Real-TimeTask Simulator for Scheduling

Saranga Mohan, Anju S Pillai, Dioline Sara

Abstract*: *Real-time systems are designed in such a way that all tasks need to guarantee its respective deadlines. This paper proposes a novel Real-Time Simulator suitable for simulating real-time scheduling algorithms without the support of a Real-Time Operating System (RTOS). This Simulator can act as an indispensable tool in teaching and research because of its capability in evaluating existing and new scheduling policies. The simulator can also be used to determine the task attributes and schedule the tasks according to different policies for static and dynamic priority systems. The same is equally effective in scheduling both synthetic and real-time workload.

Index Terms*: *Earliest Deadline First, Rate Monotonic, Real-time systems, Scheduling

I. INTRODUCTION

The principal concept of a run-time system is to deliver correct output at the correct point in time. The real-time systems will take the input from an outside event and performs a computation and produces output within a certain time. There are many types of real-time systems that have various timing constraints and uses different scheduling algorithms. Moreover, the comparison of real-time scheduling algorithms is a tedious process. For example, if a periodic event is taken, its period is important along with the other timing constraints set for the application. Similarly, if an aperiodic event is considered, its deadline is more important. The capability of a real-time system can be increased if it has both predictability and schedulability.

Real-time scheduling simulators will have a particular scheduling problem with an execution platform. They can also be used as a teaching tool to help learners to grasp knowledge related to system modeling. There are many simulators being implemented so far keeping this in mind. Some of the simulators of this kind are: RT Sim [1], Realtss [2], Cheddar [3], ERT Sim [4], STRESS [5], and Web-Enabled Framework for Real-Time Scheduler Simulator [6].

The main motivation for developing this real-time simulator is to save the time and effort put by many users who work in the real-time scheduling domain. The different policies used for scheduling is unique and is widely adopted by everyone who is doing their research. This work focuses on building software which is an initial step of the Operating System development. The simulator is tested for its functionality with both synthetic tasks and real-time tasks viz., LED blinking, LCD display, UART transmission, etc. in the ARM7 LPC2148 microcontroller. The runtime tasks are scheduled based on static or dynamic priority assignment policies and the various task attributes like execution time, period, and deadline are computed and the scheduling is performed at runtime.

Revised Manuscript Received on December 22, 2018.

Saranga Mohan, Department of Electronics & Communications, GITAM University, Bengaluru Campus, Bengaluru Rural district (Karnataka), India

Anju S Pillai, Department of Electronics & Communications, Amrita School of Engineering, Amrita Vishwa Vidya Peetham, Coimbatore (Tamil Nadu), India

Dioline Sara Department of Electronics & Communications, GITAM University, Bengaluru Campus, Bengaluru Rural district (Karnataka), India

II. SYSTEM MODEL

The task set contains periodic tasks $r = \{\tau_1, \tau_2, \tau_3, \ldots . \tau_n\}$ with period T_i, execution time C_i, priority P_i, and relative deadline D_i. Periodic tasks occur at regular intervals depending on their periods. The periodic task repeats the same schedule up to hyper period length, *H*. Run -time scheduling is a process to find the order of execution of simultaneous tasks on a processor. For a uni-processor system, there is only a single processor on which the tasks can be assigned and executed. In the case of uni-processor systems, total work load capacity should be within 1 (100%). The processor utilization is computed as:

$$U_{total} = \sum_{i=1}^{n} \frac{C_i}{T_i} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Where, U_{total} is the total processor utilization, C_i execution time of the task, T_i task time preiod and n is the total number of tasks in the system.

III. PROPOSED REAL-TIME SIMULATOR MODEL & FEATURES

In order to implement any existing or new scheduling policies into an RTOS, the operating system's kernel and internal structure have to be modified. Moreover, modifying the operating system kernel is a tedious and cumbersome process and this attributes to the reason for the implementation of only few scheduling algorithms. The proposed simulator is build using MATLAB functions which can be interacted with user using a GUI.

The highlight of the proposed simulator includes:

- User-friendly GUI and constitutes various modules which perform different functions which is very easy to implement and execute.
- Can be used as a researching tool because it aids teaching and research fraternity with little effort of real-time programming.
- Performs the testing and evaluation of scheduling policies and is used to determine the task attributes like execution time, time period and deadline.
- Also determines various scheduling aspects such as the number of preemptions, number of context switches, and energy consumption of the processor and feasibility analysis of different scheduling policies.

The salient features of RT Simulator comprise of:

A. Task Nature

The Real-time simulator has the flexibility to include task sets of

- Periodic
- Aperiodic
- Sporadic

The periodic task will occur at regular time intervals, whereas aperiodic task can arise at random time intervals. The sporadic task is similar to the aperiodic task in many cases, but there is an inter arrival time defined between the occurrence of next event.

B. Scheduling Algorithms

The allocation of resources and fixing the order of execution of various tasks in a system to guarantee time constraints, set for the application is the prime concern of scheduling. The main scheduling algorithms incorporated in the real-time simulator are listed below [7]:

- Rate Monotonic (RM) Scheduling Algorithm
- Earliest Deadline First (EDF) Scheduling Algorithm

A task set is schedulable by EDF if it satisfies the utilization test given below:

$$U_i = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

And RM schedulability is checked with the following utilization-based condition:

$$U_i = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

But this is only a sufficient condition and not a necessary condition.

C. Feasibility Analysis

Before the implementation of a particular application it is good to verify the feasibility of the method which is used. In this simulator model, the schedulability test added is Response Time Analysis (RTA) given by [8]:

$$R_i^{n+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil * C_j \ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

where:R_i represents the response time of task τ_i and C_j and T_j is the execution time and time period of the high priority task τ_j respectively.

D. Task Upload

The task needed for the execution is uploaded using this function. Some of the real-time tasks which can be run by ARM 7 LPC2148 microcontrollers are identified *viz.*, LED blinking with varying ON/OFF times, LCD display etc. These tasks are sorted in decreasing priority order by the chosen scheduling policy. They are then scheduled according to static and dynamic priority policies like RM and EDF policies. In the current model, tasks are assumed to have deadlines equal to the time periods of tasks. The source code for this task sets is written in embedded C language.

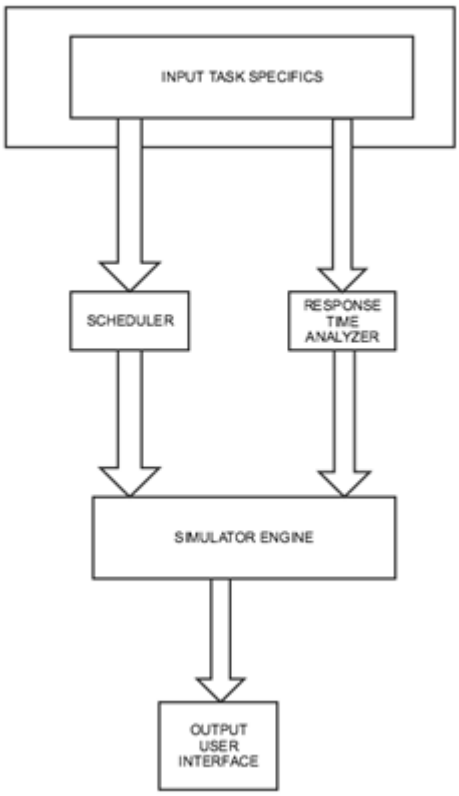

Fig. 1: Block diagram of the simulator

E. Task Generate

The proposed simulator is capable of generating synthetic task sets by standard task generation algorithms like *UUnifast* [9]. This algorithm is the most widely accepted one for task generation in this field. To execute this function, the number of inputs is given by the user and the pseudo code for algorithm development is given in Table 1:

Table 1: Pseudo code for *UUnifast* algorithm

```
function vectU = UUniFast (n, U)
sumU = U;
vectU = zeros (1, n);
    for i=1: n-1,
nextSumU = sumU. *rand^(1/(n-i));
vectU(i) = sumU - nextSumU;
sumU = nextSumU;
    end
vectU(n) = sumU;
```

F. View Graph

For better understanding, the simulation results are plotted and displayed by a view graph option. This feature makes this simulator a very friendly and understandable one. The simulator is built using MATLAB. The scheduling policies are added as *.m file.* It can be operated in Windows operating systems. Fig.1 shows the block diagram of the proposed real-time simulator. The input task specifies are the user inputs such as number of tasks and task nature; whereas scheduler will select the scheduling algorithms and analyzer will do the feasibility analysis.

IV. SIMULATION RESULTS

The following section details the features of the developed MATLAB based runtime simulator. Real-time task is selected through the task upload option, in which the execution time will be calculated by the simulator and parameters like period and deadline has to be entered by the user. All these task attributes will be displayed in a table in the GUI. Here, real-time tasks in an ARM7 LPC 2148 microcontroller like LED blinking, LCD display, UART program etc. are taken for scheduling. The uploaded task's worst-case execution time will be calculated by the simulator which is of microseconds range and will schedule according to the policy being selected. The task sets are generated using *UUnifast* algorithm. This can be selected through the task generate option. When the user selects the option of task generate, the input needed for the execution of algorithm will be asked. Here the user has to select the number of inputs and has to enter the utilization value. Then by clicking the *run* option, the task attributes like execution time, period and deadline will be displayed in the table. Then by selecting the scheduling policy needed the scheduling of non-real time tasks are done.

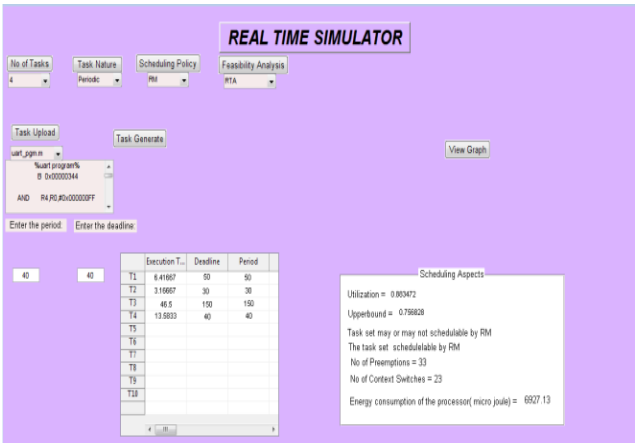

Fig.2. Scheduling of real-time tasks by RM

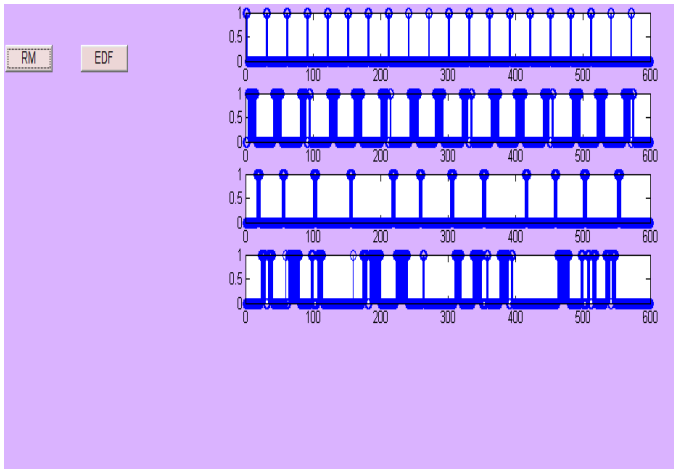

Fig. 3 Execution trace for real-time tasks by RM

In Fig: 2, the task execution scenario for a set of four input tasks with the following parameters taken is shown: worst-case execution time and time periods having the values of (6.4, 50), (3.1, 30), (46.5, 150) and (13.5, 40). The tasks when scheduled by RM policy are found feasible and the resulting execution pattern is shown in Fig.3.

After the selection of scheduling policy, for instance RM; the scheduling aspects such as utilization value, preemption count [10], and context switch count and energy consumption of the processor will be displayed. The execution trace can be viewed in a separate window while clicking the view graph option as in Fig.3.

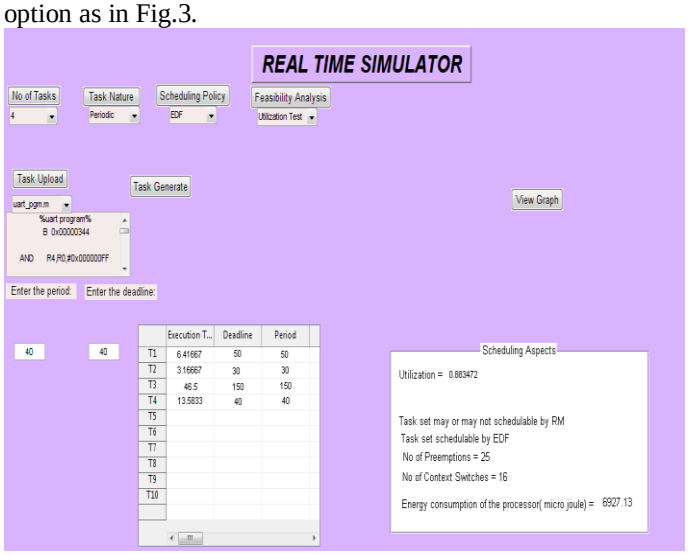

Fig. 4 scheduling of real-time tasks by EDF

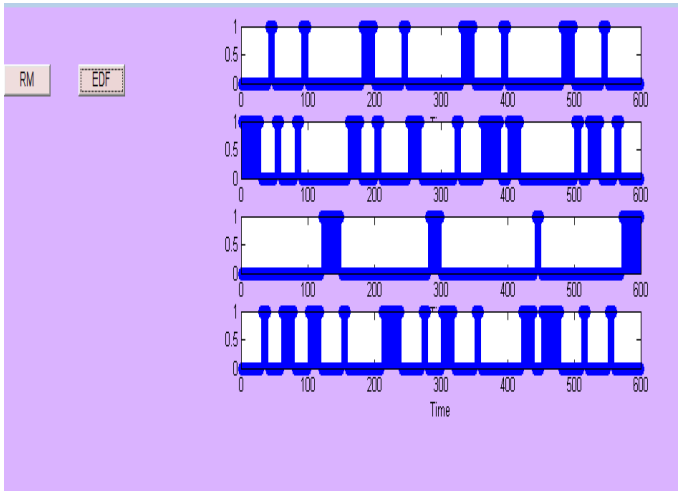

Fig. 5 Execution trace for real-time tasks by EDF

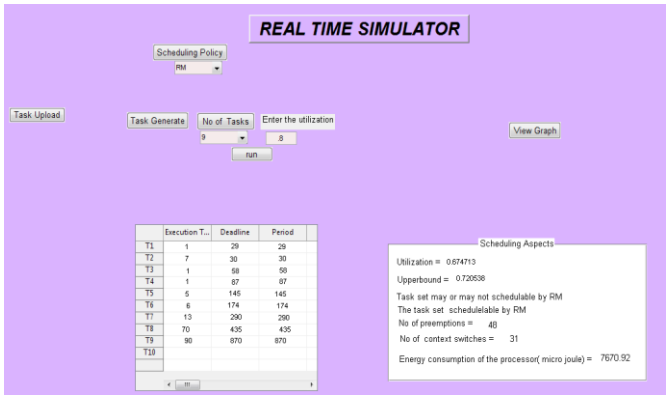

Fig. 6 scheduling of synthetic task by RM

Fig. 4 shows the execution of four real-time tasks with the same parameter values as that of RM. Here the scheduling policy used is EDF. When scheduled by EDF policy, the scheduling aspects are also calculated.

From the analysis of both the static and dynamic scheduling policy, it can be viewed in Fig.5 that the schedulability of EDF is more than RM as its utilization guarantee is 100%.

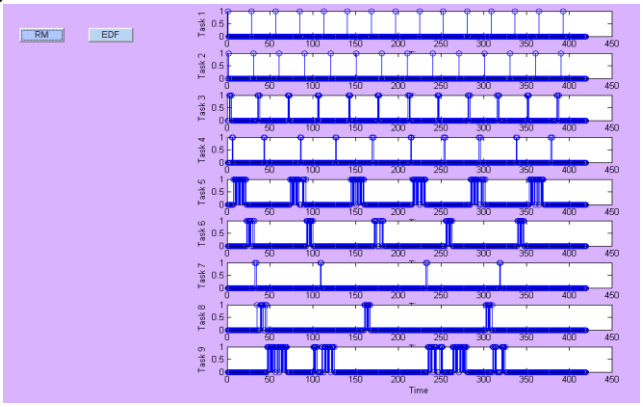

Fig.7 Execution trace for the synthetic task by RM

Fig.6 shows the scheduling of synthetic tasks in which the tasks sets are generated by the *UUnifast* algorithm. The tasks set are scheduled using a static policy like RM and its respective execution trace can be viewed in Fig. 7.

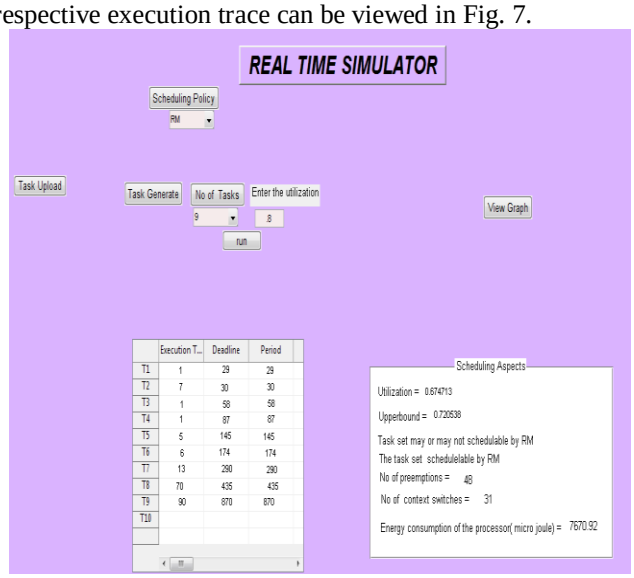

Fig.8 scheduling of synthetic task by EDF

Figure 8 shows the scheduling of synthetic tasks in which the tasks set are generated by the *UUnifast* algorithm and the tasks sets are scheduled using dynamic policy like EDF where in the execution trace can be observed in Fig.9.

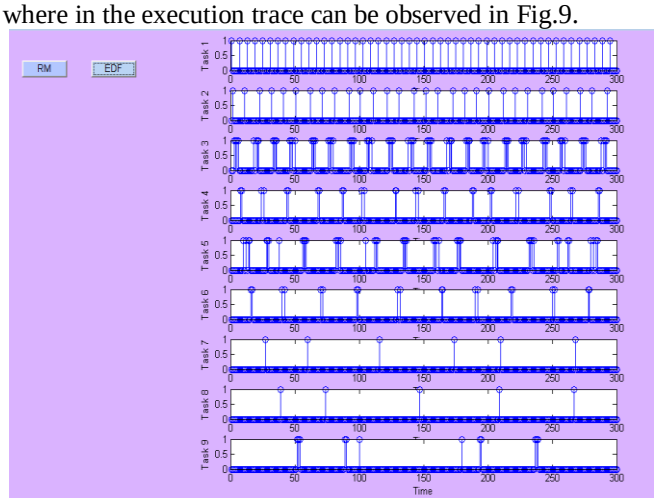

Fig 9 Execution trace for the synthetic task by EDF

Thus, with the support of the proposed real time task simulator, various task execution scenarios can be tested with different scheduling algorithms. The simulator also supports to learn and explore more about the feasibility of the input task set, knowing various attributes like: number of preemptions and context switches, energy consumed by the processor while executing the task set etc. Thus, the use of the proposed simulator is highly beneficial to the researcher and learners in the field to understand the fundamentals of the real time scheduling and have the flexibility to integrate any new scheduling algorithms of their choice to the proposed simulator with easiness.

V. CONCLUSION

This paper proposes a real-time simulator that can be used as software tool to test and evaluate the existing real-time scheduling algorithms. The proposed simulator is capable to integrate new scheduling algorithms as. mfile. This simulator is simple to implement and does not require complex integrations with real-time operating systems which are cumbersome. At present, it can be executed only in Windows operating systems and is fully integrated with MATLAB. Also, this simulator is integrated with ARM7 LPC2148 microcontroller to implement the real-time tasks with available scheduling policies at runtime. Enhancements can be done by the integration of more scheduling policies particularly for the systems consisting of sporadic and aperiodic tasks and can be interfaced to the multiprocessor domain.

REFERENCES

1. Eleardo Manacero, Marcelo B. Miola, and Viviane A. Nabuco. “Teaching real-time with a scheduler simulator”, in Proceedings of 31st ASEE/IEEE Frontiers in Education Conference, pp.15-19, Reno, NV, October 2001
2. Arnoldo Diaz, Ruben Batista and Oskardie Castro, “Realtss: a real-time scheduling simulator”, *4th International Conference on Electrical and Electronics Engineering*, pp 165-168, 2007.
3. F. Singhoff, J. Legrand, L. Nana, L. Marcé. “Cheddar: a Flexible Real Time Scheduling Framework.” *ACM SIGAda Ada Letters Edited by ACM Press*, vol 24, number 4, pp 1-8., 2004.
4. S Pillai, T. B. Isha, "ERTSim: An embedded real-time task simulator for scheduling", *IEEE International Conference on Computational Intelligence and Computing Research*, pp. 1-4, Dec 2013.
5. N. C. Audsley, A. Burns, M. F. Richardson, and A. J. Welling’s “STRESS: A simulator for hard real-time systems”, *Software Practice and Experience*, July 1994.
6. Yaashuwanth, C, Ramesh, R., "Web-Enabled Framework for Real-Time Scheduler Simulator (A Teaching Tool)”, Second International Conference on Computer Research and Development, pp.826-830, 7-10 May 2010.
7. Buttazzo, G.C, Rate monotonic vs. EDF, “*Journal of Real-Time Systems*”, pp 5-26, 2005.
8. Joseph, M., Pandya, P, “Finding Response Times in a Real-Time System”, The Computer Journal 29 (1986) 390395.
9. E. Biniand, G.C.Buttazzo, “Measuring the performance of schedulability tests,” *Real-Time Systems Journal*, vol.30, no.1-2, pp.129–154, 2005.
10. Abhilash Thekkilakattil, AnjuS. Pillai, RaduDobrin, Sasikumar Punnekkat, **“**Preemption Control using Frequency Scaling in Fixed Priority Scheduling,” *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2010.*