Scheduling and Fault-Tolerance with Free Open-Source Components for Real-Time Applications

M. Silly-Chetto[1] **and T. Garcia**[1]

Abstract: In this paper, we will describe a national project (work supported by the French research office) that aims at the improvement of embedded computing systems for applications with real-time constraints. The objective of this project is first to create a library of free software components for the design of real-time operating systems and second, to participate in the evolution of an opened community standard, Linux. The key objective is as well to demonstrate the applicability and the interoperability of these software components by simulation, by integration and finally by tests on a real application. The tests on a mobile robotic platform (an Automated Guided Vehicle) are performed to show the benefits in terms of both improved integration process and adequacy with strict requirements on safety and reliability of next generation applications.

The set of selectable schedulers we are providing permits to cope with a high variety of real time applications with Linux/Rtai [RTAI 02]. RTAI (Real-Time Application Interface) has been developed at the Department of Aerospace Engineering, Politecnico di Milano for about six years, and as RTLinux, it provides real-time functionalities to Linux. The original scheduler in Linux/Rtai simply uses static priorities and consequently does not fit to next generation applications where tasks dynamically arrive and may be subject to faults.

In reality, tasks are periodic, soft aperiodic (non periodic without deadline) or hard aperiodic (non periodic with a strict deadline). We have implemented, tested and integrated premptive and non-premptive dynamic schedulers based on Earliest Deadline First (ED). Each of them beeing is selectable from a library of modules and can schedule tasks with different timing constraints. Our next objective is to provide Linux/Rtai with a general server, namely the EDL (Earliest Deadline as Late as possible) server that uses an on-line acceptance test to prevent overload situations and is able to jointly treat periodic, soft aperiodic and hard aperiodic tasks.

In real-time environments, a fault-tolerance policy should be selected and implemented to recover from errors within a certain time limit. In the project, we consider Software redundancy and Time redundancy stategies. Software redundancy means that tasks can have different software versions, so that when a version of a task fails under certain inputs, another version can be used. With time redundancy, the task schedule has some slack in it, so that some tasks can be rerun , possibly with less precision, and still meet critical deadlines. We have integrated dynamic fault-tolerance techniques which combine the Deadline Mechanism based on a software redundancy and the Imprecise Computation method based on an iterative algorithmics., Unlike hardware fault-tolerance, software and timing fault-tolerance are new and emerging fields. In order to test, validate and verify the properties of the schedulers, a simulator has been implemented. This simulator is enough generic to accept additional schedulers. An interesting aspect of the simulator is its ability to capture input validation and logging commands from users. This language looks like the C syntax. It is currently extended to log theoretical schedulers performances and check scheduling scenarios.

[1] IRIN / University of Nantes Rue Christian Pauc 44306 Nantes cdex 03 France
maryline.silly@iut-nantes.univ-nantes.fr