperiodic tasks.

Analysis Result of Sporadic Server Scheduling Algorithm (Periodic)

TASK SET Ti (Ai,Pi,Ei)	No. of Preemptions	No. of Context Switch	Utilization	Schedulability Test	Task Missing Deadline
T1(0,10,2) T2(5,12,6) T3(6,6,3)	7	17	1.3	Fail	T2 misses deadline at 12
T1(0,20,2) T2(1,30,3) T3(2,40,4)	0	5	0.4	Success	none

Table-5.6

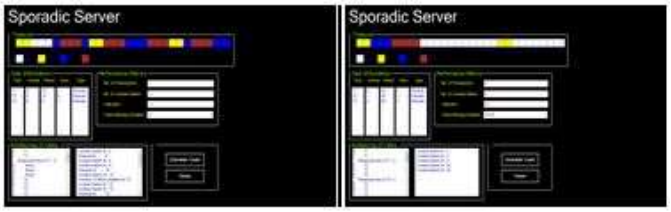


Fig-5.7

Here as shown in table 5.6 we have performance result for Sporadic Server scheduling algorithm for periodic tasks. Ai, Pi and Ei is the arrival time, Period time and execution time respectively of the given task. Server Capacity for server algorithm must be given to get Schedulability results. The same result is captured as shown in figure 5.7 for the given periodic tasks.

Analysis Result of Sporadic Server Scheduling Algorithm (Aperiodic)

TASK SET Ti (Ai,Pi,Ei)	No. of Preemptions	No. of Context Switch	Utilization	Schedulability Test	Task Missing Deadline
T1(0,5,2) A1(1,3,2) T2(5,5,2)	1	18	1.1333333	Success	None

Table-5.7

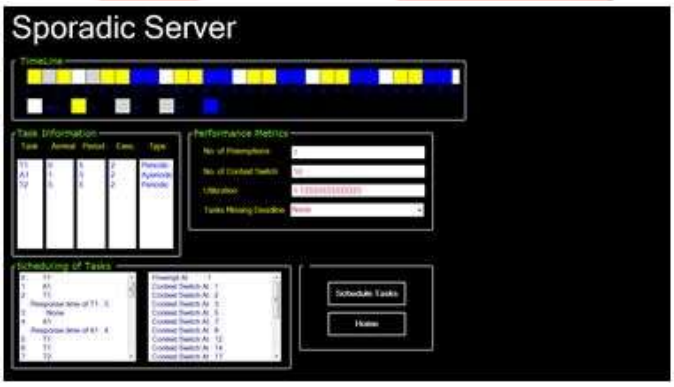


Fig-5.8

VI. CONCLUSION AND FUTURE WORKS

There are number of performance analysis tools available. Most of those provide few schedulers, only Cheddar is a tool that provides good support for multiple schedulers. Even though Cheddar gives best performance as compared to others, it does not deal very well with some of the performance metrics like utilization, schedulability test, etc. and limited to only few models mainly periodic tasks; so it doesn't support for aperiodic tasks.

By using the simulator, we can analyze the performance of the given scheduler for given set of tasks. It helps to find out which scheduler is perfect for which kind of set of tasks producing good results for the system.

Moreover, this simulator overcomes the drawbacks of previously designed simulators. It provides support for scheduling of periodic as well as aperiodic tasks. Also, it can be used for educational purpose. Hence it proves itself to be more efficient than other simulators.

REFERENCES

[1] C. L. Liu and J. W. Layland. "Scheduling algorithms for multiprogramming in a hard-real time environment", Journal of the ACM, vol 20 pp 46 - 61, January 1973.

[2] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behaviour. In Proceedings of the 11th Real time Systems Symposium, pages 166{171, 1989.

[3] J. Y. T. Leung and J.Whitehead, "On the complexity of priority scheduling of periodic, real time tasksPerformance Evaluation, 2(4):237{250, 1982.
[4] N. C. Audsley, A. Burns, R. I. Davis, K. W. Tindell, and A. J. Wellings, "Hard real time scheduling: The deadline monotonic approach", In Proceedings of the 8th IEEE Work-shop on Real time Operating Systems and Software, pages 133-137, 1991.
[5] M. Joseph and P. Pandya, "Finding response times in a realtime system", The Computer Journal 29(5), pages390-395, 1986.
[6] SorinManolache, "Schedulability Analysis of Real time Systems with Stochastic Task Execution Times", Department of Computer and Information science, IDA,LinkopingUniversity,pages {17 – 26},1988.
[7] RakeshTripathi .Early Stage Reliability and Design Assessment.LRDEBangalore.IEEE.
[8] M.-L. Yin, C. L. Hyde, and L. E. James.A petri-net approach for early-stage system-level software reliability estimation.Proceedings Annual Reliability and Maintainability Symposium, pages 100 – 105, 2000.
[9] C.Wohlin and P. Runeson. A method proposal for early software reliability estimation.IEEE, pages 156 – 163, 1992.
[10] S. Yacoub, B. Cukic, and H. H. Ammar. A scenario-based reliability analysis approach for component- based software.IEEE Transactions on Reliability, 53(4):465 – 480, December 2004.
[11] M. H. Klein, T. A. Ralya, B. Pollak, R. Obenza, and M. G. Harbour.A Practitioner's Handbook for Real time Analysis: A Guide to Rate Monotonic Analysis for Real time Systems. Kluwer Academic Publishers, 1993.
[12] X. Castillo, S.P. McConnel, and D.P. Siewiorek.Derivation and Calibration of a Transient Error Reliability Model.IEEE Transactions on Computers,31(7):658–671, July 1982.
[13] A. Campbell, P. McDonald, and K. Ray. Single event upset rates in space.IEEE Transactions on Nuclear Science, 39(6):1828–1835, December 1992.
[14] C.-J. Hou and K. G. Shin. Allocation of periodic task modules with precedence and deadline constraints in distributed real time systems. IEEE Transactions on Computers, 46(12):1338–1356, 1997.
[15] K. G. Shin, M. Krishna, and Y. H. Lee. A unified method for evaluating realtime
computer controllers its application. IEEE Transactions on Automatic Control, 30:357–366, 1985.
[16] H. Kim, A.L.White, and K. G.Shin. Reliability modeling of hard real time systems.In Proceedings 28th Int. Symp.on Fault-Tolerant Computing (FTCS-28), pages 304–313. IEEE Computer Society Press, 1998.
[17] M.B. Jones, D. Rosu, M. Rosu, CPU reservations and time constraints: efficient, predictable scheduling of independent activities, Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP '97),St. Malo, France, Oct, 1997.
[18] Labrosse, J.J, MicroC OS-II: The Real time Kernel (SecondEdition), CMPBOOKS, Manhasset NY, 2002.
[19] Fr´ed´ericRidouard, Pascal Richard, Francis Cottet," Negative results for scheduling independent hard real time tasks with self-suspensions", Proceedings of the 25th IEEE International Real time Systems Symposium ,2004.
[20] J. Goossens and P. Richard, "Overview of real time scheduling problems".
[21] GiorgioButtazzo, "Real time Operating Systems: Problems and Novel Solutions", In Proceedings of the 7th Int. Symposium on Formal Techniques in Real time and Fault-Tolerant Systems, 2002.
[22] BrinkleySprunt, "Aperiodic Task Scheduling for Real time Systems" , Submitted in Partial fulfillment of the Requirements for the Doctor of Philosophy Degree in Computer Engineering ,1990.
[23] Krishna C.M. and Shine K.G. Real time System Tata McGraw Hill 1997.
[24] J. Xu, L.Parnas, "Scheduling with release time, deadlinesprecedence and exclusion relation", IEEE Transaction on software Engineering, pages 360-69, Mar,1990.
[25] Jagbeer Singh, BichitranandaPatra, Satyendra Prasad Singh "An Algorithm to Reduce the Time Complexity ofEarliest Deadline First Scheduling Algorithm in Real-Time System", International Journal of Advanced Computer Science and Applications,Vol. 2, No.2, February 2011
[26] Martin Naedle, "A Survey of Real Time Scheduling tools", TIK-Report No. 72, November 1998
[27] F. Singhoff, J. Legrand, L. Nana, L. Marce, "Cheddar: a Flexible Real Time Scheduling Framework", EA 2215, University of Brest 20, av Le Gorgeu CS 93837, 29238 Brest Cedex 3
[28] Cyril Muller, M. Frank Singhoff, "Thesis on Scheduling methods for aperiodic tasks", Universite De BretangneOccidentale, February 2014
[29] Manish S. Raut, M.B Narnaware, "Analysis of Rate Monotonic Scheduling Algorithm on Multicore Systems", International Journal of Science and Research (IJSR), ISSN (Online): 2319-7064 Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438