Project-Team aces

Ambient Computing and Embedded Systems

Rennes

THEME 1B

Activity Report

2003

Table of contents

1. Team

Head of project-team
Michel Banâtre [Research Director]

Administrative assistant
Evelyne Livache

INRIA project staff
Gilbert Cabillic [Research Scientist, up to 30/09/2003]
Paul Couderc [Research Scientist, since 01/10/2003]

CNRS project staff
Jean-Paul Routeau [Senior technical staff]

University project staff
Frédéric Weis [Assistant Professor, IUT de Saint-Malo]

INSA project staff
Isabelle Puaut [Assistant Professor, HdR]
David Decotigny [Teaching Assistant, up to 09/01/2003]

Project technical staff
Mathieu Becus
Paul Couderc [up to 31/03/2003]
Jean-Philippe Lesot [up to 30/09/2003]
Salam Majoul [up to 30/09/2003]
Frédéric Parain [up to 30/09/2003]
Cyril Ray [since 01/10/2003]
Grégory Watts

Ph. D. student
Alexis Arnaud [Inria grant]
Carole Bonan [Inria grant]
Julien Pauty [MESR grant]
Pushpendra Singh [Inria grant]
David Touzet [Inria grant, up to 30/09/2003]
Arnaud Troël [Inria grant, up to 30/09/2003]
Claude Vittoria [Inria grant, since 01/10/2003]

Other Personnel
Mathieu Avila [Junior technical staff]

2. Overall Objectives

Research in operating systems is in constant move due to the increase of wireless architectures with limited resources such as memory, battery or processor performance, and to emerging applications in the area of ambient computing. ACES research directions are defined in this context. We mainly address two topics: embedded systems architectures and systems support for ambient computing.

- **Embedded architectures and operating systems.** Here, we are concerned with the characterization of the resource used in embedded applications in terms of computing time and energy consumption. We also address the design of operating systems taking into account hardware (limited resources) and software (hard and soft real time) constraints.
- **Systems support for ambient computing.** We are interested in the design of spatial information systems. This concept consists of building and controlling implicitly computing systems in relation

to the properties of the physical world, like the relative position of physical objects and their movements. The implementation of such systems requires efficient distributed mechanisms to spread contextual information, as well as appropriate programming models to handle the physical context of the applications.

One major objective of the ACES project is the valorization of our research results. We have recently succeeded in transfering a part of our results with Texas Instrument, which has created a "Java competence" center in Rennes. This creation is the result of a very close and beneficial collaboration which we have led for five years (see section 9.9 for more details).

3. Scientific Foundations

3.1. Introduction

Key words: *Embedded systems*, *hard/soft real-time*, *energy consumption*, *design tools*, *ubiquitous computing*, *ambient computing*, *spatial navigation*.

The following paragraphs give a quick overview of the scientific background of the ACES research activities.

3.2. Embedded systems

The term *embedded system* denotes an autonomous system which has all the hardware elements it needs to operate (processor/microcontroller, ROM/RAM, peripheral devices). Embedded systems in general highly interact with their environment. The design of embedded systems and applications is subject to a number of constraints. Programs on an embedded system often must run with *real-time constraints* due to the interactive nature of embedded systems (see paragraph 3.2.1). For cost considerations, the available resources in an embedded system are limited. Concerned resources are the available *energy* (see paragraph 3.2.2), available memory (see paragraph 3.2.3) or communication bandwidth. Finally, embedded systems may have dependability requirements (see paragraph 3.2.4).

3.2.1. Real-time constraints

Many embedded systems highly interact with the outside world through sensors, actuators or wireless communications. Such interactions result in more or less stringent timing requirements depending on the target application. Systems are termed *real-time* [43] when their correctness does not only depend on the produced results (functional correctness) but also on the time when they are produced (temporal correctness). Real-time constraints can further be classified into *hard* and *soft* constraints depending on the gravity of the consequences of a violation of a timing constraint.

3.2.1.1. Hard real-time systems.

In hard real-time systems, missing a timing constraint is considered to be a fatal fault, which may have disastrous consequences (loss of human lives, economical or ecological disasters). As examples, a late command to stop a train may cause a collision, and a bomb dropped too late may hit a civilian population instead of the intended military target. Hard real-time systems require a validation that the system always meets the timing constraints. By validation, we mean a demonstration by a provably correct procedure (schedulability analysis) or by extensive simulation and testing.

An impressive volume of research results has been produced in the last twenty years in the field of real-time scheduling and real-time schedulability analysis [42][29]. The ordering of jobs (scheduling policy) can be determined off-line or on-line, then using fixed or dynamic priorities for jobs. An example of schedulability policies is the rate-monotonic policy (RM), for which periodic jobs are scheduled according to static priorities, with the priority of a job inversely proportional to its period. Another one is the earliest deadline first policy (EDF), for which jobs are scheduled according to dynamic priorities, the most important priority being assigned to the job with the earliest deadline.

Schedulability conditions, for a given scheduling policy and a given model of the system workload, provide a verdict of the system schedulability (i.e. indicates if all timing constraints are met or not). Existing schedulability conditions mainly differ by the model of the system workload they assume (e.g. periodic, sporadic or aperiodic tasks), by the metric they aim at optimizing (e.g. maximum job lateness) and by the type of information generated (e.g. static schedule, job priorities).

Most schedulability conditions assume that the worst-case execution times (WCETs) of tasks are known. Whereas schedulability analysis is a traditional field of research in the real-time area, the design of methods for automatically obtaining the WCETs of programs is a more recent research domain which is currently very active [40]. The goal of such methods is to produce estimates of the WCET of tasks (considered in isolation) on their target hardware. To be valid for use in hard real-time systems, WCET estimates must be *safe*, i.e. guaranteed not to underestimate the real WCET. To be useful, they must also be *tight*, i.e. provide acceptable overestimations of the WCET. Indeed, largely overestimated WCET estimates result in an under-utilization of the processor at run-time.

The traditional way to determine the execution time of a program is by measurements, known as *dynamic* execution time analysis. Unfortunately, unless all the program input data can be enumerated, these methods are not guaranteed to produce safe upper bounds of the WCET. In contrast, *static WCET analysis* methods avoid the need to run the program by simultaneously considering all possible inputs, possible program flows, and how the program interacts with the hardware. They operate on models of programs: syntactic-tree [30] obtained from the program source code, or control-flow graph [41], obtained from the program binary code, to identify the possible execution paths of a program. A model of the target hardware is also required: presence of caches [35] or pipelines [48]. The actual computation of the WCET estimates is achieved thanks to the program and hardware models, for instance through recursive calculations based on the syntactic tree, or through the generation of Integer Linear Programming (ILP) problems.

3.2.1.2. Soft real-time systems.

In soft real-time systems, the meeting of timing constraints is desirable, but occasionally missing a timing constraint has no permanent negative effects. Examples of soft real-time applications are multimedia, voice over IP and video systems. For instance, in a video playback system it is not fatal to miss an occasional frame, and this is often not even detectable by the user. In general, for soft real-time systems, the failure to meet timing constraints means that the quality of service provided is reduced, but the system will still provide useful service.

Job scheduling in soft-real time systems is important. Indeed, its aim is that all timing constraints are met. However, in contrast to hard real-time systems, missing a deadline has much less severe consequences. Thus scheduling policies and validation methods are slightly different from the ones used in hard real-time systems. They do not need to consider the *worst-case* operating conditions of the system (worst-case workload, worst-case execution times). Instead, it is sufficient to have a statistical knowledge of the system timing features (workload, task execution times).

3.2.1.3. Processing power.

Besides real-time constraints, there is an increasing demand for processing power in embedded systems, due to the augmenting complexity of embedded software. This issue can be dealt with by means of software-implemented optimization methods (for instance, optimization of frequently executed code sequences) or by using specific hardware support.

3.2.2. Energy consumption

Autonomy is a common problem for a large majority of embedded systems (cellular phones, small appliances, etc). To extend the autonomy of a system, two complementary approaches exist: increasing the energy capacity of batteries or integrating an energy consumption management inside the embedded operating system, with the objective to decrease the overall energy consumption. In the context of that last approach, a set of methods (either hardware or software) has been designed like low-power hardware components or software optimizations to decrease the energy consumption. In this section we focus on software techniques.

When building the binary of an application, one software optimization consists of choosing processor instructions that consume less energy than others, in order to decrease the execution cost in term of energy of a piece of code [44].

To optimize wireless software network protocol applications, the minimization of the number of sent and received messages can be done to decrease the energy consumption of the wireless network use.

Another optimization consists in executing a treatment on a remote server if the cost of the communications introduced to execute that treatment is less than the local execution of that treatment on the appliance [31].

The design of a global software strategy that uses hardware mechanisms to decrease the global energy consumption can be done through the scheduler. First, some processors provide a way to manage several execution modes that consumed differently. So, entering a mode that consumes less enables to save energy. For example, during the inactivy periods of the system, the scheduler can activate a sleep mode [33]. Second, processors provide ways to control either the voltage or the frequency of running [46]. Thanks to the fact that the voltage of a processor influences significantly the energy consumption, the scheduler has the objective to set the most adequate voltage dynamically. Of course, time constraints of applications need to be managed and extra latencies introduced on the execution of treatments need to be supported and analyzed by the scheduler [37].

3.2.3. Memory consumption

Memory is a scarce resource in many embedded systems (from a couple of KB in smart cards to several MB in a cellular phone). As a consequence, memory management is a key issue in embedded systems. Several methods have been designed to address the limited memory of embedded systems. Examples of such methods are the use of compact interpreted bytecode (for instance Java bytecode), code compression techniques [25], and dynamic memory allocation techniques aiming at limiting memory fragmentation [32].

An additional issue to be addressed in systems with real-time requirements is that memory management is compatible with the real-time constraints of applications. Hard real-time systems are mostly based on static memory management. However, several studies aim at using elaborated memory management techniques in real-time systems: dynamic memory management and garbage collection [39] and address translation [27].

3.2.4. Fault tolerance

Some embedded systems have to operate despite the presence of faults, should they come from the hardware or from programming errors. Fault tolerance mechanisms have been the subject of many researches in the last thirty years [26]. For instance, to tolerate hardware faults, it is necessary to (i) detect errors, through the use of error detection codes or on-line diagnosis; (ii) to recover from the error, using either backward error recovery or task duplication. Additional constraints in embedded systems come from the limited resources (memory, processing power) and in some cases in the real-time constraints of applications. In particular, in hard real-time systems, the fault tolerance mechanisms (e.g. task duplication) have to be accounted for during the validation of the temporal behavior of the system.

3.3. Ambient Computing

The foundations of Ubiquitous Computing (also named Ambient Computing) have been defined by Marc Weiser in his paper titled "Some Computer Science Issues in Ubiquitous Computing" [47]. The goal of ambient computing is to provide services to users according to their current situation, and interactions have to be as implicit as possible. In his paper, Weiser noticed that using a computer today requires all the user's vigilance. The current mobile systems, used by mobile users, cannot address this type of constraints. In Ubiquitous Computing mobile systems have to be used more intuitively without explicit interactions. The design of system environments able to support such ubiquitous applications requires to address two main topics: the characterization of the different contexts in which users evolve (see paragraph 3.3.1), and the way information can be accessed by users in a mobile environment (see paragraph 3.3.2).

3.3.1. Definition and use of the context

3.3.1.1. Introduction.

The goal of ubiquitous computing is to link transparently digital environments to the physical world. This integration between both worlds (physical and digital) has to provide users with implicit and automatic interactions with their surrounding environment. In a near future, these interactions have to occur without the user being aware to use the neighbor processors.

In order to be linked to the physical world, a processor (for example an embedded device) has to be equipped with tools dedicated to the perception of its environment. Morever this perception becomes useful only if all caught information can be recognized and understood by the application. For human beings, the detection and the analysis of a given situation are two tasks that are easily carried out throughout a day. When entering in a room, we are able to know how many people are there, their identities ... The execution of these tasks becomes difficult for a processor, because it is not able to identify the pertinent information sources. So in ubiquitous computing applications delegate the tasks for catching the execution environment (also named **context**) to dedicated systems such as GPS (for localization information).

Three types of context are generally used by ubiquitous computing:

- The digital context, which includes parameters like the presence (or the absence) of a network connectivity, the available bandwidth, the connected peripherals (printer, screen) ...
- The user context, which groups all user's information: identity, preferences, localization ...
- The physical context, related to the user's environment (climatic condition, noise level, luminosity), and also date and time. This last type of context, combined with the two previous ones, can be used to provide context history.

In order to illustrate this definition of the context, let us consider the example of a virtual guide for a museum. Each visitor carries an embedded processor (for example a PDA). He can read on the screen information about the surrounding artworks. In this application, the **pertinent context** is made up of the set of artworks situated near the user.

The general architecture of a system for ubiquitous computing is based on three main components:

- The real environment, which corresponds to the entities of the physical worlds (objects, people, places ...).
- The virtual environment, made up of information systems, like the Web for example.
- The interfaces which perform the link between the two environments.

The application is able to evaluate the state of the "real world" thanks to sensing techniques. It can be the position of a person (caught with a localization system like GPS), weather information (achieved with specialized sensors) ... So for sensing the environment, ubiquitous applications have to use all techniques which allow to update automatically digital information concerning events or entities of the physical world. Conversely, interfaces can be used in order to act in the physical world through the digital environment. For example, the windows of a car can be automatically closed when it is raining.

In an ubiquitous environment, a user has an implicit access to the information space by means of its actions in the real world. It implies that the user has at one's disposal a computer, capable of enriching the environment with pertinent information and making easier the interactions with the real world. So ubiquitous computing requires the use of embedded computers with (i) user interface suited to implicit interactions and (ii) wireless communication capabilities. Indeed, the physical environment can contain several objects linked with digital information (for example, in a library, a set of books equipped with an electronic tag). It is not conceivable to use a wired network to have access to these objects (which can be mobile).

3.3.1.2. Spatial Information System.

We call Spatial Information System an information system whose data are related to the physical space. Many ubiquitous applications are spatial information systems. In such applications, information can be obtained automatically through interactions between entities from the real world. Application's execution flow is driven by spatial conditions ; in the program we find statements such as "if the user is in front of a painting then display the artist's bibliography" or "if the user enters his office then play a welcome message".

The principle is to use existing properties in the real world (such as location or physical proximity) and to extract automatically an information system from these properties. The main spatial property used by a context-sensitive system is the user's position in the physical space. This parameter is an essential piece of the global context attached to a given user. Indeed physical space is well organized: a room, a company, a library, a city can be seen as spatial organizations. For example in a library, elements are arranged in the space according to topics.

Another contextual parameter is the identity of near entities, which is dependent on the user's position in the physical space. Indeed the value of this parameter is directly linked with the physical proximity of these entities. According to the environment, other spatial parameters may be used by context-sensitive systems to enrich the context: speed, speedup, direction ... They are caught with dedicated sensors, or deduced from successive values of the user's position.

We can identify two approaches for spatial information system implementation: the logical one and the physical one.

3.3.1.3. Logical approach.

The logical approach relies on a representation of the physical world, called a digital model. It is managed by a machine that we call services platform. The digital model is used to deliver services to users according to their context. If we take the example of the virtual museum guide, this model can be a database where the information is associated with physical positions. The user's coordinates are sent to the database to get data according to the user's position.

Scalability and complexity are the main drawbacks of this approach. Firstly, the model must be up to date, so the more dynamic a system is, the more numerous model updates are needed. The services platform is a potential bottleneck if it must deliver services to all users, and if the model updates are numerous. Modeling the physical environment is always possible, nevertheless some models lead to extensive computations, like image processing.

3.3.1.4. Physical approach.

The physical approach does not rely on a digital model of the physical world. The service delivery is computed where the user is. This is done by spreading data and wireless computing devices directly in the physical environment, on the physical objects implied in the application. Each device manages and stores the data of the corresponding object. In this way, data are physically linked to objects. For example, with this approach, there is no need to update a positions database when physical objects move since the data *physically* moves with them.

With the physical approach, computations are done by the embedded devices. The devices interact when they are connected. The interactions' goal is to deliver the service to the user. The user context is represented by the set of objects connected to his device. If we come back to the museum example, the data is directly embedded on the paintings. When the visitor's guide is connected to the paintings' device, it receives the information and display it.

3.3.2. Information access in mobile environments

When developing applications in a mobile environment, two main approaches for accessing data to/from mobile entities can be envisioned: (1) to provide means to applications in order to adapt to the effects of mobility, and (2) to hide to applications the effects of the mobility. We briefly describe these two approaches.

3.3.2.1. Adaptation to mobility.

Wireless environments in which mobile computers operate can be very turbulent. First, the local resources available can be limited, and depend on the entity features. Second, the network infrastructure supporting mobile systems is highly variable. This variability suggests that the application must adapt to the local resources and/or network related changes. In other words, data accessed by the user must be adapted. One solution for adapting data is to perform the transformation of data from one representation to another. For example, depending on the actual behavior of the system (network bandwidth, load...), it implies to have the ability to manipulate different representations of the same data that are physically different but semantically equivalent (e.g., the color and black-and-white versions of a same picture).

In the paper describing the Odyssey system [36], three options for implementing adaptation mechanisms are proposed:

- The "Application-Transparent Adaptation": data are adapted only by the system and the network, they automatically handle changes in connectivity between hosts. In that case, applications have no capability to decide how to use available bandwidth, how to manage data format when end-terminal features are modified...
- The "Direct Application Adaptation": in this approach, an application is directly responsible for coping with the consequences of terminal and network changes. This approach implies that an application uses monitors that keep the track of the context of the system, by observing, among other things, the behavior of the network, the load, the CPU usage...
- The "Application-Aware Adaptation": the adaptation is made though a collaborative effort between network/system and applications. Such a solution implies that the network/system has to collaborate with the mobile entity: it has to know the location of the entity, the main features of the attached terminal (size of the screen, capacity of the CPU...). Of course the mobile entity has also the capability to interact with network components: for example, when a given application is executed, the network can inform the mobile terminal when significant changes in the availability of the needed resources occur. This solution has been retained in the Odyssey system.

3.3.2.2. Mobility hiding.

An unstable connection can isolate temporarily a mobile entity from the rest of the system. In such a situation the proposed service cannot be properly delivered because it is impossible to have access to new remote information. In order to hide disconnections to a user, prefetching techniques can be used: documents are preloaded in a local cache of the mobile terminal during high connectivity periods. So the application remains operational even when a temporal disconnection occurs (because all needed documents have been stored locally in the cache of the terminal). Data to be loaded in the cache are selected according to a combination of information provided by the user, and information automatically tracked (for example a log of all previously accessed documents).

4. Application Domains

4.1. Introduction

Key words: *Embedded systems*, *mobile applications*, *wireless appliances*, *Personal Digital Assistant*, *Spontaneous Information Systems*, *Spatial Information Systems*.

The ACES research activities have different fields in which the produced software is embedded (see paragraph 5.1). Considering the growing use of embedded systems, we do not try to describe hereafter all application domains to which our research applies. Let us quote as examples the banking applications (smart cards), telecommunications (mobile communications in particular), cars, avionics, energy production, pocket computers, SoC (Systems on Chip). In the following paragraphs we only give an overview of the domains to which our research tasks are currently applied within the framework of industrial collaborations.

4.2. Embedded multimedia applications for wireless appliances

Key words: *Java runtimes*, *embedded systems*, *soft real-time scheduling*, *multimedia applications*.

Progress in hardware technology enables to envision the use of wireless appliances for the execution of applications traditionally supported on the desktop. In that context, the user will be provided with an open environment that allows him to perform actions as diverse as Internet accesses, application downloading, and phone calls. However, the actual provision of the aforementioned environment on the wireless appliance is far from being straightforward; hardware and software solutions still need to be devised. About the hardware, Texas Instruments is going to propose a new hardware solution based on heterogeneous multiprocessor integration. About the software, a wireless software environment must meet the following requirements:

(1) Maximize efficiency so as to support efficient execution of applications, possibly run concurrently. (2) Minimize power consumption so as to provide users with highly autonomous, small-sized appliances. (3) Maximize availability so that users can benefit from their appliances despite the occurrence of failures. (4) Support multimedia applications, which have soft real-time constraints. (5) Support dynamic downloading of applications (e.g. through Internet).

A Java environment appears to be an ideal candidate to achieve requirement 5. Furthermore, the de facto standard status of Java leads to have a significant amount of available software, which will continue to grow. However, providing a Java environment for Texas Instruments appliances that both supports multimedia applications and exploits the underlying hardware potential (i.e. an environment that meets the above 5 requirements) is an open issue and constitutes the core of the collaboration with Texas Instruments (see sections 6.2.2 and 7.1.1).

4.3. Information systems for wireless appliances

Key words: *wireless communications*, *physical proximity*, *communication protocol*.

Two directions may be envisioned for wireless appliances. On the one hand, they will allow data access (for example Web access) anywhere at anytime. On the other hand, it could be interesting to investigate how to use them in order to access information system set up from the nearby user just for the current situation. In the ACES project, we study both approaches.

4.3.1. Anytime and anywhere data access.

In the next years global mobile networks will be more and more heterogeneous. High data rate (from a few hundred of Kb/s to several Mb/s) will not be available everywhere. Global coverage will be provided by low data rate cells (GSM, GPRS), and limited areas with strong densities of population will be covered with high data rate cells (Bluetooth, wireless LANs, 3G networks).

Mobile users want to access services anytime anywhere. In the context of heterogeneous cells one of the main challenges is to offer services that combine data content richness and short time delivery. This application domain has the ambition to demonstrate that contextual awareness can enhance such services offered by telecommunication operators. Two main points have to be addressed:

- To make the network heterogeneity beneficial to user, i.e. to take benefit of high data cells when they are available for users.
- To deliver data/services as soon as possible: it implies to combine contextual information (for example the location of the user in the network) with user data and services information.

4.3.2. Data access in the physical neighborhood.

To illustrate this application domain, let us consider the following example: people are attending a conference. Several sessions can occur simultaneously. Attendees are moving freely from one session to another, depending on the subject and their interests. We assume that each participant has it own wireless appliance (w-PDA in the following) with some facilities to exchange information with others in a limited range. Each w-PDA manages information, such as the owner's identity, their domain of interests, a copy of its slides, bibliographic references and URLs... In such a conference context, each physical encounter between two people could be a nice opportunity to exchange information between their w-PDA. Due to user mobility, this information exchange can take place only during a limited duration that can not be statically known. In fact, we can consider that when some w-PDA's are close enough to be able to communicate, a spontaneous information system exists between these w-PDA's. Briefly, the concept of Spontaneous Information System (SIS) can be defined as follows: a SIS is composed of mobile users equipped with a w-PDA. A SIS takes place as soon as, at least, two w-PDA's are close together (communication range of the wireless interface). Then, these w-PDA's can exchange implicitly or explicitly some information. A SIS changes (and even disappears) as soon as a w-PDA moves away one from each other. In such a context, our purpose is not to provide mechanisms to mask possible disconnection during a time period, because two w-PDA's which are cooperating at time t might not at all be able to cooperate at time t' due to their movement. In other words, the physical mobility of entities belongs to the "semantics" of the considered applications. Then, we claim that the main challenge to solve in that context is to use in the most effective way the limited communication period between w-PDA's of a SIS, assuming that this period cannot be determined *a priori*. It depends of many parameters such as the speed of the entities, communication range, communication reliability, ...

More precisely, a SIS is based on a progressive information integration scheme: information are added or deleted at any time according to w-PDA's movement. Information available from the SIS is distributed over the participating w-PDA's at time t. We call all these information the visible space of the SIS. This space must reflect dynamically at any time the set of information that are present in the SIS. Due to the limited communication time available during an "encounter", it is important to reach quickly the most relevant information (see section 6.3.2). Another point that seems natural to consider is the "fragility" of the user's physical neighborhood (see section 6.3.2), which we characterize as its remaining "time to live", depending on the users physical mobility. In many cases, links will die before being able to finish the on-going communications.

4.4. Information systems based on spatial programming

Key words: *Spatial navigation*, *physical programming*, *wireless communications*.

We call spatial programming an environment dedicated to programming with the physical approach (see paragraph 3.3.1). The data are physically attached to the objects of an application and move with them. Each data has a physical shape that defines the area where it is accessible. A process can access a data if it is inside the data's area. Each implied object executes at least one process of the application. The data manipulated by a process is addressed and accessed in the physical space. Processes' memory is the physical space. A process can read a value from the memory and/or add a new data inside the memory. A process can delete only its data. The visible data space for a process changes when it moves. The read access is a blocking operation, processes stay blocked until they are in the area of the expected data. Data is "recognized" by its type that can be a simple type like a C integer, or a composed type like a C structure. The blocking aspect of the read accesses is used to synchronize the execution of the processes on the position of the corresponding objects.

An information system based on spatial programming is distributed over objects. Since the network interfaces have a limited range, each object has a partial view of the global situation. In this way, an object can interact only with the proximate objects (see sections 5.8 and 6.3.3), other applications that must see all the objects are not programmable with this approach.

5. Software

5.1. Introduction

The research tasks conducted in the ACES project lead to the development of many softwares. These developments are mainly realized, or at least initialized within the framework of industrial collaborations, and so they are attached to the application domains covered by the project.

5.2. Scratchy Java execution machine

Members: *Michel Banâtre, Gilbert Cabillic [contact], Jean-Philippe Lesot, Salam Majoul, Frédéric Parain, Jean-Paul Routeau.*
Status: *Internal to the team. Deposit of version 1.0 of April 9 2003 by INRIA to "Agence pour la Protection des Programmes (APP)". Number: IDDN.FR.001.260058.000.S.P.2003.000.10000.*
This software has been done under the contract described in section 7.1.1. The objective of this development is to design and evaluate a Java runtime environment suitable for embedded architectures. This environment is compatible with MIDP 2.0 Java APIs compatible. It has been written from scratch and is designed modularly to enable the deep analysis and modification of each function of the machine. This way it is easy to work on the tradeoff on the whole machine (energy consumption, memory usage, cpu need).

At last, a particular layer has been designed to mask the underlying operating system interface. Thanks to that interface, our machine has been easily ported on two POSIX based operating systems (Windows and Linux) and on embedded operating system (vxWorks).

5.3. Scratchy Development Environment

Members: *Michel Banâtre, Gilbert Cabillic [contact], Jean-Philippe Lesot, Salam Majoul, Frédéric Parain, Jean-Paul Routeau.*
Status: *Internal to the team. Deposit of version 1.0 of April 9 2003 by INRIA to "Agence pour la Protection des Programmes (APP)". Number: IDDN.FR.001.260059.000.S.P.2003.000.10000.*
This software has been done under the contract described in section 7.1.1. The objective of this development is to provide a way to develop a Java runtime with modularity. Moreover, pieces of code of the Java machine can be written in different languages. During the building, some optimizations are done like data structure compaction, or procedure call optimization.

5.4. Heptane

Members: *Mathieu Avila, Isabelle Puaut [contact].*
Status *: Open source software (GPL license), available at* http://www.irisa.fr/aces/software
The aim of HEPTANE *is to produce strict upper bounds of the execution times of applications developed in C. It is targeted at applications with* hard *real-time requirements (automotive, railway, aerospace domains).*

HEPTANE *is a so-called* "tree-based" *timing analyzer: the calculation of a program's WCET highly relies on the program syntax, and uses the program syntax tree, generated from the program source code. Timing equations compute the WCET recursively in a bottom-up manner thanks to the program syntax tree. To illustrate the principle of tree-based static WCET analysis, an extremely simplified example of timing equation for a conditional construct* "CO" *with code* $"if(C)S; elseE;"$ *is* $"WCET(CO) = WCET(C) + max(WCET(S), WCET(E))"$. *Finding the WCET of a loop requires a knowledge of the maximum number of loop iterations.* HEPTANE *currently requires the user to give this information through* symbolic annotations *in the source program. Annotations are designed to support non rectangular and even non-linear loops (nested loops whose number of iterations arbitrarily depends on the counter variables of outer loops) [30]. The final WCET is computed using a symbolic evaluation tool (currently Maple and Maxima are supported).*

Efficiently modeling the microarchitecture allows to reduce the pessimism of static WCET analysis. HEPTANE *integrates mechanisms to take into account the effect of* instruction caches, pipelines *and* branch prediction. *The modeling of the* instruction cache, branch predictor *and* pipeline *produce results expressed in a microarchitecture-independent formalism, thus allowing* HEPTANE *to be easily modified or retargeted to a new architecture.*

HEPTANE *is designed to produce timing information for monoprocessor architectures (currently Pentium, and MIPS – with an overly simplified timing model). Other platforms will be evaluated in the future.*

5.5. Artisst

Members: *David Decotigny, Isabelle Puaut [contact].*
Status *: Open source software (LGPL license) available at* http://www.irisa.fr/aces/software
Artisst (Artisst is a Real-Time System Simulation Tool) is a free (LGPL) framework designed to simulate and evaluate distributed or centralized real-time systems. Contrary to most existing real-time systems simulators, it allows to simulate complex systems made of tasks performing arbitrary computations and exhibiting a complex and realistic pattern for their arrival law, synchronization relations, and execution time.

This is mainly due to the fact that both the RTOS (including its scheduler and interrupt handlers) and the application are written in a general purpose programming language (C or C++), thus allowing to implement a simulation which is very close to a real working application, by eventually reusing existing code. Furthermore, as a side effect and thanks to the modular and extensible object-oriented architecture of Artisst, the simulator is not dedicated to a particular Operating System API, but is fully customizable instead.

Artisst actually takes the form of a C/C++ library. It provides the implementation of the discrete event simulation subsystem (simulation messages, module interface), and a series of default modules, including the module in charge of simulating a real-time system made of an RTOS and of tasks.

As a result of a simulation, one can "see" the behavior of the system in the form of a Gantt chart, or through statistical analysis of a series of given evaluation metrics. One advantage of using Artisst is the ability to test the behavior of the application with different schedulers, or to test different applications which are functionally equivalent. Another advantage of Artisst over other real-time systems is its ability to take all the system overheads (scheduler, interrupt handlers, other RTOS services) into account.

5.6. Mobile terminal for new services over GPRS/802.11b networks

Members: *Michel Banâtre, Carole Bonan, Gilbert Cabillic, Jean-Paul Routeau, Julien Sevin, Grégory Watts, Frédéric Weis [contact].*
Status *: Internal to the team.*
This software has been developed under the contract described in section 7.1.2. The main objective here is to define and experiment new contextual services for mobile heterogeneous networks (see section 4.3). During the first phase of this INRIA/ALCATEL collaboration we have studied and implemented a new messaging service (based on the MMS standard [45]) on a mobile terminal. Two main constraints have been considered for the design of the terminal: (i) it must be able to support two network interfaces, GPRS and 802.11b. (ii) These two interfaces must be accessed through an "open" development environment. The term "open" means that the development environment allows to implement the network interface selection, and an MMS advanced application. We have developed the software components that adress these two constraints.

For the development environment, a J2ME implementation, which is MIDP 2.0 [34] specification compliant, has been chosen. J2ME is a Java runtime environment optimized for embedded devices. The system mechanism for dynamic interface selection is based on an extension of the existing MIDP specification: two new classes have been implemented in the terminal ($httpGPRS$ and $httpWiFi$) and linked with the two corresponding network interfaces.

This terminal and the associated messaging service have been integrated to the common INRIA/ALCATEL testbed platform (see [24] for further details).

5.7. Spontaneous Information System testbed

Members: *Michel Banâtre, Paul Couderc, David Touzet, Arnaud Troël, Grégory Watts, Frédéric Weis [contact].*
Status *: Internal to the team.*
The aim of this platform is to provide all needed mechanisms for experimenting applications based on proximate interactions (see section 4.3): evaluation of a user's physical neighborhood, management of the communication between two nodes (in spite of users' mobility), progressive information discovery to build the visible space of the SIS, representation of pertinent information in this visible space, data access inside this visible space... The testbed is implemented with PocketPC PDAs running Windows CE 3.0, and using IEEE 802.11b wireless interface. All the system mechanisms have been writen in C++. They have been used to develop and experiment several ubiquitous applications based on proximate interactions, including PERSEND, a continuous querying system of the visible space (see section 6.3.2).

5.8. Spread

Members: *Michel Banâtre [contact], Mathieu Bécus, Paul Couderc.*
Status: *Internal to the team. Deposit of version 1.0 of March 2003 by INRIA to "Agence pour la Protection des Programmes (APP)". Number: IDDN.FR.001.170029.000.S.P.2003.000.30800.*
SPREAD (Spatial PRogramming Environment Ambient computing Design) is a software system enabling simple and elegant programming of ubiquitous computing applications. SPREAD is based on the concept of physical "tuples", which are structured pieces of data associated with a shape surrounding a physical object. SPREAD allow processes to use the physical space as a simple database or associative memory, where data are represented by physical objects and data flow are implicitely related to object movements.

The system offers a simple yet powerful API to applications. This API allows to read and write tuples in the physical space, and to synchronize operations. SPREAD is based on a wireless peer-to-peer architecture, where all the participating objects host an instance of the system and the relevant application components. The targeted execution platform for SPREAD nodes are very low cost: low power devices integrating a CPU, some memory, and a short range wireless communication capability.

Currently, there are two implementations of the SPREAD system. The first one is a native Windows CE 3.0 version. PocketPC PDA's can be use for application prototyping, using either WLAN (IEEE 802.11b) wireless interface or Bluetooth radio. This version consists of 5000 lines of C/C++ code. The memory footprint of the SPREAD engine is 50 KB.

The second version is written in Java, and runs on a J2ME compliant platform. It consists of 2000 lines of Java code, and its memory footprint is of 30 KB (excluding the JVM).

SPREAD has been used to develop and experiment several ubiquitous computing application including Ubi-Bus, an operational urban transportation assistant for disabled people.

6. New Results

6.1. Introduction

The research objectives of ACES are articulated around two main topics (already briefly described in section 2.1): (i) The first one relates to the design of embedded systems, and (ii) the second one is directed towards the study of system supports for ubiquitous computing.
(i) An embedded architecture is generally characterized by a strong link with the external environment in which it evolves, which requires to take into account real-time constraints. Another important feature of an embedded architecture comes from its limited resources in terms of storage capacity, power consumption, available bandwidth for the communications with the external world. A preliminary step for designing an embedded system is to predict the amount of resources required by the embedded software. This characterization is

necessary for dimensioning embedded architectures, and also for verifying that the system will meet its real-time and resource constraints (see paragraph 6.2.1). For the effective design, we examine at the same time the low layers of the operating systems (the core of the system, which is strongly connected with the underlying embedded architecture) and the middleware layer like the Java virtual machine (see paragraph 6.2.2). We also study availability problems, to be able to execute simultaneously several types of applications with different constraints (see paragraph 6.2.4). Finally, we focus on the performance evaluation of the embedded systems (see paragraph 6.2.5).
(ii) The second activity of the ACES project is directed towards the study of system supports for ambient computing, with two main objectives. The first one is to build information systems with wireless appliances. To explore this challenging field of research, we have followed two complementary directions: new contextual services for large heterogenous networks (see paragraph 6.3.1), and applications using only proximate interactions (see paragraph 6.3.2). Finally, we explore the concept of spatial information system, which has a great potential for programming easily new ubiquitous applications (see paragraph 6.3.3).

6.2. Design of embedded systems

6.2.1. Predicting resource consumption of embedded software: static worst-case execution time (WCET) analysis and beyond

Members: *Alexis Arnaud, David Decotigny, Isabelle Puaut.*
Predicting the amount of resources required by embedded software is of prime importance for verifying that the system will fulfill its real-time and resource constraints. A particularly important point in the framework of hard real-time embedded systems is to predict the Worst-Case Execution Times (WCETs) of tasks, so that it can be proven that task deadlines will be met.

Our ongoing research concerns methods for obtaining automatically upper bounds of the execution times of applications on a given hardware. Static analysis of the source code of applications is used to identify their worst-case execution scenarios; static analysis methods have been preferred to testing because the latter class of methods requires to explore all possible inputs for a piece of software to identify its longest execution path. Hardware models of processors are used to obtain WCETs instruction sequences. The use of hardware models instead of the actual hardware allows to use static WCET analysis methods earlier in the application development life-cycle. In addition to allowing the verification of deadlines, static WCET analysis methods can help in selecting or dimensioning the hardware used in hard real-time embedded systems, and can serve at comparing different implementation strategies of applications. A result of our research on static WCET analysis is the open-source analyzer Heptane (see paragraph 5.4) aimed at obtaining WCETs on processors with in-order execution, equipped with caches and pipelined execution. The modularity of Heptane allows to port it to different target processors and programming languages.

Our most recent results concerning worst-case execution time analysis are twofold. On the one hand, we proposed analysis techniques (cache locking algorithms) for using caches in hard real-time systems; the objective here is to both take benefit of the performance enhancement provided by caches and to use caches in a predictable manner in order to be able to prove the system temporal correctness. On the other hand, we worked on on-line techniques (gain-time identification techniques) to reuse the spare CPU capacity which results from overestimated WCETs.

6.2.1.1. WCET analysis: use of cache locking to reconciliate performance and predictability.

Cache memories have been extensively used to bridge the gap between high speed processors and relatively slow main memories. However, they are sources of predictability problems because of their dynamic and adaptive behaviors, and thus need special attention to be used in hard real-time systems. A lot of progress has been achieved in the last ten years to statically predict worst-case execution times (WCETs) of tasks on architectures with caches. However, cache-aware WCET analysis techniques are not always applicable due to the lack of documentation of hardware manuals concerning the cache replacement policies. Moreover, they tend to be too pessimistic with high degrees of associativity or with some cache replacement policies (e.g.

random replacement policies). An alternative approach allowing to use caches in real-time systems is to lock their contents such that memory access times and cache-related preemption times are predictable.

We proposed a simple and easy to implement method for *static locking* of instruction and data caches, ensuring by construction that the latencies of all memory accesses are predictable [38]. The method can be applied to any cache provided that its size and associativity degree are known (no knowledge of the cache replacement policy is needed). The method can be applied even if the tasks contain statically unknown memory references (using pointers, stack allocated or dynamically allocated data structures). Furthermore, it does not require any modifications of the task code.

Experimental results show that the use of locked caches enhances the system performance compared to a system without a cache. However, such a scheme may result in a performance loss as compared to a system with a dynamic (unlocked) cache. We then identify the threshold under which this performance loss is acceptable and above which more efficient but more complex locking schemes have to be devised. Our ongoing work [6] is to extend our work towards more dynamic locking schemes. Instead of selecting cache contents for the entire execution of a task, the idea is to select different contents of the locked cache changed at statically-defined points in order to cope with the tasks dynamic behavior while staying predictable. Cache contents selection will use the strings of memory references issued by a task to define the best cache locking points so as to minimize the tasks WCETs. The expected result to find a better tradeoff between performance and predictability, and this without requiring any knowledge of the cache replacement policy.

6.2.1.2. WCET analysis: gain time identification to cope with overestimated WCETs.

WCET estimates obtained using static analysis methods are getting increasingly pessimistic as the complexity of hardware and software increases. The difference between the WCET of one task (estimated off-line) and its actual execution time (only known on-line) is known as *gain time*. Identifying gain time as soon as possible is important because it increases the number of tasks that can be accepted dynamically. While some research has already been undertaken for the identification of gain time, few work has considered the impact of gain time identification and reclaiming on static WCET analysis methods. We introduced in [7] three classes of methods for gain time identification, and discuss their impact on tree-based static WCET analysis methods.

6.2.2. Operating systems for embedded appliances

Members: *Michel Banâtre, Gilbert Cabillic, Jean-Philippe Lesot, Salam Majoul, Frédéric Parain, Jean-Paul Routeau, Pushpendra Singh.*

A Java Execution Environment (JEE) presents several advantages for embedded architectures. First, Java applications can be dynamically downloaded. Second, as Java bytecode is an intermediate code, the application can be distributed everywhere. Moreover, a Java application cannot explicitly manage memory access because Java bytecode provides no way to express and manipulate pointers in Java world. This increases the stability of all the Java world by avoiding memory access errors due to software faults or illegal intrusion. Of course, this stability depends on the JEE (Java virtual machine (JVM), and APIs implementation), and also on the underlying operating system.

The software architecture we have designed is presented in figure 1. Our Java platform (Java virtual machine and Java APIs), named Scratchy, runs on the hardware through an operating system. Scratchy is executed in several OS threads in one common space address. Scratchy has the ability to execute several applications at the same time and a scheduler inside Scratchy undertakes the scheduling of all Java threads. Native code, independent of Java world like protocol layers or driver daemons, is executed in other OS threads thanks to the help of the operating system.

We present in that section new results obtained in that area with the global objective to build a efficient Java execution environment for embedded appliances (Wireless PDAs). It is organized as follows. First, we present our Scratchy modular software architecture and our Scratchy runtime prototype. Then, we present our ongoing work about designing an operating system written in Java. At last we explain our work done in the area of availability.

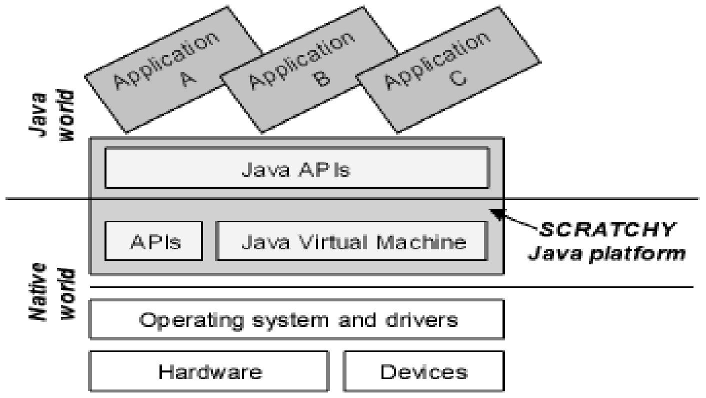

Figure 1. Global software architecture

6.2.3. Scratchy Software Architecture

Memory volume, execution time and energy consumption are critical resources for embedded systems. A flexible Java environment providing a good tradeoff between these resources is required for wireless PDA's. Only a modular approach, whose benefits have been already proven in software design, allows to achieve such a tradeoff. In fact, through modularity it is possible to specialize some parts of the JVM for a specific processor by exploiting, for instance, low power features for DSP. With a monolithic-programmed Java environment, it is difficult to change at many levels hardware resource management without rewriting the JVM from scratch for each platform.

To reach these goals we designed a modular approach described in the following. A module contains services (functions) and data type structures. Pre-hooks and post-hooks can be attached to a service. The formers are called before a service call to undertake for example resource reservation and allocation. Post-hooks are called after a service call for instance to free the resources and manage errors.

The Scratchy Development Environment (SDE), is designed to achieve modularity for our Java environment. This tool is designed to optimize time or memory overhead on the outcome source and to be the most language and compiler independent as possible. SDE takes four inputs:

- A global specification file describes services and data types by using an Interface Definition Language (IDL).
- A set of modules implements services and data types in one of the supported language mappings.
- An implementation file inside each module describes the link between specification and implementation.
- Target hardware descriptions indicate alignment constraints with their respective access cost for each target processor.

SDE chooses a subset of modules according to the targeted hardware criteria. It generates, as shown in figure 2, stubs for services, sets structures of data types, generates functions to dynamically allocate and access data types, etc. SDE works at source level, so it is the responsibility of compilers or preprocessors (through in-lining for example) to optimize a source which has potentially no overhead.

The current version of SDE supports the C language as language mapping and generates stubs for services and data type structures in that language. In order to evaluate memory expansion of SDE data type structures, their memory allocation time and service execution time, we compared the generated stubs to an equivalent program written in C. For this purpose, we compared the assembler codes obtained after optimization for both programs. We used benchmarks when the assembler codes were not comparable. The results obtained were

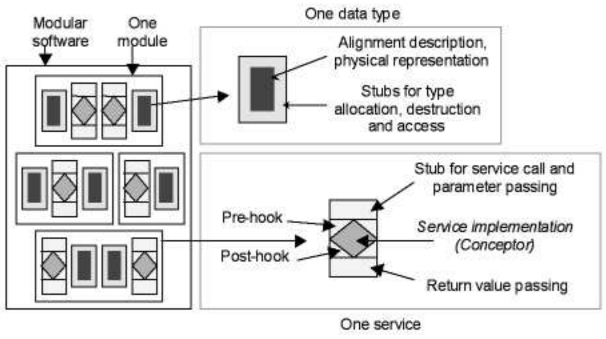
Modular software
One module
One data type
Alignment description, physical representation
Stubs for type allocation, destruction and access
Stub for service call and parameter passing
Pre-hook
Post-hook
Service implementation (Conceptor)
Return value passing
One service

Figure 2. Our modular approach

very promising as the generated stubs introduce an insignificant overhead and in some cases no-overhead at all. For example, the assembler codes of a C program calling the generated stub for a service without parameter and the C program calling directly the C function implementing that service are identical. When the service is specified with parameters, the assembler code for the SDE program contains only one supplementary statement relating to a shift of the stack pointer. We made some benchmarks to evaluate data type structures generated by SDE. We used a Pentium II processor under Linux system and using a Gnu C compiler and optimizer. As an example, figure 3 shows the time required to allocate an array generated by SDE and an equivalent C array. The time measured to access an array element is the same in both cases.

Array size	C Program (cycles)	Using SDE (cycles)	Overhead (cycles)
20	229	228	+1
80	228	228	0
280	228	228	0
400	228	228	0

Figure 3. Array allocation time comparison

6.2.3.1. Scratchy Java Virtual Machine.

Scratchy is our modular implementation of the JVM which is written from scratch in order to validate our modular approach. It relies on a MIDP [34] specification compliant. Pentium, ARM and a TI DSP TMS320C55x targets are supported by our JVM as well as a bi-processor Pentium based architecture. To make the link with an operating system, we designed a module named middleware which supports not only a Posix general operating system (Linux, Windows 2000) but also a real-time operating system with Posix compatibility (VxWorks). We are currently studying a way to design a hardware and software tradeoff for a Java environment compatible with our Scratchy Java Virtual Machine.

6.2.3.2. Building a Java operating system.

Our approach consists in transferring some native code of the JEE software in Java language aiming at exploiting the software confinement features of Java to increase the overall stability of the platform. At the opposite of current Java operating system projects like J-Kernel [28], we do not want to enable the direct use of pointers in Java. So, our approach is to provide a Java view of low-level resources without any pointer abstraction. To validate that point, we have exported the native scheduler of Scratchy in Java. Hence, we designed an abstraction for interrupt management and threads. The interface OsInterrupt specifies an interrupt handler and provides a method called handler which is invoked when an interrupt occurs. An implementation of this interface can be given at any interruption level. Each implementation has to provide its running parameters through Java fields. For example, a keyboard interrupt implementation must define a field representing the value of the pressed key. This value is set using a minimum native interrupt handler. In this way, the access of low-level IO control ports of the device is not needed and we still guarantie that no pointer is needed to write any interrupt handler in Java side. The class OsTimerInterrupt is designed to be associated with the low level timer interrupt and enables the scheduler to realize its policy. The OsThread class abstracts a Java thread context and provides the scheduler a setThreadActive method to execute its elected OsThread on the CPU. The scheduler performs a round-robin policy that manages all Java threads. It was very easy to be implemented and we showed that it is possible to transfer native code of the JEE in Java world. Our ongoing work focuses on transferring more complex JEE native software (including drivers) like graphics native APIs, sound APIs and also the garbage collector, in order to obtain the smallest native operating system needed for a JEE.

6.2.4. Availability

Today handheld devices are able to perform various common applications of mobile computing such as gathering of information, communication with other devices, making financial transactions, and running downloaded video/audio applications.

However, unlike wired environment, mobile environment has characteristics like lack of resources, frequent disconnections, limitation of bandwidth, and high mobility. Because of such constraints, faults are an integral part of wireless world. Thus providing good fault-tolerance mechanisms is necessary to ensure reliable working of mobile devices like PDA. But currently available fault-tolerance mechanisms do not provide integrated solutions, i.e. they solve one problem without considering others.

We have also observed that not much work has been done to provide a complete fault-tolerance solution for applications of different nature. For example a Mpeg application, which is showing a video clip directly from the server does not need checkpointing mechanisms but mechanisms to deal with bandwidth variation, but, a to-do list application requires checkpointing mechanisms rather than bandwidth variation.

Our approach is first to design new integrated availability mechanisms and second to define the way to transparently choose, at the loading of an application, the most suitable available mechanisms according to the application requirements.

We also designed two distributed checkpoint mechanisms adapted for appliances.

First, we designed a checkpointing algorithm suitable to deterministic applications. We used an *anti-message* technique with selective logging to achieve our aim. With the help of anti-messages, our approach avoids same computation during the recovery phase; so it provides a faster recovery and saves energy. We have limited our logging to a small period thus without causing extra overhead on mobile devices for storage. We found that our algorithm results in longer utilization of device by conserving energy then previously proposed algorithms.

Second, we designed a coordinated checkpointing algorithm for non-deterministic applications. We introduced the notion of successive checkpointing, which is based on delayed checkpointing approach, along with partial blocking to achieve a consistent checkpoint state. The algorithm reduces energy consumption by minimizing repeated computation and by avoiding useless checkpoints. For the same reason, we also have faster recovery. We also ask only minimum number of processes to take checkpoint.

6.2.5. Performance evaluation of embedded systems

Members: *Laurent David, David Decotigny, Isabelle Puaut.*

Performance evaluation is an important research topic in embedded systems because it allows to verify that the system fulfills its functional, real-time or resource constraints. The performance evaluation may be achieved using analytical methods based on models of the system (i.e. with respect to real-time requirements, schedulability analysis methods, which are based on the knowledge of a model of the worst-case system workload and on the knowledge of the tasks worst-case execution times). Alternative methods are to actually implement the target system or to rely on simulation.

6.2.5.1. Simulation infrastructure for the performance evaluation of real-time systems.

We proposed in [1] a simulation infrastructure specifically suited to the performance evaluation of real-time systems. It is named Artisst, for "*Artisst is a Real-Time System Simulation Tool*". It may be used as a complement to safe static analysis methods, especially when the temporal behavior of the system or that of its environment is not fully characterized. The infrastructure tool was designed to be as customizable as possible, and we provided it with the capacity to reuse existing application code, made it efficient, and allowed the simulated temporal behavior to be as close to the effective one as possible, thanks to the ability to adjust the timing resolution of the simulation. We also introduced a generic objet model for dynamic real-time scheduling, that can adapt to a wide variety of existing schedulers, and that has been assessed by the infrastructure.

6.2.5.2. Probabilistic schedulability analysis

Scheduling policies rely, in their scheduling decisions, on the knowledge about task characteristics such as computational times, deadlines, dependency relationships, etc. This works quite well as long as these characteristics, most importantly the execution time, of each task are fixed, or are known to have an upper bound (WCET, worst-case execution times). However, WCET estimates tend to be very pessimistic because of fluctuations in the execution times of tasks. One source of fluctuation is the use of modern processors for which the execution time of instructions is no longer constant due to features like pipelines, caches, branch prediction or out-of-order execution. Another source of fluctuation comes from the system itself, due to the presence of algorithms whose execution times depend on input data or system states. Multimedia tasks such as sampling, compression, coding, decoding, and security methods such as cryptography, fall in this category.

For the systems in which precise upper bounds of the tasks execution times cannot be obtained due to varying execution times, traditional schedulability analysis methods, based on the knowledge of worst-case execution times, are no longer generally applicable.

The objective of this research, started in September 2003, is to propose alternative (probabilistic) schedulability analysis methods suited to such systems. Methods for obtaining the distribution of execution times will be explored, based on our experience on static WCET analysis. The knowledge of the distribution of the tasks execution times will be used to provide probabilistic guarantees of the system real-time behavior.

6.3. System supports for ambient computing

6.3.1. Data delivery in heterogeneous networks

Members: *Michel Banâtre, Carole Bonan, Julien Sevin, Grégory Watts, Frédéric Weis.*

In the next years, mobile networks will be more and more heterogeneous. High data rate will not be available everywhere. In our approach, a mobile heterogeneous network consists of (i) a set of small and discontinuous high data rate cells (also named infostations or hot-spotted network), (ii) and a "legacy" network which provides a low data rate global coverage (for example a GPRS network). To support anytime and anywhere data access in such an architecture, future mobile terminals will have multiple network interfaces (see section 5.6), and the most suitable interface will be chosen by taking user's context into consideration.

The objective of this research, started in May 2003, is to propose solutions for continuous information delivery in spite of the discontinuous coverage. We have designed an approach using data prefetch mechanisms, and a distributed architecture model based on three communication links. The first link is provided by the legacy network, and is used by the mobile terminal for sending data requests to a data server. The second link is a high data rate wired network, and is used by the data server to prefetch the requested data in caches located in high data rate cells. Finally the third link is the wireless connection provided by the high data rate cells. It is used for delivering data from the cell's cache into the cache of the mobile terminal. Using this model, we will investigate the problem of determining which caches have to be loaded, according to different user's context parameters (position, trajectory...).

6.3.2. Spontaneous Information System (SIS)

Members: *Michel Banâtre, Arnaud Troël, David Touzet, Frédéric Weis.*

In the field of ubiquitous computing, short-distance wireless communication technologies make it possible to envision direct interactions between physically close enough devices. Considering each device as a potential data provider, it is now possible to build data-based applications in such environments. Due to the devices' mobility, application designers have to face continually changing sets of data providers. In such a context, enabling applications to "sense" their neighboring available data raises many challenges: (i) to provide applications with an up-to-date list of surrounding devices and (ii) to enable applications to continuously query their surrounding data providers.

6.3.2.1. Data-Based Proximate Applications.

Short-range wireless communication technologies enable to envision direct interactions between mobile devices. In the scope of data access, devices can now be considered as both data providers and data consumers. Thus, each device can be provided with a remote access to data its neighbors agree to share. Such a service enables applications to consult a set of data providers which dynamically evolves according to the mobility of the neighboring devices. The set of data sources an application may access by this way is therefore representative of its physical neighborhood. For this purpose, we have designed PERSEND [17], a tool enabling the continuous consultation of neighboring shared data.

As a large of data is today managed by databases, our system has been developed using relational databases systems (RDBMS). In this context, shared data can be modified by the mean of the data handling commands: data insertion, data removal and data update. Rather than providing a complete view of the available data, our system is designed to enable users to query these data for some specified subsets. Just as the whole set of available data, the data subsets queried by users evolve according to the set of neighboring data providers. However, they also have to reflect the conditions users specify. In order to enable applications to continually be aware of their currently available data, our system provides them with persitent data sets which match the users' conditions. For this purpose, it enlarges some works previously performed in the *continuous queries* area.

First we have studied the impact of the duration parameter introduced by continous queries applied in a SIS system. Built Continuous Result Sets (CRS) evolve according to data which are available in the devices' vicinity. Common querying languages, such as SQL, have been designed to define and build static data sets. Some of the tools and functions they provide do not succeed to manage data sets subject to variations. Thus, SQL enables users to call some aggregation functions (such as `max`, `sum`, `count`...) in the queries they define. These functions usually compute a single value from a static data set.

In order to manage changing data sets, dynamic semantics have been associated to these functions. The value these functions return has to mirror the current state of the continuous result set. For this purpose, this value has to be re-evaluated each time the continuous result set is updated. Fortunately, in many cases, the value to be returned can be computed without requiring the complete CRS to be scanned. For example, the value returned by a call to `count(*)` can be easily managed: it has to be incremented each time a row is inserted in the CRS and to be decremented for each removed row. Likewise we have defined a dynamic version of the SQL $distinct$ keyword. When specified, this keyword ensures that returned data sets are composed of distinct rows.

Then, we propose the PERSEND architecture. It is based on a Relational DataBase Management System (RDBMS). Besides the neighborhood manager, which provides applications with information on neighboring devices, PERSEND is organized around four main components: the query interface, the query parser, the update supervisor and the query descriptor manager.

6.3.2.2. Automatic neighborhood discovery.

In the field of proximate interactions, the detection and the representation of the nodes in the range of communication of another node is traditionally based on periodic presence messages. These nodes are generally supposed to have the same mobility profile and the same range of communication.

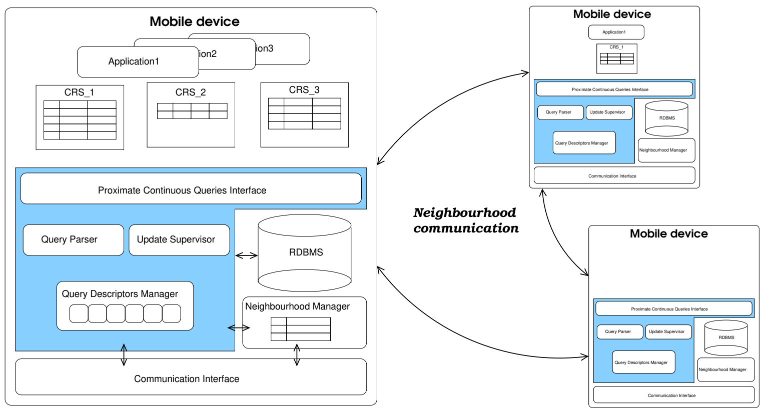

Figure 4. Architecture of the PERSEND querying architecture

First we show that these assumptions are however unsuitable to some applications using wireless appliances [23]. We also propose a new definition of the neighborhood relationship between nodes having different mobility schemes and different ranges of communication. We now consider three types of neighbors: unilateral neighbor (a terminal A is an unilateral neighbor of a terminal B if A is in the communication range of B), mutual neighbor (A and B are mutually unilateral neighbors), and strict unilateral neighbor (A is a strict unilateral neighbor of B if A is an unilateral neighbor of B, without being a mutual neighbor of B). We eventually derive from these definitions a system service allowing a new representation of the neigborhood. This service is based on the sending of presence messages, in which the list of unilateral neighbors is published. This approach allows to distinguish the different types of neighborhoods.

Then we develop a new neighborhood discovery protocol that enables the representation of these different profiles in the terminals' view of the neighborhood. We now propose a new approach based on this protocol to automatically clock the discovery process [21]. With this approach, a mobile terminal becomes able to adapt the frequency of its presence message to its mobility profile and also to the set of its unilateral neighbors. To achieve this goal, we have defined a universal parameter (R, the *detection ratio*), independent of the nodes' range. A process, based on this parameter, and able to calculate automatically the frequency for announcing the node's precence, has been proposed and evaluated.

6.3.3. Spatial Information Systems

Members: *Michel Banâtre, Mathieu Becus, Paul Couderc, Julien Pauty.*

The concept of spatial information system has a great potential for new applications, exploiting innovative computing architecture. We have been investigating both the application domain, and improvements and refinements of the SPREAD system.

Regarding SPREAD, a Java version of the system has been developed, for a broader spectrum of supported devices. We also conducted experiments to evaluate the energy consumption overhead associated with the use of SPREAD (in the WLAN case). Finally, we are working on the enhancement of the programming model to allow fine specification of geometrical shapes for tuple reading and writing. This enhancement increases the number of applications which can benefit from SPREAD. In addition, we foresee the potential for more efficient protocols taking advantage of the knowledge of spatial coverage of tuple operations. This enhancement requires relative positionning of nodes, which is not supported by standard communication technology like WLAN or Bluetooth. We plan to use GPS positioning in order to simulate relative positioning, while waiting for an appropriate local positionning technology (like UWB).

Regarding applications, a whole new class of spatial information systems has been recently introduced, with many practical applications. It consists of exploiting wireless communication for service preprocessing. These applications show an innovative use of the concepts of spatial information systems for increased efficiency, ease of use or security in existing service enhanced by ubiquitous computing. Once again, the SPREAD system shows its versatility as implementing this class of new application is simple and elegant using the SPREAD model.

7. Contracts and Grants with Industry

7.1. National contracts

7.1.1. Texas Instruments

- Number: 198C2730031303202
- Title: Real time Java Distributed Processing Environment
- Related Research activity: see section 6.2.2
- Partner: *Texas Instruments*
- Founding: *Texas Instruments*
- Starting: 01/10/1998, ending : 30/09/2001
- Extension starting: 01/10/2001, ending : 30/09/2003
- Extension starting: 01/10/2003, ending : 30/09/2006

The objective of that contract is to design and build a Java runtime suitable for embedded platforms on Texas Instruments hardware architectures.

7.1.2. Alcatel

- Number: 101C078
- Title: Adaptive service delivery in heterogeneous mobile networks
- Related Research activity: see section 6.3.2
- Partner: *Alcatel*
- Founding: *Alcatel*
- Starting: 01/09/2000, ending : 30/08/2002
- Extension starting: 01/09/2002, ending : 30/08/2004

The objective of that contract is to design, build and experiment an efficient service data delivery in heterogeneous mobile networks. The targeted solution has to be distributed between the network components and the mobile terminal.

8. Other Grants and Activities

8.1. European actions

8.1.1. Programme d'action intégré Picasso

- Title: Use of cache memories in hard real-time systems
- Partners: Irisa - Universidad Politecnica de Valencia
- Starting: January 2004, ending: December 2004.
- Note: submitted project, under review

The objective of this bilateral cooperation is to explore methods such that caches can be used in a both efficient and predictable manner in hard real-time systems.

9. Dissemination

9.1. Animation of the scientific community

9.1.1. Program committees

- PC member of IST Mobile and Wireless Communications Summit 2003, June 2003 (M. Banâtre)
- PC member of ECRTS 2003 (*15th Euromicro Conference on Real-Time Systems*), July 2003, and ECRTS 2004, July 2004 (I. Puaut)
- PC member of the International conference on Object-Oriented Real-Time Distributed Computing (ISORC), May 2003 (I. Puaut)
- PC member of the 3rd French conference on operating systems (CFSE'3), October 2003 (I. Puaut)
- PC member of the French conference on real-time systems (RTS 04) (I. Puaut)
- PC member of the ACM SIGPLAN 2004 Conference on Languages, Compilers, and Tools for Embedded Systems (I. Puaut)
- PC member of the International Workshop on Wireless Ad Hoc Networking (WWAN 2004), March 2004 (F. Weis)
- PC member of the 1st French conference "Journées Francophones Mobilité et Ubiquité", June 2004 (F. Weis)

9.1.2. Organizing and reviewing activities

- Organizer of the 3rd workshop on worst case execution time analysis, july 2004 (I. Puaut)
- Member of the reviewing committee of the special issue of the French journal on computer science *Technique et Science Informatique* on real-time systems, spring 2003 (I. Puaut)
- Member of the scientific committee of the CNRS French summer school on real-time systems, september 2003 (I. Puaut)

9.1.3. National responsibilities

- I. Puaut is secretary of the French chapter of the ACM-SIGOPS special interest group on operating systems (ASF), since october 2002
- Member of the committee attributing the prize of the best French PhD thesis in the domain of operating systems (september 2003, I. Puaut)

9.2. National and international working groups

- Participation to the International Working Group on Timing Validation set up in the framework of the Artist European network of excellence (I. Puaut)
- Participation to the CNRS "Actions Spécifiques"
  - SoC, real-time operating systems and multiprocessors (SoC RTP) (G. Cabillic, I. Puaut)
  - Compilers for embedded systems (I. Puaut)
  - Methods for the design and verification of operating systems (I. Puaut)
- Participation to the RIS (Réseau d'Ingénierie de la Sûreté de fonctionnement) working group on the use of open source software in systems with high dependability requirements. The results of the working group can be found in the Hermes book [4]. (I. Puaut)
- Participation to the GdR I3 (Information - Interaction - Intelligence), group Mobility and Ubiquitous Computing (F. Weis)

9.3. Teaching activities

- Ifsic
  - Responsability of the optional lecture on Operating Systems in masters in Computer Science (M. Banâtre, G. Cabillic, F. Weis),
  - Responsibility of the lecture on Distributed Operating Systems in "Diic 3 ARC" (final year of masters) (M. Banâtre et F. Weis),
  - Lecture in "DESS CCI" (French equivalent of Master's Degree) on Operating Systems and Embedded JAVA (G. Cabillic).
- Ecole des Mines de Nantes
  - Responsability of the lecture on distributed systems (final year of masters) computer science department (M. Banâtre),
  - Lecture on embedded JAVA (final year of masters) computer science department (G. Cabillic).
- INSA of Rennes
  - Responsibility of the lecture on Distributed Operating Systems (final year of masters) computer science department (M. Banâtre),
  - Responsibility of the optional lecture on real-time and embedded systems (final year of masters) computer science department (I. Puaut),
  - Lecture on real-time operating systems, electrical engineering department (I. Puaut).
- University of Rennes I, UFR "Structure et Propriétés de la Matière", Lecture in DESS "DRI" (French equivalent of Master's Degree) on home networking (F. Weis).
- ENSEIRB (Bordeaux), Conference on Mobile communications and ambient computing, final year of masters, November 2003 (M. Banâtre).

We also participate to the computer science *commissions de spécialistes* of universities and schools: INSA de Rennes, université de Rennes I, Université de Bretagne Sud (I. Puaut).

9.4. Internship supervision

We have supervised the following internships in 2003:

- Anne-Sophie Adde (final year of DIIC ARC, IFISIC)
- Xioaming Du (masters student, University of Rennes)
- Maxime Glaizot (masters student, University of Rennes)
- Sylvain Guillemot, (final year of "Ecole des mines de Nantes")
- Damien Martin-Guillerez, (Ecole Normale Supérieure de Cachan)

9.5. Seminar

The members of the research group gave presentations in a number of conferences and workshops (see the list of references for further details). Other talks have been given in the following manifestations:

- Worst-case performance evaluation of real-time operating systems, working day on methods for the design and validation of operating systems organized by the French chapter of ACM-Sigops, March 2003 (A. Arnaud)
- Spontenous Information Systems, rencontres INRIA/Industrie, Applications de l'informatique et de l'automatique aux transports, January 2003 (F. Weis)
- Unleashing Context-aware Application with Spatial Programming, International Forum on 4th Generation Mobile Communications, King's College, London, May 2003 (F. Weis)
- Ubiquitous Computing and Wireless Communications, INRIA working day "Dynamic Networks", Porquerolles, France, September 2003 (F. Weis)
- Ubiquitous Computing and Mobility, seminar at the University of Montreal, IRO Department, Montreal, October 2003 (M. Banâtre)
- State of the art of Worst-Case Execution Time Analysis Methods, CNRS summer school on Real-Time Systems, Toulouse, September 2003 (I. Puaut)

9.6. Thesis committees

- Andreas Ermedahl, june 2003, Uppsala university (Sweden), A Modular Tool Architecture for Worst-Case Execution Time analysis, member of the thesis committee (I. Puaut)
- Tahar Jarboui, LAAS/CNRS, Toulouse (France), may 2003, Sûreté de fonctionnement des systèmes informatiques - Etalonnage et représentativité des fautes", LAAS, Toulouse, "rapporteur" (I. Puaut)
- Ronan Amicel, Université de Rennes I, january 2003, Simulateurs de jeux d'instruction à hautes performances, member of the thesis committee (I. Puaut)

9.7. Reviewing activities

- External reviewer for a 4-year long research project submitted to the Fund for scientific research of Flanders, Belgium, march 2003 (I. Puaut)
- External reviewer of a project submitted to the CNRS ACI on secure systems, may 2003 (I. Puaut)

9.8. Patents

- G. Cabillic, F. Parain, M. Banâtre, JP. Lesot, JP. Routeau, S. Majoul, G. Chauvel, S. Lasserre, D. D'Inverno, M. Kuusela. « *Dynamically Changing The Semantic Of An Instruction.* » European patent field on June 19th 2003, number: 03291502.7.
- G. Cabillic, F. Parain, M. Banâtre, JP. Lesot, JP. Routeau, S. Majoul, G. Chauvel, S. Lasserre, D. D'Inverno, M. Kuusela. « *Embedded Garbage Collection.* » European patent field on June 19th 2003, number: 03291506.8.
- G. Cabillic, F. Parain, M. Banâtre, JP. Lesot, JP. Routeau, S. Majoul, G. Chauvel, S. Lasserre, D. D'Inverno, M. Kuusela. « *Memory Allocation In a Multi-Processor System.* » European patent field on June 19th 2003, number: 03291501.9.
- G. Cabillic, F. Parain, M. Banâtre, JP. Lesot, JP. Routeau, S. Majoul, G. Chauvel, S. Lasserre, D. D'Inverno, M. Kuusela. « *Accessing Device Driver Memory In Programming Language Representation.* » European patent field on June 19th 2003, number: 03291505.0.

- G. Cabillic, F. Parain, M. Banâtre, JP. Lesot, JP. Routeau, S. Majoul, G. Chauvel, S. Lasserre, D. D'Inverno, M. Kuusela. « *Management Of Stack-Based Memory Usage In a Processor.* » European patent field on June 19th 2003, number: 03291504.3.
- G. Cabillic, F. Parain, M. Banâtre, JP. Lesot, JP. Routeau, S. Majoul, G. Chauvel, S. Lasserre, D. D'Inverno, M. Kuusela. « *Unresolved Instruction Resolution.* » European patent field on June 19th 2003, number: 03291503.5.
- M. Banâtre, P. Couderc, G. Cabillic. « *Dispositif et procédé de gestion de données entre équipements de communication en vue de l'obtention d'un service* ». Submitted french patent.

9.9. Industrial transfers

The major industrial transfer that we have recently realized, has been conducted with Texas Instrument, a company with which we have a collaboration since five years. Texas Instrument has created in Rennes a "Java competence" center. The goal of the latter is to support the industrialization of research results. The targeted fields are attached to the mobile telephony, and more generally to embedded systems. At the present time all the members of this center come from the ACES project.

10. Bibliography

Doctoral dissertations and "Habilitation" theses

[1] D. Decotigny. *Une infrastructure de simulation modulaire pour l'évaluation de performances de systèmes temps-réel.* Ph. D. Thesis, Université de Rennes I, April, 2003.

Articles in referred journals and book chapters

[2] G. Cabillic, I. Puaut. *Design of Embedded Systems : New Challenges.* in « ERCIM News Journal », January, 2003.

[3] A. Colin, I. Puaut, C. Rochange, P. Sainrat. *Calcul de majorants de pire temps d'exécution : Etat de l'art.* in « Techniques et Sciences Informatiques », number 5, volume 22, 2003, pages 651–677.

[4] P. David, H. Waeselynck. *Logiciel libre et sureté de fonctionnement : Cas des systèmes critiques.* Hermes, 2003, chapter Participation aux chapitres *expériences d'utilisation des logiciels libres* et *validation*, (I. Puaut).

[5] D. Touzet, F. Weis, M. Banâtre. *Architectures pour l'ubiquité numériques.* in « accepted for publication in Techniques et Sciences Informatiques », 2004.

Publications in Conferences and Workshops

[6] A. Arnaud, I. Puaut. *Towards a predictable and high performance use of instruction caches in hard real-time systems.* in « Proc. of the work-in-progress session of the 15th Euromicro Conference on Real-Time Systems », pages 61–64, Porto, Portugal, July, 2003, http://www.irisa.fr/aces/doc/ps03/ecrts03_wip.pdf.gz.

[7] M. Avila, M. Glaizot, I. Puaut. *Impact of automatic gain time identification on tree-based static WCET analysis.* in « Proc. of the 3rd International Workshop on worst-case execution time analysis, in conjunction with the 15th Euromicro Conference on Real-Time Systems », Porto, Portugal, July, 2003, http://www.irisa.fr/aces/doc/ps03/ws_wcet.pdf.gz.

[8] G. CABILLIC. *Techniques de minimisation de la consommation d'énergie dans les systèmes embarqués.* in « Actes de l'école Architectures des systèmes matériels enfouis et méthodes de conception associées », March, 2003.

[9] P. COUDERC, M. BANÂTRE. *Ambient computing applications: an experience with the SPREAD approach.* in « 36th Hawaii International Conference on System Sciences », January, 2003.

[10] P. COUDERC, M. BANÂTRE. *SPREADing the Web.* in « Proc. of the 8th International Conference on Personal Wireless Communications (PWC) », Venise, Italy, September, 2003.

[11] P. COUDERC, M. BECUS. *Fast context discovery using Bluetooth.* in « Proc. of 12th IST Summit on Mobile and Wireless Communications », pages 215–220, Aveiro, Portugal, June, 2003.

[12] J. PAUTY, P. COUDERC, M. BANÂTRE. *Logical versus physical programming for ubiquitous applications.* in « Proc. of First Workshop on Intelligent Solutions in Embedded Systems (WISES) », pages 51–256, Vienna, Austria, June, 2003.

[13] I. PUAUT. *Méthodes de calcul de WCET (Worst-Case Execution Times) - Etat de l'art.* in « Actes de l'école d'été CNRS temps-réel », Toulouse, France, September, 2003.

[14] P. SINGH, G. CABILLIC. *A Checkpointing Algorithm for Mobile Computing Environment.* in « Personal Wireless Communication Conference », September, 2003.

[15] P. SINGH, G. CABILLIC. *Fault Tolerance and Availability in Mobile Computing Environment.* in « International Conference on Parallel and Distributed Processing Techniques and Applications », June, 2003.

[16] P. SINGH, G. CABILLIC. *Succesive Checkpointing Approach for Mobile Computing Environment.* in « International Conference on Wireless Networks », June, 2003.

[17] D. TOUZET, F. WEIS, M. BANÂTRE. *PERSEND: Enabling Continuous Queries in Proximate Environments.* in « Proc. of the Workshop on Mobile and Ubiquitous Data Access (WMUIA) », Udine, Italy, March, 2003, http://www.irisa.fr/aces/doc/ps03/wmuis.ps.gz.

Internal Reports

[18] J. PAUTY, G. CABILLIC. *Local Checkpointing for Embedded Java Applications.* Technical report, IRISA Number 1537, May, 2003.

[19] I. PUAUT, A. ARNAUD, D. DECOTIGNY. *Performance analysis of static cache locking in multitasking hard real-time systems.* Technical report, IRISA Number 1568, October, 2003.

[20] P. SINGH, G. CABILLIC. *Fault tolerance and mobile environment.* Technical report, IRISA, To Appear, 2003.

[21] A. TROËL, F. WEIS, M. BANÂTRE. *Découverte automatique entre terminaux mobiles communicants.* Technical report, IRISA Number 1570, October, 2003.

[22] A. TROËL, F. WEIS, M. BANÂTRE. *Prise en compte du mouvement dans les systèmes de communication sans-fil.* Technical report, IRISA Number 1508, January, 2003.

[23] A. TROËL, F. WEIS, M. BANÂTRE. *Représentation du voisinage physique dans les interactions de proximité.* Technical report, IRISA Number 1551, August, 2003.

Miscellaneous

[24] J. SEVIN, G. WATTS. *MMS demonstrator - Specification of the terminal architecture - Deliverable of the project Advanced service delivery in heterogeneous mobile network, INRIA/ALCATEL collaboration.* july, 2003.

Bibliography in notes

[25] L. R. C. ANF L. P. SCHULTZ, C. CONSEL, G. MULLER. *Java Bytecode Compression for Low-End Embedded Systems.* in « ACM Transactions on programming languages and systems », number 3, volume 22, May, 2000, pages 471–489.

[26] J. ARLAT, J. P. BLANQUART, A. COSTES, Y. CROUZET, Y. DESWARTE, J. C. FABRE, H. GUILLERMAIN, M. KAÂNICHE, K. KANOUN, J. C. LAPRIE, C. MAZET, D. POWELL, C. RABÉJAC, P. THÉVENOD. *Guide de la sûreté de fonctionnement.* Cépaduès, 1995.

[27] M. D. BENNETT, N. C. AUDSLEY. *Predictable and Efficient Virtual Addressing for Safety-Critical Real-Time Systems.* in « Proc. of the 13rd Euromicro Conference on Real-Time Systems », pages 183–190, Delft, The Netherlands, June, 2001.

[28] G. BLACK, P. TULLMANN, W. HSIEH, J. LEPREAU. *Techniques for the Design of Java Operating Systems.* in « Usenix Annual Technical Conference », June, 2000.

[29] G. C. BUTTAZZO, editor, *Hard real-time computing systems - predictable scheduling algorithms and applications.* Kluwer Academic Publishers, 1997.

[30] A. COLIN, I. PUAUT. *Worst Case Execution Time Analysis for a Processor with Branch Prediction.* in « Real-time systems journal », number 2-3, volume 18, May, 2000, pages 249–274.

[31] M. FLINN, M. SATYANARAYANAN. *Energy-Aware Adaptation for Mobile Application.* in « Proceedings of ACM SOSP », 1999, pages 48-63.

[32] M. S. JOHNSTONE, P. R. WILSON. *The Memory Fragmentation Problem: Solved ?.* in « Proc. of the 1rst international symposium on memory management », number 3, volume 34, pages 26–36, 1998.

[33] J. LORCH. *A complete picture of the energy consumption of a portable computer.* Masters thesis, Computer Science, University of California at Berkeley,, 1995.

[34] S. MICROSYSTEMS. *JSR-000118 Mobile Information Device Profile 2.0.* November, 2002.

[35] F. MUELLER. *Timing Analysis for Instruction Caches.* in « Real-time systems journal », number 2, volume 18, May, 2000, pages 217–247.

[36] B. NOBLE, M. SATYANARAYANAN, J. TILTON, J. FLINN, K. WALKER. *Agile application-aware adaptation for mobility.* in « Proceedings of the 16th Symposium on Operating Systems Principles », 1997.

[37] P. PILLAI, K. G. SHIN. *Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems.* in « Proceedings of the 18th ACM Symp. on Operating Systems Principles », 2001.

[38] I. PUAUT, D. DECOTIGNY. *Low-complexity algorithms for static cache locking in multitasking hard real-time systems.* in « Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS02) », pages 114-123, Austin, Texas, December, 2002.

[39] I. PUAUT. *Real Time Performance of Dynamic Memory Allocation Algorithms.* in « Proc. of the 14th Euromicro Conference on Real-Time Systems », Vienna, Austria, June, 2002.

[40] P. PUSCHNER, A. BURNS. *A Review of Worst-Case Execution-Time Analysis.* in « Real-time systems journal », number 2-3, volume 18, May, 2000, pages 115-128, Guest Editorial.

[41] P. PUSCHNER, A. V. SCHEDL. *Computing Maximum Task Execution Times – A Graph Based Approach.* in « Proc. of IEEE 1997 Real-Time Systems Symposium », volume 13, Kluwer Academic Publishers, pages 67–91, 1997.

[42] J. STANKOVIC, M. SPURI, M. D. NATALE, G. BUTTAZZO. *Implications of Classical Scheduling Results For Real-Time Systems.* in « IEEE Computer », number 6, volume 28, June, 1995, pages 16–25.

[43] J. STANKOVIC. *Strategic directions in real-time and embedded systems.* in « ACM Computing Surveys », number 4, volume 28, December, 1996, pages 751–763.

[44] V. TIWARI, S. MALIK, A. WOLFE. *Power Analysis of Embedded Software: A First Step Towards Software Power Minimization.* in « IEEE Trans. on Very Large Scale Integration Systems », 1994, pages 437-445.

[45] 3. G. P. - 3. T. 2. V5.4.0. *3GPP Multimedia Messaging Service - Functional Description.*

[46] M. WEISER, A. DEMERS, B. WELCH, S. SHENKAR. *Scheduling for reduced CPU energy.* in « Proceedings of Operating System Design and Implementation (OSDI), Monterey, CA », 1994.

[47] M. WEISER. *Some Computer Science Issues in Ubiquitous Computing.* in « Communication of the ACM », volume (7)36, 1993, pages 75-83.

[48] N. ZHANG, A. BURNS, M. NICHOLSON. *Pipelined Processors and Worst Case Execution Times.* in « Real-time systems journal », number 4, volume 5, October, 1993, pages 319–343.