ARTIST 2

Network of Excellence

IST-004527 ARTIST2:
Embedded Systems Design

Activity Progress Report for Year 2

JPIA-Platform

A Common Infrastructure for Adaptive Real-time Systems

Clusters:

Adaptive Real-Time

Activity Leader:

Giorgio Buttazzo (Scuola Superiore S. Anna)
http://feanor.sssup.it/~giorgio/

The objective of this activity is to show how current operating systems and network protocols have to be extended to support emerging real-time applications that exhibit a high degree of complexity and operate in dynamic environments.

The impact on operating system standards (like RT-POSIX and OSEK) as well as network protocols will also be taken into account.

Table of Contents

1. Overview of the Activity

1.1 ARTIST2 Participants: Expertise and Roles

Cluster Leader: Giorgio Buttazzo – Scuola Superiore S. Anna (Italy)
Activity coordinator, kernel maintenance, development of robotic applications.

Team Leader: Luis Almeida – University of Aveiro (Portugal)
networking platform, development of distributed applications.

Team Leader: Gerhard Fohler – Technical University of Kaiserslauten (Germany)
video streaming applications, scheduling.

Team Leader: Michael Gonzalez Harbour – University of Cantabria (Spain)
definition of the POSIX operating system interface.

Team Leader: Alan Burns – University of York (UK)
feasibility analysis of fixed priority real-time systems.

Team Leader: Eduardo Tovar – Polytechnic Institute of Porto (Portugal)
distributed applications and QoS over heterogeneous networks.

1.2 Affiliated Participants: Expertise and Roles

Team Leader: Ivo De Lotto – University of Pavia (Italy)
kernel maintenance.

Team Leader: Paolo Gai – Evidence s.r.l. (Italy)
real-time kernels and operating systems standards.

Team Leader: Lucia Lo Bello – University of Catania (Italy)
Distributed real-time applications.

Team Leader: Pau Marti – Universitat Politècnica de Catalunya (Italy)
control applications.

1.3 Starting Date, and Expected Ending Date

Starting date: September 1st, 2004

Expected ending date. September 2007, when all the partners become familiar with the Shark kernel and have identified the key kernel features for each real-time application domain.

1.4 Baseline

At the beginning of the activity, most of the partners were not familiar with the Shark operating system and could not exploit the potentiality of the kernel in developing software applications using advanced real-time mechanisms, such as dynamic scheduling, proirity inheritance, resource reservation, aperiodic servers, and overload management algorithms. Moreover, they could not practically evaluate the impact of a specific kernel mechanism on the performance of a dynamic real-time application, simply because the existing kernels usually provide fixed policies that cannot be easily replaced or modified by the user.

However, each partner had a solid theoretical expertize in real-time systems and also some practical experience in one or more application domains. These conditions were exploited to evaluate the suitability of several kernel mechanisms to support the development of different types of real-time applications, including monitoring systems, robotics, control systems, and multimedia processing.

In particular, each team working in this activity gave a complementary contribution to define a common infrastructure for the development of advanced real-time systems:

- Scuola Superiore S. Anna, together with its affiliates University of Pavia and Evidence, has a strong expertize in kernel development, so it ensured the proper support to the other teams for maintaining the Shark kernel and teaching how to use it.
- University of Aveiro, Polytechnic Institute of Porto and University of Catania (affiliated to Pisa) have a strong expertize in real-time networks, which they used to evaluate the impact of kernel mechanisms on distributed real-time applications.
- Mälardalen University (now replaced by Technical University of Kaiserslauten) had already worked on video streaming and multimedia applications, therefore it committed itself to test the kernel on media processing.
- Universitat Politècnica de Catalunya, expert in control systems, gave the availability for testing the kernel on critical control applications.
- University of Cantabria and University of York, actively involved in standards for operating systems interfaces, gave support for implementing a POSIX interface.
- University of York, expert in feasibility analysis of fixed priority systems, gave support for implementing and analyzing new scheduling schemes.
- UP Madrid and its affiliate Univ. Carlos III of Madrid, expert in quality of service management, were the ideal candidates for evaluating the impact of adaptive kernel strategies in multimedia systems.

1.5 Problem Tackled in Year2

The objective of this activity in the second year was to deploy a working platform for experimenting RTOS mechanisms in control and distributed systems. Each partner selected a specific application with the aim of evaluating the impact of kernel policies on system performance. Sample real-time applications included mobile robotics, distributed systems, multimedia processing, and process control systems.

The specific kernel mechanisms that have been considered for evaluation include:

- **Periodic task scheduling algorithms**. Periodic task scheduling is extremely important since most software control systems are implemented through a set of periodic activities performing data sampling, sensory processing, control, action planning, and actuation. Although not strictly necessary, periodic execution simplifies the design of control algorithms and allows using standard control theory to guarantee system stability and performance requirements. When several of such activities execute concurrently in the same processor, however, each task may experience a variable delay and jitter, mainly due to the interference created by the other tasks. If not properly taken into account, delays and jitter may degrade the performance of the system and even jeopardize its stability. Fixed priority schemes and deadline-based algorithms (like EDF), both available into Shark, were evaluated in terms of overhead, complexity, jitter and robustness in handling overload conditions.
- **Aperiodic service mechanisms**. In addition to periodic tasks, most of real-time applications include a number of event-driven activities generated by asynchronous

interrupts coming from peripheral devices. If not properly handled, aperiodic activities may create transient overload conditions and introduce unbounded delays in periodic control tasks that could degrade system performance in an unpredictable fashion. Several aperiodic service policies, available into Shark, have been tested and evaluated, including Polling Server, Sporadic Server, Deferrable Server, Total Bandwidth Server and Constant Bandwidth Server.

- **Resource access protocols**. When tasks interact through shared memory buffers, additional blocking times can be introduced in task execution due to the access of mutually exclusive resources. Several solutions have been proposed in the literature to bound the maximum delay caused by resource conflicts (e.g. Non-Preemptive Protocol, Higher Locker Priority, Priority Inheritance Protocol, Priority Ceiling Protocol, Stack Resource Policy, Immediate Priority Ceiling, etc.). All these protocols are available in Shark as a specific resource modules and can be used in combination with different scheduling algorithms. In addition, Shark also includes a non blocking communication mechanism (namely, the Cyclic Asynchronous Buffer), based on memory duplication, which represent an interesting alternative for the communication among periodic tasks with different periods. Testing the effectiveness of such inter-task communication mechanisms was one of the objective of this research activity.
- **Overload management techniques**. Dynamic real-time systems are subject to unpredictable load variations which may cause transient or permanent overload conditions. If the system is not designed to handle such a situations, the system performance may degrade in an unpredictable fashion. Transient overload conditions due to execution overruns can be efficiently handled by *Resource Reservations* techniques, which basically isolate the temporal behavior of a task (or subset of tasks) protecting the rest of the systems from potential overruns. On the other hand, *Elastic Scheduling* provides an effective solution to cope with permanent overload conditions. According to this method, task utilizations are treated as flexible springs that can be compressed (by enlarging periods) to reduce the load up to a desired value. Such novel techniques (not yet available in commercial operating systems), have been implemented into Shark as basic scheduling modules to be tested and evaluated in real-world control applications.

1.6 Comments From Previous Review

1.6.1 Reviewers' Comments

FROM REVIEW 1
This task is focused on the selection of a common real-time environment and to the porting of various components on this environment to share a common real-time platform for experimentation.
A major part is missing in the deliverable - the list of requirements that led to the selection of the SHARK kernel and other components, the potential list of other kernels/components considered, and the reasons for their rejection.
The work reported is substantial and is clearly a work of integration between several European teams around a common infrastructure. The picture presented on this topic during the presentation would strengthen the deliverable.
The spreading of excellence could be reinforced by considering the placement of the kernel and components that can be shared by non-ARTIST2 partners under open source in an external consortium such as ObjectWeb which would increase the momentum around this infrastructure.
The deliverable is rejected for the reasons given here. The deliverable must be updated and resubmitted before the deadline given in section 9.

FROM INTERIM REVIEW 1
This task is focused on the selection of a common real-time environment and to the porting of various components on this environment to share a common real-time platform for experimentation.
The requirements that led to the selection of the SHARK kernel are now clearly stated.
The work reported is substantial and is clearly a work of integration between several European teams around a common infrastructure and repository.
The deliverable is accepted as it is.

1.6.2 How These Have Been Addressed

The reasons for selecting the Shark kernel have been precisely explained.

2. Summary of Activity Progress

2.1 Previous Work

It focused in building a common kernel infrastructure for testing novel scheduling algorithms and resource management policies in different application environments. To do that, we had to work with a real-time kernel with a modular structure, in which the implementation of a mechanism were as much as possible independent on the implementation of the other parts. In additions, the kernel had to be open source to allow extentions, well documented, and availble for PC-based platforms, supporting a sufficient number of devices to allow partners to develop control applications. Finally, to compare the performance of different algorithms, we wanted a kernel able to measure the execution of application tasks. Operating systems like RT-Linux and RTAI were excluded due to their quite complex structure and the dependency of the scheduler with the other kernel mechanisms. The only kernel we found suitable for this project was Shark, for the reasons explained in Deliverable 2-2 JPIA-a-ART-Y1.

Once the kernel was selected, we developed a web site (http://feanor.sssup.it/retis-projects) to create a forum for the various Shark users, to exchange messages, search for questions, etc.

The main action was to organize a workshop to introduce the Shark kernel to all the partners of the ART cluster, enabling the participants to quickly use the kernel, develop simple real-time applications, and implement novel scheduling algorithms. The workshop was held in Pontedera, Pisa (Italy), at the Scuola Superiore Sant'Anna, from February 28 to March 4, 2005.

A collaboration was started with the University of Siena at Arezzo, to enable the users at the development of real-time control applications on robot devices that are available in their control laboratory. The Scuola Universitaria Professionalizzante della Svizzera Italiana (SUPSI) was also contacted for porting a code generator for Scilab/Scicos tools toward the Shark platform.

The kernel was maintained by writing and updating the existing documentation, removing bugs, and modifying the internal structure of the scheduler. The **task_endcycle()** system call was modified to make it sensitive to system termination.

In addition, a graduate thesis was started at the ReTiS Lab for porting the low-level layer of Shark (OsLib) to the L4 microkernel.

The University of Aveiro, in collaboration with the University of Pavia, started investigating a kernel methodology for supporting the hard real-time communication over a dedicated Ethernet network. Finally, new device drivers were implemented based on Linux technology.

2.2 Current Results

2.2.1 Technical Achievements / Outcomes / Difficulties encountered

A new kernel release: Shark 1.5.1
A new kernel release, Shark 1.5.1, was issued on July 25, 2006, introducing several new features with respect to the previous version of the kernel (Shark 1.5). The work has been carried out at the University of Pavia, in collaboration with the Scuola Superiore Sant'Anna and Evidence S.r.l. The new kernel includes the following additions:

- Event filters have been added to the Tracer to allow the user to select the events to be monitored at runtime. Moreover, the tracer output can now be saved on local disks, in addition to remote servers.

- The support for the USB has been introduced. The support for Host and Hub devices is fully working; USB mouse, keyboards, joystick and joypad are already fully supported, while PWC chipset-based webcams are working with negligible problems on some specific machines.
- The IntDrive interrupt server for Linux-imported drivers has been updated in order to correctly implement the ideas proposed in the related scientific paper. There is now a new interface for initialization, and some delicate internal mechanisms has been fixed.
- A support for Dynamic Voltage Scaling has been provided to allow S.Ha.R.K. to exploit the power management techniques available in the most recent processors. AMD PowerNow and Intel Centrino SpeedStep are currently fully supported. The feature has been implemented as a kernel module and it is fully compliant with the Linux CPUFreq driver.
- A S.Ha.R.K. Quick Guide has been released to simplify the work to the new users. It covers the basic topics for getting familiar with S.Ha.R.K., such as installation, basic application development, system configuration, and remote execution of S.Ha.R.K. applications to improve productivity.
- Other new documents include the manual for the new S.Ha.R.K. Tracer (which covers the initialization, usage, and log saving procedures) and the HTML and TXT versions of the kernel manual. In particular, the TXT version is useful while developing under DOS, where both PDF and HTML documents cannot be handled.
- Several bugs have been fixed. The most relevant concern:
  - the dependency of the tracer library from the network library;
  - a bandwidth leakage in the EDF scheduling module;
  - dependencies of ".h" files into makefiles;
  - a mistake in the group_activate call, which tried to start tasks rejected from the guarantee test;
  - a wrong behavior of the IntDrive in particular cases.

Difficulties encountered:

- The PWC chipset-based USB webcams must include proprietary undisclosed code that must be properly linked and handled while building the USB library.
- The DVS only support BIOS features. ACPI features are not yet handled. Since many of nowadays power-aware features are supported through ACPI only, some features are still missing in S.Ha.R.K. Moreover, several BIOSes often supply buggy implementations of power-aware features, making the S.Ha.R.K. support unstable on some machines.
- Testing the new features required the execution of the kernel on a variety of different machines.
- While improving the Tracer, and concurrently writing the documentation, it has been difficult to understand the already available features in terms of configuration flags during initialization, since no documentation was already available and was not possible to contact the old maintainer.

URL: http://shark.sssup.it/

URL: http://shark.sssup.it/repository/shark

A repository of real-time applications

A repository for all real-time applications developed using the Shark kernel was created to facilitate the users in the development of new real-time software. The repository includes a a folder of supported applications and a folder of all unsupported software. The supported applications are tested and maintained by the developers to be consistent with the current kernel version, while the unsupported folder includes all programs, demos, and advanced applications developed under Shark and made available by the maintenance team.

Difficulties encountered: Documentation and comments of unsupported applications are not always complete since they are left to the users.

URL: http://shark.sssup.it/repository/applications

URL: http://shark.sssup.it/repository/unsupported

A repository for scheduling modules and resource management

A repository of all kernel modules developed for the Shark operating system was created to facilitate the users in the development of new kernel mechanisms. The modules include scheduling modules (implementing periodic schedulers, aperiodic servers, or overload management policies) and resource modules (implementing concurrency control protocols for accessing shared resources). Each modules contains the C source code and required headers compliant with the Shark module specifications.

URL: http://shark.sssup.it/repository/modules

Shark at University of York

This work has been done by Alex Zerzelidis at the Computer Science Department of University of York. The Preemption Level Protocol (PLP) allows EDF to be used on top of priority queues and was introduced in

> Burns, A., Wellings, A.J., and Taft, T. S., "Supporting Deadlines and EDF Scheduling in Ada", Lecture Notes in Computer Science, Springer-Verlag, Volume 3063 / 2004.

Alex Zerzelidis extended the PLP to include multi-unit resources. To achieve that, two amendments were introduced to the original protocol:

Amendment 1 (applies to point 2 of Section 3) Resource ceiling preemption levels *are dynamic* and any resource *R* has a current ceiling $\lceil R \rceil$ defined as

$$\lceil R \rceil = \max(\{0\} \cup \{\pi(\tau) \mid v_R < \mu_R(\tau)\}) .$$

where *π(τ)* is the preemption level of task *τ*, v_R denotes the number of units of *R* that are currently available and μ_R is the maximum requirement of task *τ* for *R*. That is, the current ceiling of resource *R* is the maximum of zero and the preemption levels of all the tasks that may be blocked directly when there are v_R units of *R* available.

Amendment 2 (applies to point 4 of Section 3) When a task accesses a resource, the current ceiling level of the resource is recalculated and the task's active priority is raised to the new current ceiling level of the resource. The active priority of all other tasks already accessing the resource remains unchanged. When the task releases the resource the ceiling is recalculated.

The PLP and its multi-unit extension was implemented on the SHARK RTOS as a new scheduling module in the kernel, called MPLP (Multi-unit PLP), which is based on the EDF and POSIX modules. The module can be registered as usual in the function

```
TIME __kernel_register_levels__(void *arg)
```

if we include a call to `MPLP_register_level(int flags, int prioritylevels)`.

The module implements PLP on a set of priority levels (their number specified by the `prioritylevels` parameter) and allows tasks to share multi-unit resources. Tasks can be created at runtime and declare a list of resources to use.

Shark at the Technical University of Kaiserlautern
Shark has been used at the Technical University of Kaiserlautern (TUKL) as a platform for video processing applications and, in particular, for experimenting adaptive resource management policies for user quality control. The methods estimate resource demands for frame decoding and use Shark scheduling algorithms for guarantees.

At TUKL, the Shark kernel was also adopted for education to teach real-time scheduling in an undergraduate course. Two labs are being developed: in one lab, students are required to develop a scheduling algorithm, analyse it and test it on the Shark kernel; in the other lab, students have to implement a simple video processing algorithm running on Shark, learning implementation and overhead issues.

URL: http://rts.eit.uni-kl.de/research/adaptive-rts

Shark for control applcations
A contact has been established with Dr. Sjur Vestli, from Logobject AG – Switzerland (http://www.logobject.ch/), for a possible use of Shark in robotic applications, and in particular for the control of autonomous vehicles. The group is currently using a fixed priority kernel for Power PC platforms, but they are moving to PC architectures and are looking for a kernel supporting dynamic scheduling and advanced resource management techniques for an efficient exploitation of the onboard resources.

URL: http://www.logobject.ch/

Shark for education in a Master Course for IIT graduate students
Shark was used as a sample real-time kernel in a Master course on Real-Time Systems organized in Pisa by the Scuola Superiore Sant'Anna, from March 2006 to May 2006, for 20 Indian graduate students selected from the India Institute of Technology (IIT). Students worked in groups of two or three people to develop a number of real-time concurrent applications and make experience in using advanced scheduling techniques.

Shark for education at University of Catania
Shark was used as a sample real-time kernel in a graduate course on Real-Time Systems given by Lucia Lo Bello at the University of Catania, from March 2006 to June 2006. Students worked in groups of two or three people to develop a number of real-time concurrent applications and make experience in using advanced scheduling techniques. A description of the projects is available at

URL: http://www.diit.unict.it/users/llobello/retisnetlab/shark.htm

Shark at the University of Illinois – Urbana Champaign
A testbed was developed at the Real-Time Systems Lab of the Computer Science Department of University of Illinois at Urbana-Champaign to show the applicability of real-time OS Shark and real-time MAC protocol RI-EDF for distributed control applications, where sensors/actuators are connected through a wireless channel. A wireless distributed control system for an inverted pendulum was built as a testbed environment. In the application, camera sensors are used in complement with potentiometer sensors on the cart to balance the pendulum pole. Real-time sensory acquisition was performed on a PC using the S.Ha.R.K. operating system <http://shark.sssup.it/>, whereas Berkeley Mica2 motes running TinyOS <http://www.tinyos.net/> were used for wireless communication with the RI-EDF protocol. We were able to successfully control the pendulum and utilize the extra network bandwidth for other real-time communications on the same shared channel using RI-EDF protocol. More details can be found at

URL: http://pertsserver.cs.uiuc.edu/~mcaccamo/IPC/index.html

Shark for robot control at University of Dresden
A contact has been established during ECRTS 2006 with the robotic group at the University of Dresden, for a possible use of Shark in robot control applications. The group is currently using a simple priority-based kernel developed in their lab for Motorola 68020 architectures, and then adapted to run on Power PC platforms.

2.2.2 Publications Resulting from these Achievements

1. Mauro Marinoni and Giorgio Buttazzo, "Adaptive DVS Management through Elastic Scheduling", Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2005), Catania, Italy, September 2005.
2. Gianluca Franchino, Giorgio Buttazzo, and Tullio Facchinetti, "A Distributed Architecture for Mobile Robot Coordination", Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2005), Catania, Italy, September 2005.
3. Paulo Pedreiras, Paolo Gai, Luis Almeida, and Giorgio Buttazzo, "FTT-Ethernet: A Flexible Real-Time Communication Protocol that Supports Dynamic QoS Management on Ethernet-based Systems", IEEE Transactions on Industrial Informatics, Vol. 1, No.3, August 2005.
4. M. Caccamo, T. Baker, A. Burns, G. Buttazzo, and L. Sha, "Real-Time Scheduling for Embedded Systems", in Handbook of Networked and Embedded Control Systems, D. Hristu-Varsakelis and W. S. Levine Editors, Birkhauser, Boston, 2005.
5. Giorgio Buttazzo, "Real-Time Operating Systems: The Scheduling Aspects", in The Embedded Systems Handbook, Edited by Richard Zurawski, CRC Press, 2005.
6. Giorgio Buttazzo et al. "Adaptive Real-Time Systems for Quality of Service Management", in Embedded Systems Design, The ARTIST Roadmap for Research and Development, Lecture Notes in Computer Science, Vol. 3436, Edited by Bruno Bouyssounouse and Joseph Sifakis, Springer, 2005.
7. Tullio Facchinetti, Giorgio Buttazzo, Mauro Marinoni, and Giacomo Guidi, "Non-Preemptive Interrupt Scheduling for Safe Reuse of Legacy Drivers in Real-Time Systems", IEEE Proceedings of the 17th Euromicro Conference on Real-Time Systems, Palma de Mallorca, Spain, July 2005.
8. Enrico Bini, Giorgio Buttazzo, and Giuseppe Lipari, "Speed Modulation in Energy-Aware Real-Time Systems", IEEE Proceedings of the 17th Euromicro Conference on Real-Time Systems, Palma de Mallorca, Spain, July 2005.
9. Mauro Marinoni, Giorgio Buttazzo, Tullio Facchinetti, and Gianluca Franchino, "Kernel Support for Energy Management in Wireless Mobile Ad-Hoc Networks", Proc. of the Workshop on Operating Systems Platforms for Embedded Real-Time applications (OSPERT 2005), Palma de Mallorca, Spain, July 5, 2005.
10. Mauro Marinoni and Giorgio Buttazzo, "Balancing Energy vs. Performance in Processors with Discrete Voltage/Frequency Modes", Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Sydney, Australia, August 2006.
11. Hoai Hoang, Giorgio Buttazzo, Magnus Jonsson, and Stefan Karlsson, "Computing the Minimum EDF Feasible Deadline in Periodic Systems", Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Sydney, Australia, August 2006.
12. Giorgio Buttazzo and Paolo Gai, "Efficient Implementation Of An EDF Scheduler For Small Embedded Systems", Proceedings of the 2nd Workshop on Operating Systems Platforms for Embedded Real-Time applications (OSPERT 2006), Dresden, Germany, July 2006.
13. Giorgio Buttazzo, "Real-Time Operating System Support for Energy-Aware Computing", Automazione e Strumentazione, 14(1), pp. 88-95, January 2006.

2.2.3 Keynotes, Workshops, Tutorials

First European Laboratory on Real-Time and Control for Embedded Systems
RETIS Lab, Scuola Superiore Sant'Anna, Pisa, Italy, July 10-14, 2006.

A graduate course on real-time and control for embedded systems was held in Pisa (Italy), at the RETIS Lab of the Scuola Superiore Sant'Anna, on July 10-14, 2006. It was organized in collaboration with the Control cluster lead by Karl-Erik Arzen.

The purpose of the course was twofold. The first objective was to provide the most important concepts and methodologies used in developing real-time embedded systems, including fundamentals of real-time scheduling, operating systems, distributed systems, and control theory. The second and more challenging goal was to show how to apply theory into practice, teaching students how to develop simple real-time distributed control applications using the state-of-the-art technologies available into the Shark kernel.

URL: http://www.artist-embedded.org/FP6/ARTIST2Events/Events/RT-Control/

Outcomes
The course was very successful and actracted 30 participants from all over the world (10 different countries). Among them, some came from industry and research centers, indicating a great interest on such types of activities. Each participant had the possibility to closely interact with all the teachers, for the entire duration of the course, to ask specific questions and clarify obscure points. In additions, 5 lab assistants helped the participants to quickly get familiar with the Shark kernel and followed them during the software development process. Thanks to the joint cooperation of the involved ARTIST2 teams, 10 robot control devices and 6 distributed applications were prepared for the course, and each participant had the opportunity to design, develop, and test a control application of his choice, working alone or together another participant. All the projects developed in the course are available in the following web site:

URL: http://shark.sssup.it/artist2/course06/

Difficulties encountered
Setting up the control devices for the practical experiments took the major effort in the preparation of the course. Sensory acquisition interfaces and servomotor control boards had to be developed and tested for each robot device and the total cost (hardware + labor) required for this phase significantly overcome the allocated budget. If the course has to be repeated, as suggested by the enthusiastic response, it requires a more substancial economical support from ARTIST2.

Course: Improving your research skills: a mini workshop for graduate students
RETIS Lab, Scuola Superiore Sant'Anna, Pisa, Italy – March 14-17, 2006
Speaker: Lui Sha (University of Illinois at Urbana Champagne, USA)

This course was aimed at teaching students how to do research. In particular, it addressed the problems of how to learn, formulate and solve research problems, and how to communicate.
Research means re-search: searching again and again in the product space of problem formulations and solutions until potentially high impact technologies is found. The efficiency of any search depends greatly on the methods we use, no matter the search is for oil under the ground or for new knowledge. In this mini-workshop, Professor Lui Sha shared his experience on research and education with the students, who have been guided to discover their skills and better organize their research plans. Students have learnt how to organize teams to create and improve research plans.

URL: http://feanor.sssup.it/~giorgio/sha06

Keynote: Real-Time Issues in Mobile Wireless Networks
Speaker: Giorgio Buttazzo

Conference: 9th Int. Conference on Principles of Distributed Systems (OPODIS 2005)
Pisa, Italy – December 12-14, 2005

This talk presented some of the most challenging problems to be solved in order to support the development of mobile wireless networks of cooperating robots. Some of these problems include the real-time execution of acquisition and control processes, the efficient management of computational resources, the software control of energy consumption, the real-time communication protocols on wireless networks, and the development of distributed agreement algorithms for reaching a consensus in collective decisions.

URL: http://www.di.unipv.it/OPODIS2005/

Keynote: Towards Component-Based Operating Systems
Speaker: Giorgio Buttazzo
Conference: Workshop on Operating Systems Platforms for Embedded Real-Time applications (OSPERT 2006)
Dresden, Germany – July 4, 2006

This talk presented the characteristics and the advantages of having a component based operating system, also discussing the difficulties to be solved and possible solutions that can be adopted at the kernel level.

URL: http://www.cs.ucsc.edu/~sbrandt/OSPERT.html

Keynote: Real-Time, Distributed Control Systems: Performance, Resource Planning, Applications
Speaker: Josep Fuertes
RETIS Lab, Scuola Superiore Sant'Anna, Pisa, Italy – May 3, 2006

In real-time control systems, the interplay of real-time constraints and control specifications can be somewhat subtle. In order to successfully analyze the behavior of a distributed control system, the type and location of jitter and delay must be characterized, and their effect on the performance of the control system understood. This talk presented how timing affects control and real-time performance and how the group at the University of Catalonia is facing the analysis and design methods for distributed control systems that cannot guarantee equidistant sampling and actuation.

The talk gave an introduction of control theory centred in the time related properties, explaining the relation between continuous and discrete sampled control systems and clarifying the interaction between performance specifications of control and real-time systems.

3. Future Work and Evolution

3.1 Problem to be tackled over the next 18 months (Sept 2006 – Feb 2008)

In the next 18 months we are going to use Shark for understanding how to build a component-based operating system, where most of the available kernel features can be composed together to create several user-defined configurations.

In particular, a component based approach should separate mechanisms from policies in order to replace a scheduling algorithm or a resource management protocol without affecting the applications and the others components. In addition, it should allow combining different scheduling disciplines to support the development of hierarchical software architectures.

There are several benefits in adopting a component based approach at the operating system level. First of all, it would be possible to enhance the functionality of the kernel by adding new blocks, depending of the application requirements, so tailoring the kernel to the specific system to be developed. Secondly, it would facilitate and speed up the integration of novel research results, which could increase efficiency and/or predictability. Finally, it would simplify the process of porting the kernel on different platforms, so reducing the time to market and the development costs on upgrades (since only small parts should be developed).

However, there are several practical and theoretical problems to be solved, since most of the mechanisms implemented in a kernel (like scheduling, resource protocols, interrupt handling, aperiodic servers, synchronization and communication) heavily interact with each other and have a high degree of inter-dependencies.

We plan to treat such problems by addressing the following issues:

- decoupling scheduling algorithms from applications;
- decoupling scheduling mechanisms from scheduling policies;
- decoupling scheduling algorithms from resource management protocols;
- combining resource reservation with resource management protocols.

3.2 Current and Future Milestones

1. **(achieved)** Year1: Initial definition of the operating system and network features. *The SHARK operating system developed at the Scuola Superiore Sant'Anna of Pisa has been identified (for the reasons explained in* Deliverable 2-2 JPIA-a-ART-Y1*) as the most suited kernel for building a common infrastructure to perform advanced experiments on real-time systems.*
2. **(achieved)** Year2: Deploy a working platform for experimenting RTOS and network development. *The SHARK operating system was upgraded according to the partners' needs and deploied on each partner site. A specific workshop has been organized in Pontedera (Pisa) to teach partners how to use the kernel for writing a real-time application and how to write new scheduling and resource modules.*
3. **Year3: Participate in the evolution of RTOS and networking standards, by introducing advanced scheduling methods for enhancing the predictability of real-time systems and handle their increased complexity.**
4. **Year3: Identify the problems to be solved for developing a component-based real-time operating system.**

3.3 Indicators for Integration

The indicators for integration will be of different kinds, including:

- real-time control applications developed by the joint collaboration among partners, each focusing on a different aspect (e.g., control, scheduling, analysis, languages, etc.) related to his peculiar expertize;
- specific scheduling and resource modules developed by the partners and integrated in the kernel;
- new device drivers developed for the Shark kernel by the cluster members;
- joint publications of research papers;
- joint events (e.g., workshops, summer schools, graduate courses, etc.) organized by the cluster members to spread real-time technologies.

3.4 Main Funding

The basic research on real-time operating systems and advanced scheduling techniques come from the following National and European projects:

National project PRIN-MIUR 2004: “Real-Time Operating Systems for Supporting Cooperating Autonomous Robots”. Project No. 2004095094_003.

European Strep project: “FRESCOR - Framework for Real-Time Embedded Systems based on Contracts”.