Middle-East Journal of Scientific Research 22 (4): 488-493, 2014
ISSN 1990-9233

DOI: 10.5829/idosi.mejsr.2014.22.04.21911

Web-Enabled Java Framework Task Scheduler Simulator (A Teaching Tool)

C. Yaashuwanth

Department of Information Technology,
Sri Venkateswara College of Engineering Chennai, India

Abstract: The main objective of this paper is to develop a new Web-enabled simulation framework: to study and evaluate the performance of various uniprocessor real-time scheduling algorithms for real-time scheduling. Task ID, deadline, priority, period, computation time and phase are the input task attributes to the scheduler simulator and chronograph imitating the real-time execution of the input task set and computational statistics of the schedule are the output. The proposed framework for the scheduler simulator is mainly developed to be used as a teaching tool. The Web-based deployment of the simulator enables the user a platform-, machine- and software-independent utilization of the technical resource.

Key words: Real-time · Scheduler · Simulator · Pre emptions and context switch

INTRODUCTION

A real-time computing system can be defined as a real-time application which is expected to respond to stimuli within some small upper bound on response time and any late result is cas bad as a wrong one. Thus correctness of a real-time system could be stated true with logical perfection in the computational result and its timeliness. For real-time embedded systems, the primary objective is to ensure that all tasks meet their deadlines. A schedule can be feasible or optimal: a feasible schedule orders tasks making them to meet all their deadlines; an optimal schedule is one which ensures that failures to meet task deadlines are minimized. The scheduler is responsible for coordinating the execution of several tasks on a processor. The scheduler may be preemptive or non-preemptive.

Study of Schedulers and Simulators: The purpose of task scheduling is to organize the set of tasks ready for execution by the processor system so that performance objectives are met [1]. The order of these tasks is called a 'schedule'. The scheduler for hard real-time systems must coordinate resources to meet the timing constraints of the physical system which implies that the scheduler must be able to predict the execution behavior of all tasks within the system [2]. Unless the behavior of a real-time system is predictable, the scheduler cannot guarantee that the computation deadlines of the system will be met. The requirement of predictability differentiates real-time systems from conventional computing environments and makes the scheduling solutions for conventional systems inappropriate for real-time systems. The scheduling theory provides numerous schedulability tests for each scheduling policies and locking protocols used in real-time systems [3-7]. Early work was limited to 'rate monotonic' task priorities with deadlines equal to periods and used a notion of 'processor utilization' to assess schedulability. More recent works have extended schedulability analysis to apply to any fixed-priority scheduling policy and to support arbitrary deadlines [8, 9].

Real-time simulation tools speed up the decision making processes during the selection of suitable scheduling algorithm for a real-time embedded application. They also stand as teaching tool helping learners of real-time system grasp the core ideas related to system modeling quickly. There have been various simulator frameworks created for this purpose, too [10-13]. The performance analyses of the above mentioned simulator frameworks were carried out and it is found that these simulators are not user friendly and hence the need for developing a new Web-based simulator framework was discovered.

Corresponding Author: C. Yaashuwanth, Department of Information Technology,
Sri Venkateswara College of Engineering Chennai, India.

The study of existing frameworks of simulation clearly reveals that each tool is better in its own way. Thus an appreciable combination of values of each of the tools is chosen and an earnest attempt has been made to make the proposed framework to be more flexible for the future users for trying different other combinations of evaluation criteria that may be of interest for different real-time resource capacities.

Need for Scheduler Simulator: From design perspective, real-time systems can be approached from different views. As an example, engineers prefer to deal with hardware control while computer scientists prefer to deal with the system modeling. The system modeling will explain how to model task interactions and how to allocate processor time for each task. The system modeling is burdensome because there are many different scheduling policies and scheduling problem is known as a strongly complex one. As a consequence of complexity, most learners feel that the scheduling theory is only a collection of rules that have to be memorized [14]. Therefore, much of the attention is not paid to the fact that the most important concept is not the exact description of a rule but what kind of conditions and problems are better suited for each rule. The foresaid misunderstanding can be solved, assigning jobs that require not only the resolution of a schedule but the experimentation with the problem. Although this can be done by hand, it has limitations due to the exponential growth in the resolution time with the problem size. The use of simulation technique would thus help circumvent this issue.

Significance of Web-based Deployment: There are vital reasons behind the implementation of the simulator as a Web-enabled framework and Web-based deployment as pointed out here. They are ease in deployment and enhancement of functionality. Anytime, anywhere access to users any computer with Web connectivity can be used for learning and teaching [15]. Easy access to users over the Internet since no extra hardware or software is required to access the application. The general block diagram of web enabled deployment is shown in Figure 4.1.

Advantages of proposed simulator: Platform independent access and use of the simulator for learning, User-friendly interface that requires minimal training / re-training, users will be able to access only the latest implementation of the simulator with no ambiguity of versions of the application, ease of maintenance from a

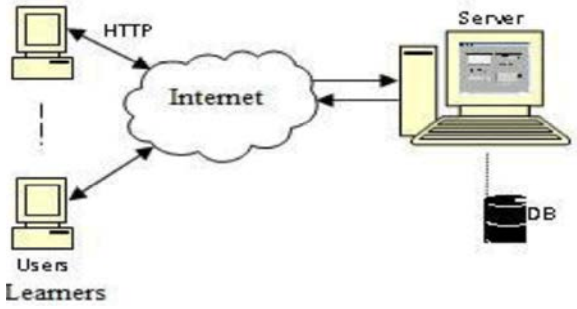

Fig. 4.1: Deployment of the simulator through Web

Programming / maintenance group perspective.and run Simulation to view the chronogram (timing diagram) and understand the way the tasks are scheduled in real-time using the selected scheduling policy.

Implementation Principles

Design of Proposed Web-Enabled Simulator: This section describes the development of the proposed simulator framework in java environment. A framework for evaluation of a scheduling algorithm must satisfy characteristics such as simplicity, compatibility with the PC platform usage of the standard operating system functions, accuracy of results, ease of use etc. Majority of these requests are aimed for use in the visual user interface that looks as shown in the Figure 3. The proposed Web-enabled scheduler simulator could be operated through a Web browser through a set of click-on and data input windows.

Scheduling algorithm evaluation and analysis tool performs the task definition, task sets generation, execution of selected algorithms, execution analysis of the execution and results are displayed. The performance evaluation of the real-time scheduling algorithms is carried out based on the results obtained through computational analysis. The proposed simulator has one input panel (scheduling input form) and two output panels (statistics form, graphical view panel). The sample code of interface between input and output panel is shown in Figure 5.2.

Scheduling Input Form: The task parameters (eg arrival time, service time etc) can be defined by the user in the scheduling input form. similarly the users can also select the scheduling algorithm. The control functions are also defined in the scheduling input form. The sample input code is shown in Figure 5.3 and the user view of the input panel is shown in Figure 5.4.

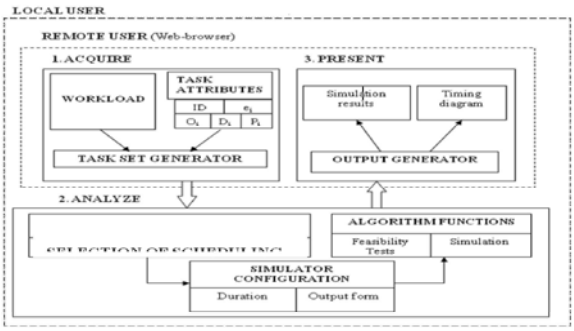

Fig. 5.1: Block Diagram of proposed simulator

```
public class scheduleAlgo extends Applet implements Observer
 {
  GraphicalForm graph;
  Status Report stats;
  Input Form input;
  Scheduler lesson;
  Vector Q;
  public void init()
          {
                  graph = new graphicalForm ();
                  graph.show();
                  graph.resize(300,300);
                  stats = new status Report();
                  stats.pack();
                  stats.show();
                  stats.resize(300,300);
                  input = new inputForm(this);
                  input.show();
                  input.resize(300,300);
                  lesson=null;
                  Q = new Vector(1,1);
          }
```

Fig. 5.2: Sample code for interface between three Panels of real time Simulator

```
public class inputForm extends Frame
   {
   TextArea proc,arriv,serv;
   Panel knobs;
   Choice alg;
   CheckboxGroup functions;
   Checkbox[] fun;
     Packet info;
   UserInterface vigil;
-----------------------------------------------------------------
fun[0] = new Checkbox("Clear",functions,false);
     fun [1] = new Checkbox("Run",functions,false);
     fun[2] = new Checkbox("Pause",functions,false);
     fun[3] = new Checkbox("Resume",functions,false);
     fun[4] = new Checkbox("Quit",functions,false);
-----------------------------------------------------------------
Panel labels = new Panel ();
     labels.setLayout(new BorderLayout());
     labels.add("West", new Label("Arrival Time:"));
     labels.add("Center", new Label("Process Name:"));
     labels.add("East", new Label("Service Time:"));
```

Fig. 5.3: Sample code for input form

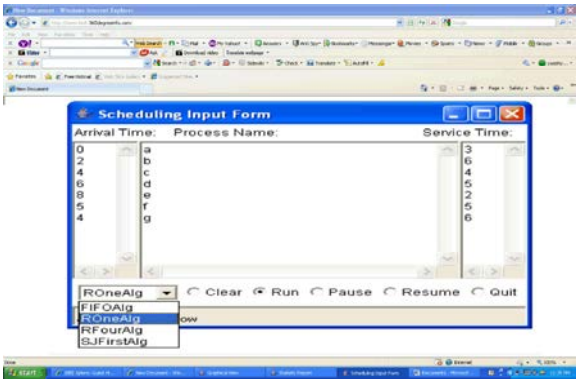

Fig. 5.4: Task scheduling input form in the simulator

The proposed simulator has been developed and deployed in the web platform and the users can access the simulator..

Statistics Form: The statistics form gives the statistics which is computed during the execution of tasks in a simulator. Criteria such as task waiting time, turn around time etc are computed and are visible to the user in this panel as shown in Figure 5.5 and 5.6.

The most successful scheduling algorithm for the periodic tasks scheduling is the one that has minimal response times, minimal number of tasks with missed deadlines and maximal resource utilization in the given workload and with other parameters.

The complete task model is too complex for implementation and some of the task parameters are hence ignored. In real-time systems two characteristics of tasks are considered to be of primary interest: criticality or importance; and timing. Task importance is frequently a subjective issue, whereas timing is objective. The essential timing attributes of tasks are deadline (T_D), worst-case computation time (Tc_w) and period (T_p),

Graphical View Panel: In this panel the task execution is shown in the form of an output graph to the user as shown in Figure 5.7 and 5.8

Hardware Co Processor: The target processor receives the values from the front end of the system This target processor controls all the devices on receiving the input, performs the commands and also displays the result in the front end of the system for user to view it. Figure 6.1 shows the block diagram of the hardware, Figure 6.2 shows the tasks in the hardware and figure 6.3 shows the overall setup.

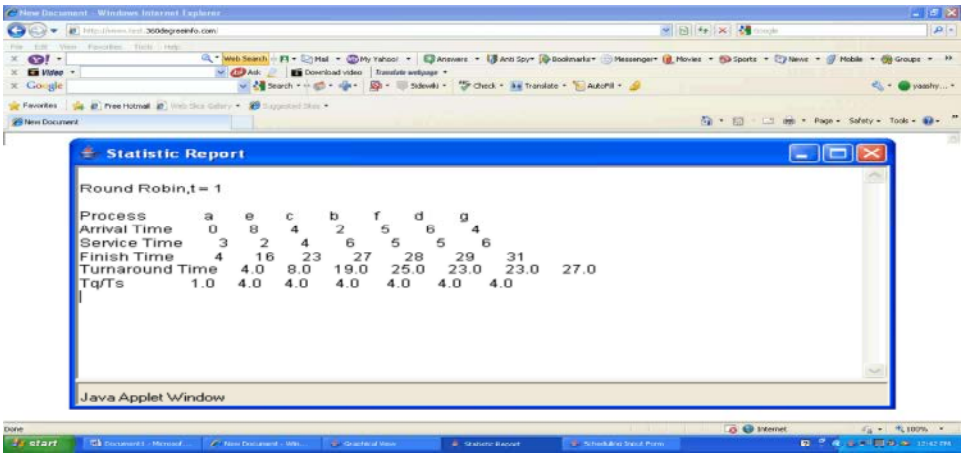

Fig. 5.5: Statistics Form

```
public statusReport() {
    super("Statistic Report");
    this.setLayout(new BorderLayout());
    pad = new TextArea(30,30);
    pad.setEditable(true);
    this.add("Center", pad);
  } // set display
  public void report(Vector R,String title) {
    pad.appendText("\n"+title+"\n\n");
    out = R;
    display();
  } // report statistics to notepad
-------------------------------------------------------
private void display()
        {
    onWork temp;
    P = "Process";
    A = "Arrival Time";
    S = "Service Time";
    F = "Finish Time";
    Tq = "Turnaround Time";
    Tqs = "Tq/Ts";
```

Fig. 5.6: Sample Code for statistics form

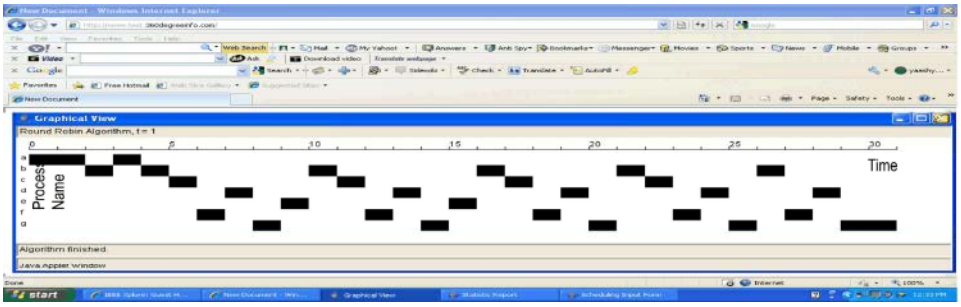

Fig. 5.7: Output (timing diagram)

```
public graphicalForm ()

        {
                super("Graphical View");
                board = new Canvas();
                StatusLine = new TextField(30);
                statusLine.setEditable(false);
                algTitle = new TextField(30);
                algTitle.setEditable(false);
                this.setLayout(new BorderLayout());
                add("North", algTitle);
                add("South", statusLine);
                add("Center", board);
        }
```

Fig. 5.8: Sample code for Graphical form

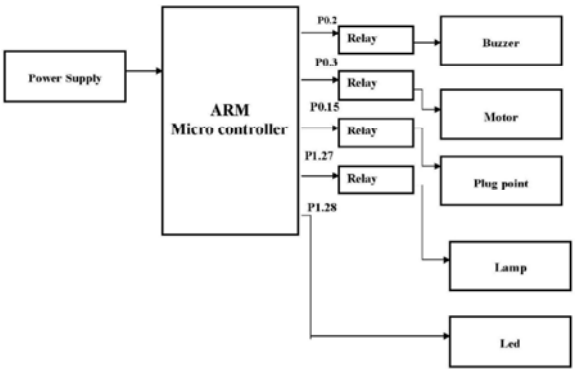

Fig. 6.1: Shows the block diagram of the hardware

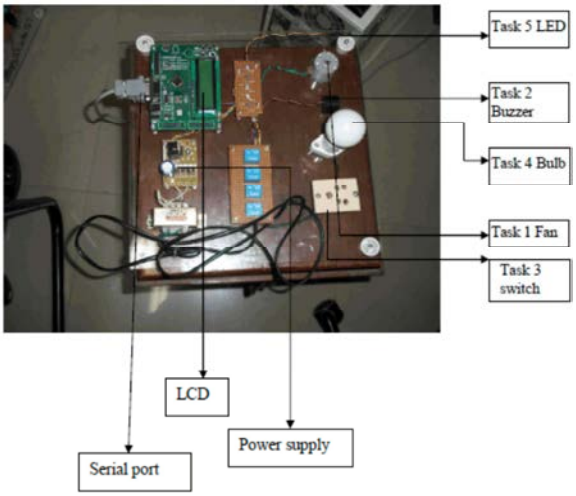

Fig. 6.2: Shows the tasks in the hardware

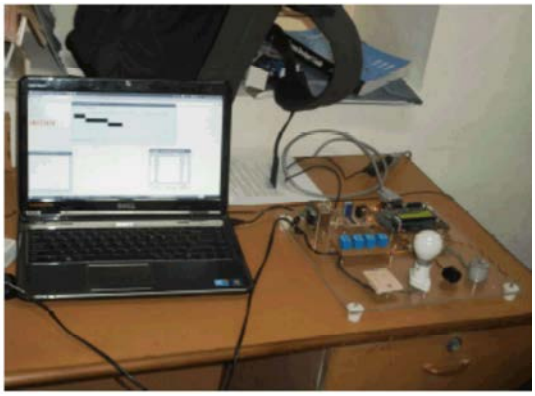

Fig. 6.3: Shows the overall setup

CONCLUSION AND FUTURE WORK

The proposed Web-enabled framework gives the developer the possibility to evaluate the schedulability of the real time application. Numerous benefits were quoted in support of the Web-based deployment technique employed. The framework which is proposed can be used as an invaluable teaching tool. Further, the GUI of the framework will allow for easy comparison of the framework of existing scheduling policies and also simulate the behavior and verify the suitability of custom defined schedulers for real-time applications. In addition, the real-time scheduler co-processor hardware can help the learners have a closer view of scheduling tasks in real-time hardware.

Future work includes integration of various other uniprocessor scheduling algorithm to the input frame, implementation of multiprocessor and aperiodic real-time schedulers in the simulator for exploring the full spectrum of real-time scheduling theory and development of co-processor architecture to complete the teaching tool.

REFERENCES

1. Aleardo Manacero Jr. Marcelo B. Miola and Viviane A. Nabuco, 2001. Teaching real-time with a scheduler simulator, in the proceedings of 31st ASEE/IEEE Frontiers in Education Conference, pp: 15-19.
2. Arnoldo Diaz, Ruben Batista and Oskardie Castro, 2007. Realtss: a real time scheduling simulator, in the proceedings of 4th International Conference on Electrical and Electronics Engineering, pp: 165-168.
3. Barbara Korousic-Seljak, 1994. Task scheduling policies for real-time systems, in Microprocessors and Microsystems, 18(9): 501-511.
4. Shih Chi-scheng, Lui Sha and Jane Liu, Scheduling Tasks with Variable Deadlines, in the proceedings of 7th IEEE International Symposium on Real Time Technology and Applications, pp: 120-122.
5. Cooling, J.E. and P. Tweedale, 1997. Task scheduler co-processor for hard real-time systems, in Microprocessors and Microsystems, 20(9): 553-566.
6. Bini Enrico and Giorgio C. Buttazzo, (Nov. 2004), Schedulability Analysis of Periodic Fixed Priority Systems, in IEEE Transactions on Computers, 53(11): 1462-1473.
7. Jan Blumenthal, Frank Golatowski, Jens Hildebrandt and Dirk Timmermann, 2002. Framework for Validation, Test and Analysis of Real-Time Scheduling Algorithms and Scheduler Implementations, in Proceedings of the 13th IEEE International workshop on Rapid System Prototyping, pp: 146-152.

8. Krishna, C.M. and K.G. Shin, 1997. Real-Time Systems, MIT Press and McGraw-Hill Company, pp: 5-150.
9. Ramamritham Krithi and John A. Stankovic. 1994 Scheduling Algorithms and Operating Systems Support for Real-Time Systems, in Proceedings of the IEEE, 82(1): 55-67.
10. Liu, C.L. and James W. Layland, 1973. Scheduling Algorithms for Multipro gramming in a Hard Real-Time Environment, Journal of the ACM, 20(1): 46-61.
11. National Instruments Corporation, 2005. LabVIEW™ Fundamentals.
12. Reinder J. Bril and J.L. Cuijpers, 2006. Analysis of hierarchical fixed-priority pre-emptive scheduling revisited, in Technische Universiteit Eindhoven (TU/e) CS-Report, pp: 06-36.
13. Sha, L., T. Abdelzaher, K. Arzen, A. Cervin, T.P. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky and A. Mok, 2004. Real time scheduling theory: A historical perspective, in Real-Time Systems, 28(2): 46-61.
14. Sha, L., R. Rajkumar and J.P. Lehoczky, 1990, Priority Inheritance Protocols: An Approach to Real-Time Synchronization, in IEEE Transactions on Computers, 39(9): 1175-1185.
15. Singhoff, F., J. Legrand, L. Nana and L. Marce, 2004, Cheddar: a Flexible Real Time Scheduling Framework, in the proceedings of ACM SIGADA'2004 International conference on Ada, pp: 1-8.