Teaching Real-Time with a scheduler simulator

Aleardo Manacero Jr., Marcelo B. Miola, Viviane A. Nabuco

Abstract—*In this paper we describe a scheduler simulator for real-time tasks, RTsim, that can be used as a tool to teach real-time scheduling algorithms. It simulates a variety of preprogrammed scheduling policies for single and multi-processor systems and simple algorithm variants introduced by its user. Using RTsim students can conduct experiments that will allow them to understand the effects of each policy given different load conditions and learn which policy is better for different workloads. We show how to use RTsim as a learning tool and the results achieved with its application on the Real-Time Systems course taught at the B.Sc. on Computer Science at Paulista State University - Unesp - at Rio Preto.*

Index Terms—*hard real-time tasks, real-time scheduling, simulators*

Introduction

Real-time systems comprises a somewhat large body of knowledge. It 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 modelling. We adopt the second approach since our students come from a computer science major.

From this approach, the relevant subjects become how to model task interactions and how to allocate processor time for each task. While the first subject can be easily performed through several techniques, like timed petri nets, and implemented by simple concurrency mechanisms, the second is burdensome because there are many different scheduling policies and the scheduling problem is known as a strongly complex one (usually falling into the NP-complete category).

As a consequence of this complexity, most students feel that this part of the subject is only a collection of scheduling rules that have to be memorized. Therefore, they do not pay attention to the fact that the most important concept is not the exact description, or execution, of a rule but what kind of conditions and problems are better suited for each rule.

This misunderstanding can be solved if the instructor leads the class in the right direction, 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. Our simulator can be used to circumvent this problem without the need for commercial products.

RTsim is a simulator of real-time scheduling algorithms developed at Unesp - Rio Preto. Initially it was designed to be a tool to aid designers of real-time software to decide which scheduling policy is the most effective for their case. However, one year after the conclusion of its prototype we found out that it also was a great tool to teach scheduling policies from a modelling perspective. The goals were redefined and now RTsim project is entirely devoted to the built of a teaching-aid tool.

In the next section we introduce the RTsim project, presenting a brief history of its development and status. Along this historic review we provide some terminology in order to establish a common pattern to the reader. We follow this with a thorough description of the Real-Time Systems course and how we use RTsim inside it. Finally, we conclude pointing out some of the results observed with its application in the past three years.

The RTsim project

This project started in 1996, as a work made by an undergraduate student for the Capstone Design course. As we already stated, its primary goal was to be a tool that could be used by a systems designer working with real-time problems. That version only simulated five single-processor algorithms. This restriction is not a problem since the algorithms were chosen in order to provide a good range of applications. The choices made at that time proved to be good enough, since we continue using the same algorithms for single-processor systems.

The single-processor algorithms are the **Generalized Rate Monothonic** (GRM) algorithm, used for periodic tasks, the **Sporadic Server** and the **Deferrable Server**, that are used for aperiodic tasks, and the **Priority Inheritance** and the **Priority Ceiling** protocols, that are used for tasks with critical sections [5], [11], [12]. As one can see, these algorithms provide some flexibility on the problem to be simulated that is right for a teaching tool, where the goal is to allow sound understanding about the mainstream scheduling algorithms on real-time systems.

After the conclusion of that prototype, the project was temporarily abandoned until 1998, when the course on real-time systems was taught for the first time in our Computer Science major. By then, the instructor noticed that the simulator could be an interesting tool when he was teaching the scheduling algorithms. With that experiment, it was possible to detect some missing functionalities, which led to constant upgrades since then. Table I lists some of them and their status in the current version of RTsim.

Today RTsim project involves five people among faculty

Manacero and Miola are with Department of Computer Science and Statistics - Unesp, São José do Rio Preto, Brazil. E-mails: aleardo,mbmiola@dcce.ibilce.unesp.br.

Nabuco is now with ICEC, São José do Rio Preto, Brazil. E-mail: viviane.nabuco@icec.com.br

Function	Status on current version
Simulation of multi-processors	Five algorithms are implemented
Simulation of algorithms that were not preprogrammed	Simple algorithms can be “learned” by RTsim
Automatic comparison between similar algorithms	Performed for preprogrammed algorithms that are applied for the same class of problems
Help on-line	Help is active for most of the operations
Schedule provided by an user being compared step-by-step with RTsim’s schedule	Under implementation

TABLE I
MISSING FUNCTIONALITIES ON RTSIM’S ORIGINAL PROTOTYPE

and grad and undergrad students. The work goes on the direction of finishing all the functionalities mentioned on Table I, plus the porting of its interface to the Java language, the inclusion of other topics on real-time systems, and the addition of the ability of adapt itself to different learning styles.

The use of RTsim is performed through a set of click-on and data input windows. We will not describe its operation in this paper because our goal is the description of its use as a teaching tool, what can be done without the knowledge about what to click in order to get a schedule plot. Some documents (in portuguese) describe its operation, and a version in English is under review [3], [7], [8].

Teaching with RTsim

The first application of RTsim as an aid to teach real-time schedulers was made with a quite simple prototype. After that experiment two other classes went through the Real-Time Systems course, both using different versions of the software, since it is under constant upgrade. Although it is not a finished piece of software it is possible to verify the effects of its application on students enrolled in the course.

For the sake of clarity, we have first to describe the Real-Time Systems course, since different views can be given to this area of knowledge. Our course aims a better understanding of how real-time systems software works and what is needed to provide such operation. Table II lists the topics taught during the course and are spread along a four months period.

Looking at the topics taught in the course one can see that topics 4 and 5 may use the tool. We will describe now how this can be done and what are the effects of doing so.

	Topic
1	Basics of real-time, clocks, time relationships
2	Modelling real-time applications
2.1	Modelling using temporized petri-nets
2.2	Modelling using formal description languages
3	Introduction to RTOS
4	Schedulers for RT systems
5	Load sizing, balancing and stability
6	Embedded systems

TABLE II
REAL-TIME SYSTEMS COURSE CONTENTS

Teaching schedulers

This is the most obvious use for RTsim since it is a simulator of scheduling algorithms. The efficiency of such use is heavily dependant of the algorithms taught by the instructor since the tool has only a small set of preprogrammed scheduling algorithms. The algorithms that can be promptly simulated are separated into algorithms for single and multiprocessor systems.

For single processor systems we use the five algorithms mentioned earlier. For multiprocessor systems the algorithms that are already preprogrammed are the **miopic**, the **bidding**, the **foccused bidding**, the **split** and the **module-allocation** [2], [6], [9], [10]. We will not describe them here since they are not in the scope of this paper.

For both systems, single and multiprocessor, the use of RTsim is similar. The instructor teaches the algorithms and assigns lab works to the students. These works assume several distinct forms, such as:

- Scheduling specific sets of tasks with random occurrence instants for the aperiodic tasks;
- Scheduling specific sets of tasks with specific occurrence instants for the aperiodic tasks;
- Scheduling specific sets of tasks for two similar algorithms.

These assignments provide data for the students to compare the efficiency of each algorithm for different restrictions, and verify the influence of overall workload, or of each task’s load, on the scheduling. The results from RTsim come in graphical and written forms. In the written form, it generates a series of files containing the output data from the schedule, which are rather hard to be read but provide details not observable in the graphical form.

In the graphical form RTsim generates a gantt chart with the schedule, as shown on Figure 1 for a comparison between the sporadic and deferrable server algorithms. On that figure, one sees when each task instance occupied the cpu. This window also enables the generation of a postscript file, containing the chart, and a zoom of part of the schedule, providing a closer view on a time interval where some interesting event

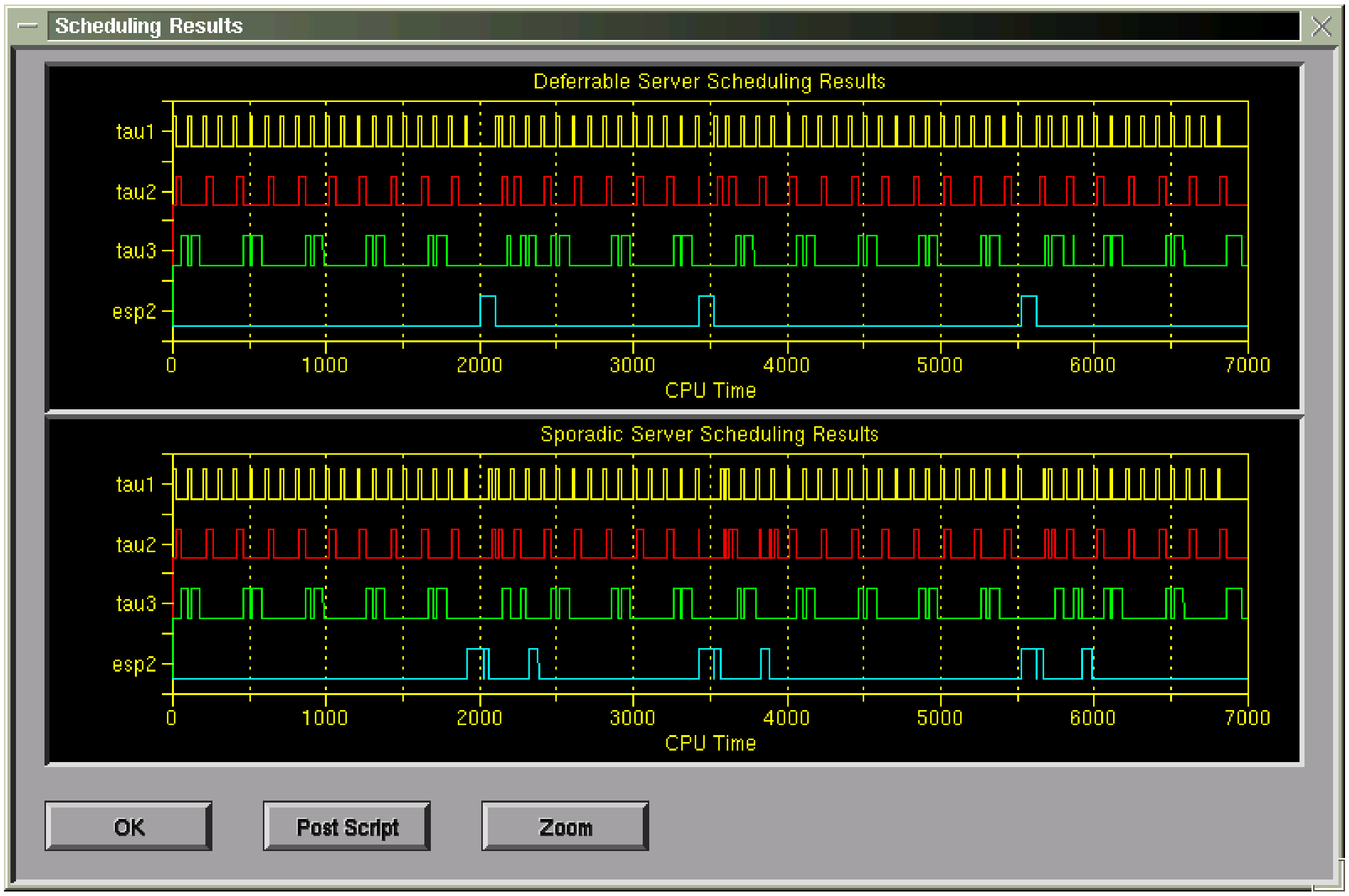


Fig. 1. Gantt chart for a simulated schedule comparing two algorithms.

took place.

Following the gantt chart, the simulator generates one (sometimes two) table(s). The first one, appearing just when at least one task misses its deadline, is shown in Figure 2. It provides the user with information about absolute delay, ratio between delay and the task load and between delay and the deadline for tasks that missed the deadline. In all three cases RTsim gives the average and the worst values and the standard deviation. It also shows the number of tasks that lost their deadlines and the number of scheduled tasks. Since these values do not determine which tasks missed their deadlines, the user may check for specific tasks if necessary.

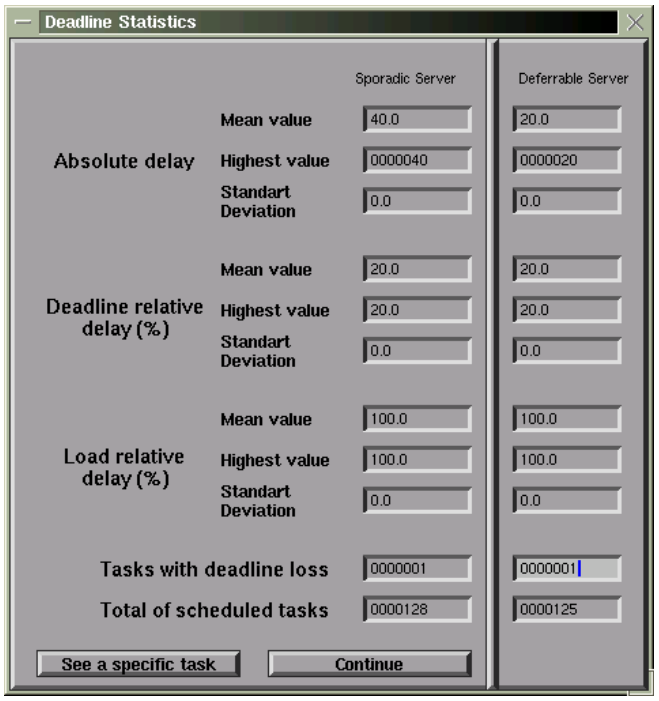
The second table (Figure 3) appears in every execution, showing the average values for performance parameters, such as turnaround and waiting times, and task and system laxities. The system also provides these informations for specific tasks if demanded. With such data it is possible to compare the efficiency of a given algorithm when applied to a given set of restrictions.

The figures presented here depict the results for a simulation comparing two algorithms. For the simulation of a single algorithm, only one plot will be shown and only the first column of each table will appear. For multiprocessor algorithms the results are separated for each specific node.

If the instructor needs, or one student wants, to test different algorithms, he/she will be capable to do these simulations for simple algorithms. That is, RTsim is able to simulate scheduling algorithms that are guided by a small set of combinations of standard parameters for real-time tasks. In this case the user will formulate the algorithm through a series of queries about relevant parameters, such as deadline or laxity, their relationships, and, if necessary, a simple equation that should be evaluated in order to find the task that has the best value (and will be dispatched) at every decision point. After the insertion of all scheduling rules, RTsim will provide the same results that it would provide for the preprogrammed algorithms.

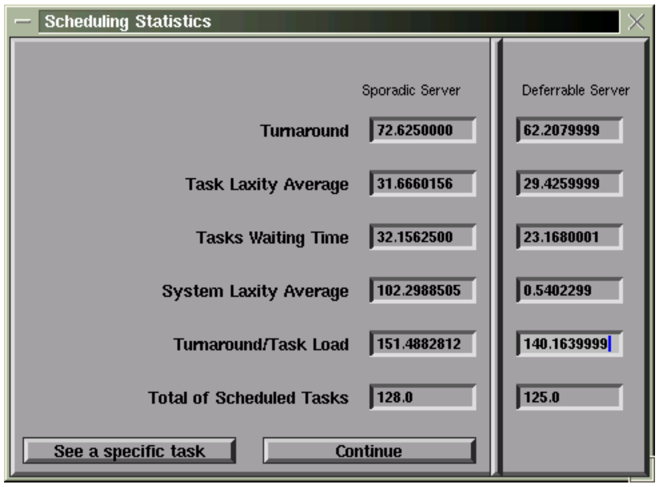
Teaching load sizing, balancing and stability

Some of the work with load analysis was already described in the previous section. At this point, the instructor would assign problems that are more complex than those solved while learning scheduling algorithms. These new assignments should include the simulation of small variants of each set of tasks in order to verify the behavior of a given algorithm un-



		Sporadic Server	Deferrable Server
Absolute delay	Mean value	40.0	20.0
	Highest value	0000040	0000020
	Standart Deviation	0.0	0.0
Deadline relative delay (%)	Mean value	20.0	20.0
	Highest value	20.0	20.0
	Standart Deviation	0.0	0.0
Load relative delay (%)	Mean value	100.0	100.0
	Highest value	100.0	100.0
	Standart Deviation	0.0	0.0
Tasks with deadline loss		0000001	0000001
Total of scheduled tasks		0000128	0000125

Fig. 2. Output table for deadline statistics.



	Sporadic Server	Deferrable Server
Turnaround	72.6250000	62.2079999
Task Laxity Average	31.6660156	29.4259999
Tasks Waiting Time	32.1562500	23.1680001
System Laxity Average	102.2988505	0.5402299
Turnaround/Task Load	151.4882812	140.1639999
Total of Scheduled Tasks	128.0	125.0

Fig. 3. Output table for performance statistics

der different restrictions. The assignments may assume the following forms:

- Simulation of a set of specific tasks, changing their loads until the rate of lost deadlines drops below a certain threshold, which characterizes a sizing limitation simulation;
- Simulation of a set of specific tasks, changing their loads until the rate of lost deadlines surpasses (or the systems laxity becomes smaller than) a certain threshold, characterizing a stability study;
- Simulation of different starting conditions for the multiprocessor algorithms that rely on task migration, in order to verify the efficiency of load balancing.

As one can see, in all of these assignments the user has to collect the same data he/she collects while simulating simple runs of a scheduling algorithm. The difference is that, here, the user has to pay attention to what is changing (or has to change) in the set of tasks in order to achieve the expected conclusion.

These experiments provide enough data to discuss the algorithms and the impact of working with mean or worst values for task loads. It is instructor's duty to arrange the assignments in such way that the student can clearly understand the implications of a good (or bad) choice of scheduling policy for a given situation. This arrangement is easy to achieve since it is also obvious that to ensure continuity the student has to learn the policy before he/she learns about the impact of variations on its application.

Achieved results and conclusions

At this time, we firmly believe that the use of RTsim helps the students to better understand the differences between scheduling policies. Unfortunately, we cannot compare the efficiency of such use with the learning of this topic without the tool, since the course was not taught even a single time in such way, and the classes are too small to provide experimentation with two groups.

The number of students enrolled in the classes that used this tool was 16 in 1998, 21 in 1999 and 18 in 2000, which are reasonable enrolments for a course that is not mandatory for every student. As we already stated, each class used a different version of RTsim. The results, however, are quite similar every year, except for the range of different experiments that could be performed.

In its first trial we could only assign experiments with single processor policies. That prototype did not provide many statistics to the user, which led to experiments aiming mostly the identification of differences between similar algorithms. From that class we could identify most of the problems and flaws that are now fixed or under implementation. It was also from that experience that we decided that RTsim could be turned to an educational tool.

At that time we also examined, and dismissed, the Hartstone Benchmark experiment at Rostock University [1], in Germany, which uses a commercial benchmarking environment (the Hartstone) to perform the same kind of simulations we do. Although this tool provided much more information than RTsim was capable to do by that time, we preferred its dismissal due to three conditions: it was not purely oriented to real-time applications, it did not provide some features that really characterize a learning environment, and it was not easily available for modifications.

After the 1998 class, RTsim went through several modifications, most of them originated by criticisms from students. For the 1999 class the simulator already included few improvements but could provide mostly the same informations found for the previous class. From this application we decided that the student should be able to test new algorithms, besides those that were already implemented and those (the multiprocessor algorithms) that were under implementation. This functionality was inserted during the year 2000.

For the 2000 class we had the multiprocessor policies added and a better set of statistics. The kind of assignments for this class was not much different from the previous ones, that is, the instructor could ask just for the schedule of several sets of tasks, including tasks for multiprocessor systems now. The instructor was also able to ask for more detailed comparisons between algorithms and analysis of load influence. Actually, this was the first time where all the different assignments listed in the previous section could be performed without an extra effort from the students.

It was also with this group of students that we noticed that the ability of comparing a schedule made by the software against another made by a student could be an excellent functionality to be added. We are now working on such functionality, which should be ready for the 2002 class.

Some other functionalities are also under implementation or study. They are:

- Built of Java interfaces to replace the X11-based interfaces that are in use;
- Built of a Java version that enables its use on a heterogeneous network, mainly into virtual labs and in the world wide web;
- Inclusion of the ability to mold itself to different learning styles. At this point we are working on the Kolb model [4];
- Inclusion of other topics of the real-time systems course, such as the simulation of timed petri net models.

From all experiments made, and students returns, we can conclude that the use of RTsim has greatly improved the learning of scheduling policies for real-time systems. The students have not only learned how the schedules are performed but also why they are performed that way, and what are the implications of such schedule. Therefore, they have a better understanding that every policy is not just a different rule to arrange the cpu occupation but also that each one have distinct effects on what tasks could be delayed and what will not.

This kind of understanding is exactly what we expected from the students. This understanding matches the course goals, that are the modelling of real-time tasks and their interactions. Therefore, we may assure that the introduction of such software in this course is already a success and could be even more successful with the addition of all functionalities under development.

Acknowledgements

We want to acknowledge the National Research Council - CNPq - and the Research Support Foundation of State of São Paulo - FAPESP - which have conceded grants for this project. We also want to give a special thanks to Luciana Pavani and Rodrigo Kehdy, who worked in previous versions of RTsim. We also want to acknowledge the strong contributions given by all students enrolled in the three classes where RTsim was used. Their part on this work is essential since they provided all results and directions for its continuity.

References

[1] Golatowsky, F., Timmermann, D., "Using Hartstone uniprocessor benchmark in a real-time systems course", in *Proc. of 3rd IEEE Real-Time Systems Education Workshop*, p 77-84, Poland, 1998.

[2] Hou, C., Shin, K.G., "Allocation of periodic task modules with precedence and deadline constraints in distributed real-time systems", *IEEE Trans. on Computers*, v. 46, n. 12, p 1338-1355, 1997.

[3] Kehdy, R.B., "Simulation of hard real-time scheduling algorithms", *DCCE Internal Report Inf-01/99*, (in portuguese), São José do Rio Preto, 1999.

[4] Kolb, D.A., "The learning style inventory: Technical manual", McBer, Boston, MA, 1976.

[5] Liu, C.L., Layland, J.W., "Scheduling algorithms for multiprogramming in a hard real-time environment", *Journal of ACM*, v. 20, n. 1, p 46-61, 1973.

[6] Manimaram, G., Murthy, C.S.R., "An Efficient dynamic scheduling algorithm for multiprocessor real-time systems", *IEEE Trans. on Parallel and Distributed Systems*, v. 9, n. 3, p 312-319, 1998.

[7] Nabuco, V.A., "Knowledge acquisition of scheduling rules for a scheduler simulator", *DCCE Internal Report Inf-03/00*, (in portuguese), São José do Rio Preto, 2000.

[8] Paula Bueno, L.P., "A simulator for performance analysis of real-time scheduling algorithms", *Graduation Design Work*, (in portuguese), São José do Rio Preto, 1996.

[9] Ramamritham, K., Stankovic, J.A., Zhao, W., "Distributed scheduling of tasks with deadlines and resource requirements", *IEEE Trans. on Computers*, v. 38, n. 8, p 1110-1123, 1989.

[10] Ramamritham, K., Stankovic, J.A., Shian, P., "Efficient scheduling algorithms for real-time multiprocessor systems", *IEEE Trans. on Parallel and Distributed Systems*, v. 1, n. 2, p 184-194, 1990.

[11] Sha, L., Rajkumar, R., Lehoczky, J.P., "Priority inheritance protocols: an approach to real-time synchronization", *IEEE Trans. on Computers*, v. 39, n. 9, p 1175-1185, 1990.

[12] Sprunt, B., Sha, L., Lehoczky, J.P., "Aperiodic task scheduling for hard real-time systems", *Journal of Real-Time Systems*, v. 1, n. 1, p 27-60, 1989.