Realtss: a real-time scheduling simulator

Arnoldo Díaz, Ruben Batista and Oskardie Castro
Department of Computer Systems
Instituto Tecnológico de Mexicali, México
{adiaz, rbatista, ocastro}@itmexicali.edu.mx

***Abstract*—Real-time scheduling theory has shown an impressive evolution in the past few years. As a consequence of the intensive research done in this area a lot of new scheduling policies had been proposed to date. Nevertheless, just a few of such scheduling policies are available in existing Real-Time Operating Systems (RTOS). In this paper, we describe *Realtss*, an open source real-time scheduling simulator which is suited to simulate real-time scheduling algorithms without the need of implement them in a RTOS. Realtss is an invaluable teaching and researching tool since existing and new real-time scheduling policies can be easily evaluated.**

***Index Terms*—real-time scheduling, real-time simulation**

I. Introduction

Real-time scheduling theory has shown a transition from cyclical executive based infrastructure to a more flexible scheduling models such as fixed-priority scheduling, dynamic-priority scheduling, soft real-time applications, feedback scheduling or extended scheduling [20]. Nevertheless, just a few scheduling algorithms have been implemented in existing Real-Time Operating Systems (RTOS). In fact, recent studies show that almost every existing real-time operating system provides only POSIX-compliant fixed-priority scheduling [19] since it can be easily implemented in commercial kernels, even though real-time systems are commonly composed of hard and soft real-time tasks and in many cases these systems are better scheduled using dynamic-priority policies [1]. Also, it's been shown that dynamic-priority scheduling allows a better utilization of system resources [6].

In order to test and evaluate existing and new scheduling policies they could be integrated into a RTOS, but it is necessary to modify the operating system internal structure, which requires special programming skills. Furthermore, modifying the operating system internal kernel represents a difficult and time-consuming task, which is one of the reasons why so few scheduling policies have been already implemented.

Another alternative to test and evaluate scheduling policies had been proposed recently through a framework to implement new scheduling algorithms into a RTOS without modifying its internal structure [2], [9], [8]. Nevertheless this framework must be implemented in the RTOS, and the use of its API implies the use of advanced programming techniques.

In this paper we describe Realtss, an open source real-time scheduling simulator which is suited to simulate real-time scheduling algorithms without the need of implement them in a RTOS. Realtss is an invaluable researching tool since existing and new real-time scheduling policies can be easily implemented and their performance evaluated. Furthermore the proposed simulator can be used as a teaching tool since students can learn real-time scheduling without the need of knowing real-time systems programming.

The remainder of this paper is organized as follows: Section 2 gives a brief overview of the related work. In Section 3 we show and explain the main characteristics of Realtss. Section 4 presents an example of the use of the simulator. Finally, Sections 5 is for conclusions and future work.

II. Related work

The scheduler is an operating system module that implements a set of algorithms to allocate resources and to control access to shared resources among jobs [16]. In a real-time operating system, the scheduling algorithms implemented in the scheduler must guarantee the accomplishment of the task's deadlines.

To guarantee the accomplishment of the real-time system timing constraints a lot of feasibility tests have been proposed. They can be divided in polynomial time tests and exact tests. The first ones can be used for on-line guarantee of real-time applications. However, they provide only a sufficient schedulability condition, which may cause a poor processor utilization. On the other hand, exact tests provide a necessary and sufficient schedulability condition, but are too complex to be executed on line for large task sets since commonly the complexity of these tests is pseudopolynomial [5].

On the other hand, the performance of a real-time system scheduling algorithm can also be evaluated through the use of exhaustive simulation. Some real-time scheduling simulators have been proposed. Initially, they were build to address a particular scheduling problem or execution environment [15], [22], [23].

One of the first notable works was published in [3], where Audsley *et al.* proposed a collection of CASE tools for

1-4244-1166-1/07/$25.00 ©2007 IEEE.
IEEE Catalog Number: 07EX1762C
ISBN: 1-4244-1166-1
Library of Congress: 2007923398

analyzing and simulating the behavior of hard real-time safety-critical applications. The proposed simulator, called STRESS, included a simulation language to specify both the system environment and the task parameters. It included a graphical front-end for control and display, some feasibility tests and support for multiprocessing and networking. Since the simulator was built upon a simulation language, it imposed a limit in the scope of scheduling algorithms that could be simulated.

De Vroey *et al.* proposed in [7] a language for defining scheduling algorithms for hard real-time systems and a tool to simulate the behavior of such systems on a predefined task set. Their proposal was similar to STRESS but presented an extended language in order allow the simulation of newly devised scheduling policies.

Kramp *et al.* proposed FORTISSIMO in [14]. It was also based on STRESS. Their proposal was not a ready-to-run application but an open framework to facilitate the development of tailor-made real-time scheduling simulators for multiprocessor systems. FORTISSIMO was not a simulation application but provided a basic infrastructure to build a real-time scheduling simulator.

In [18] Manacero *et al.* presented a very interesting project named RTSIM, a set of real-time scheduling simulator tools aimed primarily to be used as a teaching tool. It is a collection of programming libraries written in C++ for simulating real-time control systems. RTSIM is still an active project and its code has been recently released as open source.

Finally, Jakovljevic *et al.* presented in [12] a research in application of object-oriented language (Java) in development of real-time system simulation. They described advantages and disadvantages of Java, and gives a critical overview of necessary modifications to make Java an acceptable choice for real-time systems.

The proposals presented so far required the use of a specific language to define the attributes of the tasks set and the simulation parameters. In contrast, the tool proposed in this paper does not required the use of a specific language to build or use the simulators. It provides a framework for developing real-time scheduling simulators and it is not limited by any simulation language. It is composed of a graphical front-end to use the scheduling algorithms and evaluate their performance. In the next section we present the architecture of Realtss.

III. REALTSS

Realtss is intended to be used to test and evaluate existing and new scheduling policies. It has been written in TCL and it is distributed as open source software. Its modular design allows the integration of additional scheduling algorithms seamlessly. New scheduling algorithms can be added as modules written in TCL, C or C++ and it is not limited by a particular simulation language. It can be executed in many operating systems, such as Linux or Windows.

Figure 1 shows the architecture of the proposed tool. Realtss has a modular design. The *Functions* module implements a set

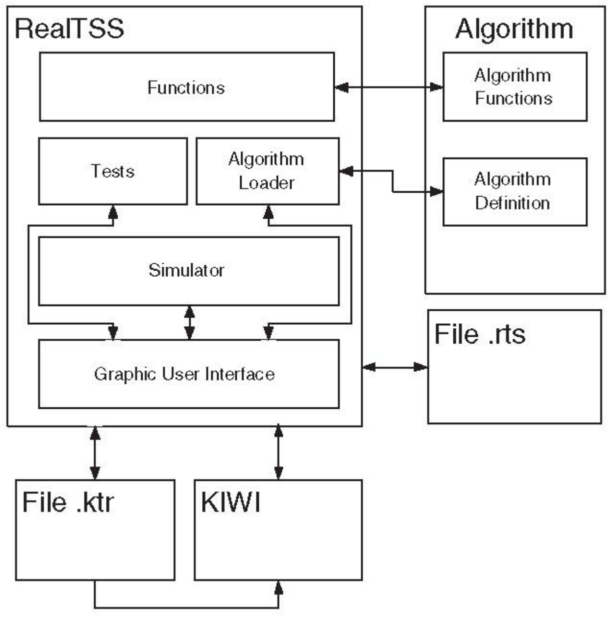


Fig. 1. Realtss Architecture

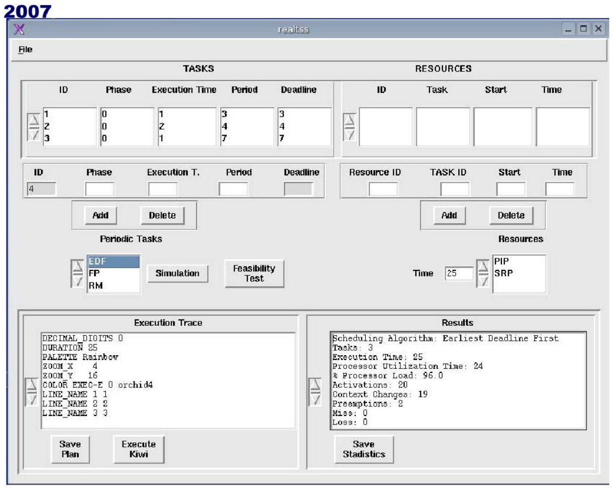
of functions to work with the defined tasks sets. The *Algorithm Loader* searches and loads the scheduling simulators modules that complaints with the definitions of the proposed framework. To add a new scheduling simulator is just needed to follow a simple definitions guidelines and to add the module into the *./algos* folder. The scheduling simulator module is composed of two files. The first one is a *.tcl* file. It contains the simulator itself and it can use any of the simulation functions already available in Realtss. The second one is a *.def* file and it includes the algorithm definitions in order to be used by Realtss. On the other hand, the *Tests* module includes the schedulability tests for the scheduling policies implemented. New schedulability tests can be easily added. The parameters of the tasks sets can be saved in a *.rts* file in order to be used in the future with any of the scheduling algorithms implemented in the simulator.

After a simulation run, some new files are generated. One of them includes the simulation results to be shown in the Realtss *Graphical User Interface*. The other one is a *.ktr* file, which in turn is used by the Kiwi program. Realtss is fully integrated with Kiwi [10], a graphic application which displays tasks execution trace logs. Results of the simulations are generated in a kiwi-compatible format in order to be displayed properly with the use of this tool. After the simulation test have been executed, the user can analyze the behavior of the tasks set once it has been scheduled with the scheduling policy selected. An execution chronogram can be visualized using the kiwi application.

The use of the simulator involves three steps most of the times: definition of parameters, simulation and results analysis. In the first stage the user defines the parameters of the tasks set. The parameters that can be defined are period, worst-case execution time, phase and deadline.

The real-time system to be simulated can be composed of hard

1-4244-1166-1/07/$25.00 ©2007 IEEE.
IEEE Catalog Number: 07EX1762C
ISBN: 1-4244-1166-1
Library of Congress: 2007923398



ID	Phase	Execution Time	Period	Deadline
1	0	1	3	3
2	0	2	4	4
3	0	1	7	7

ID	Phase	Execution T.	Period	Deadline
4				

Resource ID	TASK ID	Start	Time

Periodic Tasks: EDF, FP, RM

Time: 25

Resources: PIP, SRP

```
DECIMAL_DIGITS 0
DURATION 25
PALETTE Rainbow
ZOOM_X    4
ZOOM_Y   16
COLOR EXEC-E 0 orchid4
LINE_NAME 1 1
LINE_NAME 2 2
LINE_NAME 3 3
```

```
Scheduling Algorithm: Earliest Deadline First
Tasks: 3
Execution Time: 25
Processor Utilization Time: 24
% Processor Load: 96.0
Activations: 20
Context Changes: 19
Preemptions: 2
Miss: 0
Loss: 0
```

Fig. 2. Realtss Graphical User Interface

and soft real-time tasks and shared resources. Additionally, if the scheduling policy involves the use of resources (mutexes) their parameters can also be defined accordingly. Figure 2 shows the Realtss GUI.

We have already integrated some scheduling algorithms into Realtss:

- POSIX-complaint Fixed-Priorities [11]
- Rate Monotonic (RM) [17]
- Earliest Deadline First (EDF) [17]
- Deadline Monotonic (DM) [17]
- Priority Inheritance Protocol (PIP) [21]
- POSIX-complaint version of the Priority Ceiling Protocol (PCP) [13]
- Stack Resource Protocol (SRP) [4]

Scheduling policies can be selected using a selection list. A simple mechanism has been devised to integrate additional scheduling policies into Realtss. New scheduling modules build following a simple guidelines must be in a proper directory in order to appear in the simulator's selection list.

Once the parameters of the tasks set have been defined and the scheduling policy has been selected, a feasibility test or a simulation test can be executed for a given simulation time.

Besides the graphical data displayed, a lot of valuable information is provided by Realtss. This information is very useful at evaluating the performance of the scheduling algorithm used. The information provided includes:

- Processor utilization
- Response time
- Tasks waiting time
- Deadline misses
- Overruns

Since the parameters of the tasks can be saved into a file, they can be simulated using different scheduling algorithms and their performance can be evaluated.

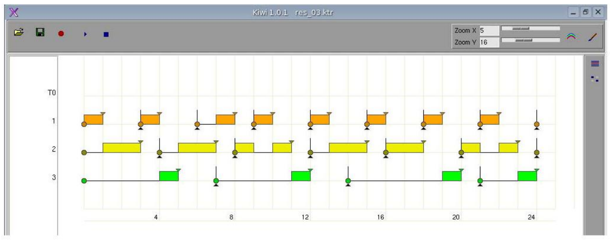


Fig. 3. Execution chronogram displayed using the Kiwi application

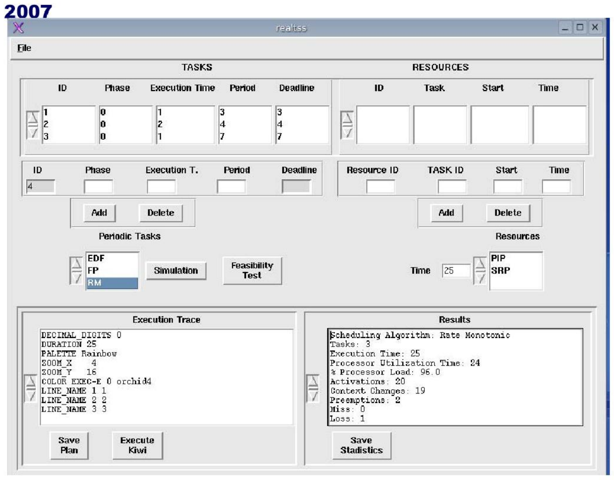
IV. Example

Figure 2 shows a real-time system composed of three hard real-time tasks. The parameters of the tasks used in the example are: τ_1 = (3, 1), τ_2 = (5, 2) and τ_3 = (8, 1). The first parameter represents the task's period and the second one its worst-execution time. In the example, the scheduling policy used is Earliest Deadline First. EDF is a dynamic scheduling policy where the task with the earliest deadline has the highest priority. The simulation time used in the example was set to 25 units. Simulation results are shown in the Realtss GUI, as it can bee seen in Figure 2. On the other hand, Figure 3 shows the execution chronogram of the tasks set scheduled using EDF as displayed by the Kiwi application.

Realtss can be used to evaluate the performance of a scheduling algorithm and to compare different scheduling policies among them. In order to illustrate the goodness of the proposed tool, the tasks set used in the previous example was also scheduled using the RM algorithm. RM is static scheduling policy where the task with the shortest period has the highest priority. Figure 4 shows the simulation results and it can be seen that in the simulation period a task missed its deadline once. This fact can also be viewed graphically in Figure 5, where τ_3 didn't executed its first activation and consequently missed its deadline. This example shows that EDF is able to schedule a tasks set that RM can not schedule correctly.

As shown in the example, Realtss can be used as a research tool since it allows the evaluation of the performance of scheduling policies and to compare them using the same or different tasks sets. Furthermore, it can be used as a teaching tool. For instance, a student can graphically view the difference between a static scheduling policy against a dynamic one. In the example discussed in this section, it can be seen how when using a static scheduling algorithm the priority of tasks never change. On the other hand, when a dynamic scheduling algorithm is used the priorities of tasks may change in time. To illustrate it, in Figure 5 τ_1 always preempts τ_2 and τ_3 since the priority assigned to τ_1 by the scheduling algorithms never changes (τ_1 is always the most priority task). In contrast, Figure 3 shows how τ_2 preempts the third activation of τ_1 since when using dynamic scheduling algorithms the priority assigned to every task may change at a given scheduling time (τ_1 is not always the most priority task).

1-4244-1166-1/07/$25.00 ©2007 IEEE.
IEEE Catalog Number: 07EX1762C
ISBN: 1-4244-1166-1
Library of Congress: 2007923398



TASKS

ID	Phase	Execution Time	Period	Deadline
1	0	1	3	3
2	0	2	4	4
3	0	1	7	7

RESOURCES

ID	Task	Start	Time

ID	Phase	Execution T.	Period	Deadline
4				

Resource ID	TASK ID	Start	Time

Time 25

Execution Trace

```
DECIMAL_DIGITS 0
DURATION 25
PALETTE Rainbow
ZOOM_X    4
ZOOM_Y    16
COLOR EXEC-E 0 orchid4
LINE_NAME 1 1
LINE_NAME 2 2
LINE_NAME 3 3
```

Results

```
Scheduling Algorithm: Rate Monotonic
Tasks: 3
Execution Time: 25
Processor Utilization Time: 24
% Processor Load: 96.0
Activations: 20
Context Changes: 19
Preemptions: 2
Miss: 0
Loss: 1
```

Fig. 4. Example using Rate Monotonic scheduling policy

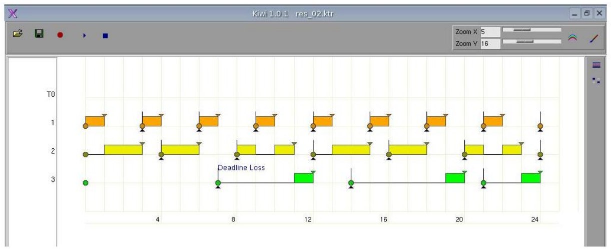


Fig. 5. Kiwi display of the RM example

V. Conclusions and Future Work

Even though a lot of scheduling algorithms have been proposed just a few of them are available to implement real-time systems applications. In order to use, test and evaluate a scheduling policy it must be integrated into a Real-Time Operating System, which is a complex task. Simulation is another alternative to evaluate a scheduling policy. Unfortunately, just a few real-time scheduling simulators have been developed to date and most of them require the use of a specific simulation language.

In this paper we have presented Realtss, an open source real-time scheduling simulator. Realtss is aimed to be used to test and evaluate existing and new real-time scheduling algorithms. It is composed of a graphical front-end to set task's parameters and to view simulation results. Its modular design allows the integration of additional scheduling policies seamlessly. New scheduling algorithms can be added as modules written in TCL, C or C++ and are not limited by a particular simulation language. It can be executed in many operating systems, such as Linux or Windows.

On the other hand, the simulation information provided by Realtss allows the evaluation and comparison of the performance of different scheduling algorithms. Its simplicity of use and its fully integration with Kiwi, a a graphic application which displays tasks execution trace logs, makes Realtss an invaluable teaching tool.

Future work includes the integration of more scheduling policies, in particular to schedule real-time systems constituted of periodic and aperiodic tasks with shared resources. Also, the support for multiple processors and networking.

References

[1] Abeni, L. and Buttazzo, G. Integrating multimedia applications in hard real-time systems. In *Proceedings of the 19th IEEE Real-Time Systems Symposium*, pages 4–13, Madrid, Spain, Dec 1998.
[2] Aldea, M. and Gonzalez-Harbour, M. Posix-compatible application-defined scheduling in marte os. In *Proceedings of the 13th Euromicro Conference on Real-Time Systems*, pages 67–75, June 2001.
[3] N. C. Audsley, Alan Burns, M. F. Richardson, and Andy J. Wellings. Stress: a simulator for hard real-time systems. *Software - Practice and Experience*, 24(6):543–564, June 1994.
[4] Baker, T. P. Stack-based scheduling of real-time processes. *Real-Time Systems Journal*, 3(1):67–99, Mar 1991.
[5] Bini, E. and Buttazzo, Giorgio C. Schedulability analysis of periodic fixed priority systems. *IEEE Transactions on Computers*, 53(11), Nov.
[6] Buttazzo, G.C. Rate monotonic vs. edf: Judgment day. *Journal of Real-Time Systems*, 29(1):5–26, Jan 2005.
[7] Stephane de Vroey, Joel Goossens, and Christian Hernalsteen. A generic simulator of real-time scheduling algorithms. In *Proceedings of the 29th Annual Simulation Symposium*, pages 242–249, New Orlean, LA, April 1996.
[8] Diaz, A., Ripoll, I., and Crespo, A. Extending the posix-compatible application-defined scheduling model. In *Proceedings of the 26th IEEE Real-Time Systems Symposium (WiP)*, Miami, Fl, Dec 2005.
[9] Diaz, A., Ripoll, I., and Crespo, A. A library framework for the posix application-defined scheduling proposal. In *Proceedings of the 2nd IEEE International Conference on Electrical and Electronics Engineering*, pages 21–26, Mexico City, Sep 2005.
[10] Agustin Espinoza. Kiwi user guide. Technical report, Universidad Politecnica de Valencia, 2003. Available on-line at http://www.dsic.upv.es/users/ia/sma/tools/kiwi/index.html.
[11] IEEE Std 1003.1, 2004 Edition. *The Open Group Technical Standard Base Specifications, Issue 6. Base Definitions*. Institute of Electrical and Electronic Engineers and The Open Group, Apr, 2004.
[12] G. Jakovljevic, Z. Rakamaric, and D. Babic. Java simulator of real-time scheduling algorithms. In *Proceedings of the 24th International Conference on Information Technology Interfaces*, volume 1, pages 411–416, Cavtat, Croatia, June 2002.
[13] Klein, M.H. et. al. *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis of Real-Time Systems*. Kulwer Academic Publishers, 1993.
[14] Thorsten Kramp, Matthias Adrian, and Rainer Koster. An open framework for real-time scheduling simulation. *Lecture Notes in Computer Science*, 1800:766+, 2000.
[15] Jane W. S. Liu, Juan-Luis Redondo, Zhong Deng, Too-Seng Tia, Riccardo Bettati, Ami Silberman, Matthew Storch, Rhan Ha, and Wei-Kuan Shih. Perts: A prototyping environment for real-time systems. In *Proceedings of the 14th IEEE RealTime Systems Symposium*, pages 184–188, Raleigh-Durham, North Carolina, Dec 1993.
[16] Liu, J.W.S. *Real-Time Systems*. Prentice Hall, 2000.
[17] Liu, J.W.S. and Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61, May 1973.
[18] 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*, pages 15–19, Reno, NV, October 2001.
[19] Ripoll I. et al. Rtos state of the art analysis. Technical report, OCERA Project, 2003.
[20] Sha, L., Abdelzaher, T, Arzen, K., Cervin, A., Baker, T.P., Burns, A., Buttazzo, G., Caccamo, M., Lehoczky, J., and Mok, A. Real time scheduling theory: A historical perspective. *Real-Time Systems*, 28(2):46–61, Nov 2004.
[21] Sha, L., Rajkumar, R., and Lehoczky,J.P. Priority inheritance protocols: An approach to realtime synchronisation. *IEEE Transactions on Computers*, 39(9):1175–1185, Sep 1990.
[22] Alexander D. Stoyenko. A schedulability analyzer for real-time euclid. In *Proceedings of the 8th IEEE Real-Time Systems Symposium*, pages 218–227, San Jose, CA, Dec 1987.
[23] H. Tokuda and C. W. Mercer. Arts: a distributed real-time kernel. *ACM SIGOPS Operating Systems Review*, 23(3):29–53, July 1989.

1-4244-1166-1/07/$25.00 ©2007 IEEE.
IEEE Catalog Number: 07EX1762C
ISBN: 1-4244-1166-1
Library of Congress: 2007923398