INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team aces

Ambient Computing and Embedded Systems

Rennes

THEME COM

Table of contents

1. Team

Head of project-team
Michel Banâtre [Research Director]

Administrative assistant
Evelyne Livache

INRIA project staff
Paul Couderc [Research Scientist]

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

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

INSA de Rennes staff
Isabelle Puaut [MC (Assistant professor at INSA) up to 08/31/04]

Project technical staff
Mathieu Becus
Gregory Watts [up to 30/09/2004]
Cyril Ray
Julien Sevin

Ph. D. student
Alexis Arnaud [Inria grant, up to 30/09/2004]
Carole Bonan [Inria grant]
Julien Pauty [MESR grant]
Pushpendra Singh [Inria grant, up to 31/08/2004]
David Touzet [Inria grant, up to 30/04/2004]
Arnaud Troël [Inria grant, up to 30/04/2004]
Claude Vittoria [Inria grant]
Cedric Mosch [Inria grant]
Arnaud Guiton [Inria grant, since 01/10/2004]
Mazen Tlais [Inria grant, since 15/10/2004]

Other Personnel
Mathieu Avila [Junior technical staff, up to 30/09/2004]
Laurent David [Post-doctoral student, up to 08/31/2004]

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.

3. Scientific Foundations

3.1. Introduction

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

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*, available memory or communication bandwidth. Finally, embedded systems may have dependability requirements (see paragraph 3.2.2).

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* [22] 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 [21][14]. 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 [19]. 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 [16] obtained from the program source code, or control-flow graph [20], 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 [17] or pipelines [25]. 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. 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 [15]. 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" [24]. 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 3.3.3).

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 the captured information can be interpreted 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).

3.3.2. 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.

3.3.2.1. 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.

3.3.2.2. 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.

3.3.3. 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.

3.3.3.1. 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 [18], 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.

3.3.3.2. 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 pre-loaded 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

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

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

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

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 6.3 and 7.1.1).

4.3. Information systems for wireless appliances

Keywords: *communication protocol*, *physical proximity*, *spatial programming*, *wireless communications*.

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. Information systems based on spatial programming

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 5.5 and 6.5.2), 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. Heptane

Participants : *Mathieu Avila, Isabelle Puaut [contact].*

Status *: Open source software (GPL license) available at address http://www.irisa.fr/aces/software.*

The aim of HEPTANE is to produce 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 implements a so-called *"tree-based"* computation method, as well as an *IPET* computation method. The former 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. The latter generates linear contraints from the program assembly code, from which the program control flow graph is extracted, and then uses an ILP solver for the actual WCET computation.

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. In 2004, our research work on cache locking has been integrated into Heptane.

HEPTANE is designed to produce timing information for architectures Pentium1, Hitachi H8/300, strongARM, and MIPS (with an overly simplified timing model).

5.3. Artisst

Participants : *Isabelle Puaut [contact].*

Status *: Open source software (LGPL license) available at address http://www.irisa.fr/aces/software.*

Artisst (for Artisst is a Real-Time System Simulation Tool) is a libre (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 largely 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.

5.4. 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 and discontinuous networks (see section 4.3). During the this INRIA/ALCATEL collaboration we have studied and implemented a new messaging service (based on the MMS standard [23]) 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.

The Java MIDP 2.0 environment [26]was chosen to demonstrate an implementation of the principle, since its portability make it applicable to a wide range of devices. Connections to network services are typically made through a well-defined Java interface called HttpConnection, which takes a URL as a parameter. Normally for an URL such as Òhttp://path/to/resourceÓ, the protocol (in this case: http) is independent of the underlying network interface, and data is sent to the TCP/IP stack which uses the IP routing table to choose the appropriate network interface. This implementation adds functionality to the Java virtual machine such that the protocol

of the specified URL may explicitly choose the network interface. That is, an application may create an HttpConnection using the URL httpGRPS://path/to/resource, which will always make the network connection over the GPRS interface. Likewise, the same application may specify the httpWifi://path/to/resource URL, which would always make the connection over the WLAN interface. After an HttpConnection is created for the application, the underlying implementation is completely transparent to the mobile application. The network layer is responsible for assuring that the correct network interface is used for the protocol specified by the application. The network layer informs the mobile application of the available network interfaces so the application can choose intelligently between them, according to its own criteria. This is accomplished using two mechanisms: an application can choose to be notified of network interface changes using an internal datagram, or it can find the current network interface availability by querying MIDP properties. In addition, the network layer may be extended to call additional applications when a network interface becomes available. For example, this would permit a mobile phone to automatically launch an authentication client on entry into a WiFi hotspot.

This terminal and the associated messaging service have been integrated to the common INRIA/ALCATEL testbed platform.

5.5. Spread

Members: *Michel Banâtre [contact], Mathieu Bécus, Paul Couderc.*
Status: *Internal to the team.*
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 [6], an operational urban transportation assistant for disabled people.

6. New Results

6.1. Introduction

The ACES project is currently very active in three main research activities

- Worst-case execution time analysis for embedded systems
- Design of Java based operating system
- System support for ambient computing

In the following we give the major research results we got from these activities.

6.2. Embedded systems

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

Members: *Alexis Arnaud, Isabelle Puaut.*

Participants : *Alexis Arnaud, Laurent David, Mathieu Avila, 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 5.2) 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 (see paragraph 6.2.1). On the other hand, we worked on probabilistic WCET analysis (see paragraph 6.2.1) to obtain information on the *distribution* of the execution times of pieces of software instead of just an upper bound.

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.

In cache locking schemes, given a task, its code is subdivided into a number of regions. Each such region is associated a cache contents. Consequently, executions of the task are subdivided into temporal windows, in each of which the cache contents is static. We have shown in previous work [12] that using a single region lacks scalability with respect to the tasks memory requirements. Thus, our ongoing work (PhD thesis of Alexis Arnaud) focuses on the design of algorithms for (i) partitioning the task code into regions, and (ii) determining for every region a cache contents.

We have explored two types of methods. The first one is a greedy algorithm exploiting the task control flow graph of the task, as well as its execution profile (trace of memory references), while the second one uses genetic algorithms. The first performance results show that the measured performance of applications using a locked cache are comparable with the performance using a standard (dynamic) cache.

6.2.1.2. Probabilistic WCET 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 so 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 the input data or system state. 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.

Using static analysis, we have proposed in [7] a method to obtain the probabilistic distributions of execution times. It assumes that the given real time application is divided into multiple tasks, whose source code is known. Based only on this source code, the proposed technique allows designers to associate to each possible execution path a probability to go through this path and its resulting execution time, ignoring in this paper hardware considerations. A source code example is scrutinized to illustrate the method.

6.2.2. Performance evaluation of embedded systems

Members: *Laurent David, 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.

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

We have proposed 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.

6.2.2.2. 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. Java operating system design

Members: *Michel Banâtre, Arnaud Guitton, Cedric Mosth, Cyril Ray, Jean-Paul Routeau, Claude Vittoria*

6.3.1. Introduction

The main goal of our researches in Java operating system area is to design a Java operating system to run on a heterogeneous multiprocessor architecture. Some of the processor of this architecture are especially dedicated to a fast execution of "bytecode" programs which are possibly the results of Java program compilation. It is important to know that we don't want to rewrite a general purpose operating system in Java such as Linux or Symbian. Instead we want to design a new operating system written in Java which is going to provide the best implementation of Java features such as objects or thread on the top of the hardware architecture we contributed to the design last two years. In order to have a good start of this design work, we have initiated different research tasks we detailed now.

6.3.2. Background on the "Software" view of the hardware architecture.

The first point we consider is the hardware architecture. It is a multiprocessor architecture, built around an MPU, an ARM in our case, and one or more specialised processors, bytecode execution oriented, the JSM processors. JSM basic design principles followed some strong requirements: To run Java compliant programs,(according to SUN's references), without any requirements about a particular JVM or an operating system. To be very efficient: such a requirement is provided using hardcoded bytecodes and by the combination of a stack and a RISC/DSP architecture. To have a low power consumption as embedded and mobile phone will be the target device for the expected applications. To be low cost, which implies a simple design.

Without going into a detailed presentation of the processor design, some strong points are important to highlight some points which have a direct impact on the operating system design. The first one we consider is the data flow execution of a "bytecode" program, which may be the result of a Java compilation. In that case two kind of instruction set can be executed, on one hand "bytecode" instructions directly implemented in the hardware, which is the semantic defined by SUN, on the other hand, sequences built from a complementary instruction set the C-ISA (Complementary Instruction Set Architecture). sequences are used to build bytecodes too complex to be implemented directly in hardware. For efficiency reasons, the C-ISA also supports DSP instructions.

In order to take into account that the JVM is a stack machine, the JSM processor supports a hardware memory stack management. A cache associated to this stack ensures an efficient access to the top of the stack.

The whole multiprocessor architecture is a shared memory one, which leads to the well-known problems of bottleneck when the JSM and the MPU are sharing data. To avoid such situations, the JSM has its own local caches with a write-back strategy to update the shared memory. The cache design takes into account low power memory consumption requirements. The MPU is mainly dedicated to low priority tasks such as garbage collector, some interrupt management, compilation for optimisation reasons.

6.3.3. The notion of "bytecode" machine.

The notion of "machine language" is not new, many attempts to build such architectures have been done in the years 70, without getting the planned business success. We can name the Burroughs machine designed to run Algol-60 programs or the SAR designed to support Algol-68. Other researches have provided nice results about the data type implementation and the protection mechanisms on programming languages such as Hydra from CMU or the iAPX432 from Intel.

Behind all these research activities, the goal is always the same: to run a programming language directly on an hardware architecture. Such an approach provides obvious benefits due to the powerful programming

language semantics directly provided by the hardware. However these solutions have some drawbacks due to the programming of communication facilities between programs and the hardware, an other problem is related to efficiency as it is difficult to build complete hardware architecture to directly run programs, in that case the architecture is emulated. The last reason explaining the lack of success of these experiences is the choice of the programming language, which is not very suitable to run popular applications

Our challenge is to demonstrate that this "machine language" approach based on Java is becoming realistic. Two main reasons to support this fact: first Java is a secure object oriented programming language well suited to a many classes of applications. The portability is ensure by the existence of a JVM used to run Java programs. Second, the availability, in a near future, of the JSM processor designed inside a partnership between Texas-Instruments and the ACES/INRIA team. This processor runs bytecode in a very efficient way. Today our main goal is to design the operating system, written in Java, to support the running of Java applications. For sake of simplicity we assume the availability of a Java compiler to produce byte code for the JSM, then we speak of "byte-code" machine instead of "machine language".

6.3.4. Notion of primitive objects.

Our "bytecode machine" has to support mainly programs written in Java. To do so, for example we need to get a memory management unit well adapted to Java objects and a support for thread and their related synchronisation rules. We need also handlers to manage devices such as screen, keyboards, communication boards, (WiFi, Bluetooth,...). All this material is part of our "bytecode" machine operating system. The crucial question we have is about the concrete implementation of these mechanisms. A first approach would be to start from a classical system such as Linux written in the C-ISA language and to port it on the JSM. Another one, which is our solution, is to re-write from scratch the operating system in Java, with such a approach the operating system take the benefit of Java in large. But the main difficulty remains the access to the hardware functionalities from Java programs.

Our solution to this problem lies on the notion of primitive object. A primitive object is characterized by its own API, which can be used in Java programs as ordinary Java objects. But the associated methods are directly built in C-ISA to access hardware resources. Obviously, these methods don’t define the complete management of the resources, the other part is done using a set of Java objects which interact with the primitive objects to access to the hardware. The hard problem is to identify the minimal set of primitive objects associated to each resource and to build them. Our solution is different from the micro-kernels one, they propose a set of minimal concept (task, memory message,...) to support operating systems based on the client-server paradigm. Our goal to support only Java program execution and their possible interaction with the hardware, it is the purpose of the notion of primitive objects.

6.3.5. Current works.

Today we are mainly involved to the design of this operating system. Our first proposals about primitive objects are related to interrupt management and process scheduling. Obviously other operating systems functionalities have to be considered. Two PhD students have just started. One considers the java memory management and its relation to the virtual memory management, the other is devoted to the thread management. For these two activities the muti-processors and multi-Java applications will be considered.

Even if the JSM is not yet available, we already consider the experimentation phase. An experimentation platform is under construction on the top of Linux. Currently we are implementing early versions of thread management, and interrupt management with mono-application, multi-thread and mono-processor assumptions.

To summarize this section related to our research on Java operating system, we think to have put in advance the majority of the problems to provide a Java operating system for Java programs, such programs are running onto a "bytecode" machine. Obviously such a system sill need future (and hard) works in order to consider problems related to multi-applications, the persistency of Java object and running of concurrent JVM on the same hardware architecture. Publications on this work are under preparation.

6.4. Fault-tolerant environment for mobile devices

Members: *Pushpendra Sigh*

Wireless technology provides us the opportunity of being connected, while on move. As the technology advances, portable computing devices are entering into our life in the form of cell phones, PDA (Personal Digital Assistant) etc. The trend shows that soon they will be dominating mobile computing environment. However because of lack of resources and high mobility, applications running on mobile devices suffer from faults and need fault tolerance mechanisms to guarantee their performance. In the future world, availability will become a very important criterion for user. Thus there is a need to have a composite fault tolerance policy aimed for such devices. Such policy should be adaptable to the user's needs and should be able to choose appropriate fault tolerance mechanism at a given time without user intervention.

Our research is aimed at developing such a fault tolerance policy for embedded mobile devices where we can determine suitability of a fault tolerance mechanism for the application with respect to the needs of user and available resources. More precisely, we are developing methods to define requirements of applications and services offered by a particular fault tolerance mechanism in order to find out compatibility between the two at any given time and choose the most suitable one.

Another part of out work has been to develop two distributed coordinated checkpointing algorithms, one for deterministic applications and another for non-deterministic applications, and a game application for mobile devices to examine our thesis. The practical implementation of our work has been done on WTK 2.0 with J2ME-MIDP. It has been divided into two parts. In the first part, we provide a simulation environment with imaginary applications and mechanisms to show working of our scheme. In the second part, we do an actual implementation of our work with our checkpointing mechanisms and game application. We have shown how to profile applications and mechanisms and how it can be integrated in our work successfully.

6.5. System supports for ambient computing

6.5.1. 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. Global coverage will be provided by 2G/2.5G cellular systems. 3G systems (UTMS) will be deployed in limited areas with strong densities of population. As a result, high data rate (several Mb/s) will not be available everywhere, and the delivery of large amount of information to people on the move will remain limited and expensive. In this study, we have designed a new network architecture to address this problem. This architecture is based on the deployment of high data rate Pico cells. The coverage is not continuous, so the cost of this infrastructure is many times less than that of a cellular system. The main challenge is to implement continuous data access for mobile users, in spite of the discontinuous coverage offered by this architecture. We strong believe that efficient solutions can be provided, using data caching techniques and mobility management mechanisms (for ex. user move anticipation).

The objective here is to propose solutions for continuous information delivery in spite of the discontinuous coverage. Our first contribution is an architecture model for representing the different components of a mobile heterogeneous network (mobile terminals, access controllers, legacy network ...). We mainly focus on the problems of resource sharing and the use of caching and prefetching mechanisms.

Our model is 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. To cope with the discontinuous coverage of the network, we store data (with caching mechanisms) close to mobile users, just before data delivery. So the placement policy of data within the architecture is conditioned by the knowledge of users mobility. The goal here was to define a representation of the users mobility in the network architecture, and to use this model for placing data using limited and oriented flooding mechanisms.

Through this architecture model, we made an analogy between heterogeneous mobile networks and multiprocessor architectures (for example the mobile terminal can be considered as a processor). This approach allows to map and extend existing caching mechanisms taking into account the specific constraints of a discontinuous mobile network. This architecture and the attached mechanisms have been evaluated with a simulation platform.

To support anytime and anywhere data access in such an heterogenous and discontinuous architecture, future mobile terminals will have multiple network interfaces, and the most suitable interface will be chosen by taking user's context into consideration. This point has been studied (see section 5.4, and [10]).

6.5.2. 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, two importants improvement have been made. The first one is the addition of geometrical addressing for tuples operations. With these extensions, each tuple can be associated to a specific geometrical shape around the physical object hosting the process which publishes the tuple. When a process needs to read data, it can specify a shape for the read operation. Only tuples which shapes intersect the shape of the read operation are matched against the read pattern. This improvement extends the expressivity of the spatial programming model. In particular, the model can now address some interesting application cases of spatial programming, which were not possible with the previous version. An implemententation of these extensions was done for the SPREAD system. Currently, we use the global positionning system for geometrical sensing, but ideally the system would make better use of local positionning technologies, such as ultra wide band communications.

Another new result in the context of the SPREAD system is the support of atomic destructive read operation, called $take$, similar to the in operation supported in traditionnal tuple space such as Linda. Such operation allows a mobile device a to read a tuple t published by another mobile device b and removing it from b atomically. This semantic is very useful for applications built on the spatial programming model. However, such semantic cannot be guaranteed in the context of mobile devices using limited range communication. The problem is equivalent to an atomic token passing between two mobile devices that has to be commited in a limited time (because of device mobility and limited communication range). Unfortunately, it is well known that atomic commitment requires a potentially infinite number of messages over an unreliable communication medium, before eventually committing. As the general problem is not solvable, we proposed a two approaches which work under relaxed constraints.

The first one relaxes the guarantee of atomic commitment: the protocol is allowed to fail in a marginal proportion of cases. To reduce the failure rate, we propose a geometrical constraint by limiting the distance d of peer devices with which a node can start the protocol. This idea is to preserve enough time to complete the commit phase. Simulations of an urban scenario shows that, considering a set of Wifi devices moving at speed ranging from 0 to 50km/h, allowing $take$ operation only with peers distant of $d < 50m$ achieves 100% success rate. Errors start to occur at $d > 50m$.

In case where a strict guarantee is required for the atomic committment, we propose to relax the constraint on limited communication range: this mean that communication range can be extended in order to extend the communication time required to complete the commit phase. An example of implementation of this solution would be to rely on a global cellular network in the case failures of the commit phase using the short distance interface (which can be limited, as shown with the first solution). Of course this solution would involve a higher cost than the first one (no guarantee) in terms of both energy and network usage.

This work has been submitted for publication.

Finally, we introduced an improved communication architecture in the context of Bluetooth [13]. Bluetooth is a becoming a common communication interface for communication between personal devices that are close together. However, its standard architecture raises strong limitations, as devices must discover peer devices

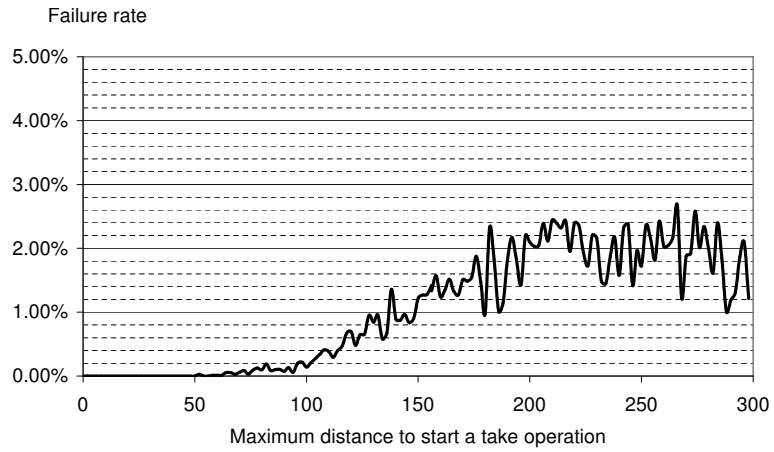

Figure 1. take failure rate vs. initiate distance threshold

using a time consuming discovery procedure, and cannot communicate while engaged in the discovery mechanism. Hence, *spontaneous communications*, such as required in the SPREAD system, are very challenging to achieve given bluetooth limitations. Our solution to overcome this problem problem involves a multi-unit bluetooth architecture, which significatively improves reactivity (figure 2). The idea is to dedicate one unit to the discovery procedure, while handling communications with another (or eventually several other) one(s).

Regarding applications, two applications were developped. The first one, Ubi-Q, consists of exploiting wireless communication for service preprocessing. The second one, the physical media space, is a collaborative context-aware system whose development is just starting.

Ubi-Q shows 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. This application also shows an original way of using close interactions (in the physical space) in order to efficiently allocate ressources on a system involving exclusive access in part of the processing. This principle finds a real world application in fast distributions services, such as automatic telling machine or video renting machine for example.

The physical media space is a context-aware media system, similar to the concepts introduced a previous application called *WebWalker*. The idea of the system is to allow *implicit structuring* and *adressing* of digital media collection (captured from the physical space) using contextual properties, such as time and space. Concept of context-aware information system is not new, and numerous systems have been describes in the litterature. However, in these system the content is not created implicitely as the data are captured (ie, as a new picture is shot). In the physical media space, we propose to allows such dynamic content contribution by structuring automatically the information system in the physical space. Several issues in this system require further research. In particular, a data storage architecture and a communication architecture that would efficiently support contextual data access and indexing are to be investigated.

7. Contracts and Grants with Industry

7.1. National contracts

7.1.1. Texas Instruments

- Number: 198C2730031303202

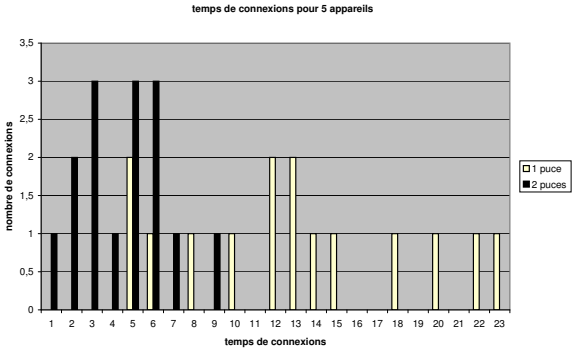

Figure 2. Connection times: standard bluetooth vs. dual unit

- Title: Real time Java Distributed Processing Environment
- Related Research activity: see section 6.3
- 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: Advanced service delivery for 4G discontinuous networks
- Related Research activity: see section 6.5.1
- Partner: *Alcatel*
- Founding: *Alcatel*
- Starting: 01/09/2002, ending : 30/10/2004
- Extension starting: 01/11/2004, ending : 31/10/2005

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.

8.1.2. Coordinated Action: Embedded WiSeNts

- Title: Cooperating Embedded Systems for Exploration and Control featuring Wireless Sensor Networks
- Partners: Technische Universität Berlin (Germany), University of Cambridge (UK), University of Copenhagen (Denmark), Swedish Insitute of Computer Science (Sweden), University Twente (Netherlands), Yeditepe University (Turkey), Consorzio Interuniversitario Nazionale per l'Informatica (Italy), University of Padua (Italy), Swiss Federal Institute of Technology Zurich (Switzerland), Asociación de Investigación y CooperaciónIndustrial de Andalucía (Spain), Institut National de Recherche en Informatiqueet en Automatique (INRIA), Universität Stuttgart (Germany)
- Starting: September 2004, ending: August 2006.

Embedded WiSeNts aims to increase the awareness and to find out a vision as well as a researh roadmap towards wireless sensors networks of cooperating embedded systems, within the academic community and, most importantly, within the manufacturers of proper technologies as well as potential users community.

8.2. French initiative for research in security and informatics

8.2.1. ACI: Mosaic

- Title: Mobile System Availability Integrity and Confidentiality
- Partners: Institut Eurecom, Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA), Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS)
- Starting: September 2004, to August 2007

The MoSAIC project is studying new fault tolerance and security mechanisms for mobile wireless devices in ambiant intelligence applications. We focus on sparse self-organized networks, using mostly one-hop wireless communication.

9. Dissemination

9.1. Animation of the scientific community

9.1.1. Program committees

- PC member of the first International workshop on systems and networking for smart objects (SANSO 2005), July 2005 (M. Banâtre)
- PC member of Second European Workshop on Wireless Sensor Networks (EWSN 2005), February (M. Banâtre)
- PC member of LCTES2004 (ACM SIGPLAN 2004 Conference on Languages, Compilers, and Tools for Embedded Systems), 2004 (I. Puaut)
- PC member of ECRTS 2004 (*16th Euromicro Conference on Real-Time Systems*), july 2004, and ECRTS 2005, july 2005 (I. Puaut)
- PC chair of the 4th French conference on operating systems (CFSE'4), Le Croisic, France, april 2005 (I. Puaut)
- PC member of the French conference on real-time systems (RTS 05) (I. Puaut)
- PC member of the International Workshop on Wireless Ad Hoc Networking (WWAN 2005), June 2005 (F. Weis)
- PC member of the 2nd French conference "Journées Francophones Mobilité et Ubiquité", (Ubimob'05) June 2005 (F. Weis)
- PC member of Histoire de l'informatique et des Télécommunication, Novembre 2004 (M. Banâtre)

9.1.2. Organizing and reviewing activities

- Organizer of the 3rd workshop on worst case execution time analysis, Catania, Italy, july 2004 (I. Puaut).
- Co-organisation thematic day “systèmes temps-réel” in the framework of the french chapter of ACM-Sigops, november 2004 (I. Puaut, with Y. Trinquet and S. Faucou, IRRCCyN)
- Michel Banâtre is member of the scientific committee of the encyclopedie des systèmes d’information (Vuibert) and scientific leader of the section related to Architectures and Operating Systems

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.

9.2. National and international working groups

- Participation (as an affiliated member) to the Compiler and Timing Analysis Working Group of Artist2 (network of excellence) (I. Puaut)
- Participation to the CNRS AS (Actions Spécifiques)
- Participation to the CNRS "Actions Spécifiques"
  - adéquation architecture-systèmes d’exploitation pour le temps-réel (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),
- Ecole des Mines de Nantes
  - Responsability of the lecture on distributed systems (final year of masters) computer science department (M. Banâtre),
- 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).
- ENST Bretagne, Lecture on Wireless LANs (final years of masters) (F. Weis).
- ENSEIRB (Bordeaux), Conference on Mobile communications and ambient computing, final year of masters, November 2004 (M. Banâtre).
- INSA de Lyon, Lecture on Ambient Computing, February 2004 (M. Banâtre, P. Couderc)

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).

I. Puaut is responsible of the 4rth year in computer science of INSA de Rennes.

F. Weis participates to the computer science *commissions de spécialistes* of universities and schools: Université de Lille 1.

9.4. Internship supervision

We have supervised the following internships in 2004:

- Anne-Sophie Adde (masters student, University of Rennes)
- Guillaume Membre (masters student, University of Nantes)
- Xavier Le Bourdon (INSA Rennes)
- Richard Turmel (INSA Rennes)
- Abdelkrim Khartoch (final interniship, ENSAO, Oujda, Morocco), on the design of a cache locking scheme using genetic algorithms, february-june 2004.
- Gael Legargeant (undergraduate student, INSA of Rennes), on the performance evaluation of genetic algorithms for cache contents selection, june-august 2004.
- Antoine Luu (Engineer student, ENSEIRB Bordeau)
- Arnaud Guitton (Master and Engineer student, ENSEIRB Bordeau)

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:

- Journées d'Etudes SSE, Très courte ou très longue portée, quelle place pour les communications radio, CNAM, Paris, Juin 2004 (P. Couderc)
- Systèmes d'Information Spontanés, Journée de l'action spécifique CNRS-GET, Systèmes répartis et réseaux adaptatifs au contexte ("Context-Aware"), AS 150, CNAM, Paris, Avril 2004 (F. Weis)
- Ubiquité numérique : principes, architectures et applications, Sixième Ecole d'Hiver des Télécommunications (EcoTel 2004), Réseaux sans fils, Zarzis, Tunisie, Décembre 2004 (F. Weis)
- *Méthodes de verrouillage de caches dans les systèmes temps-réel strict multi-tâches*, Action Spécifique (AS) CNRS adéquation architecture-systèmes d'exploitation pour le temps-réel, Paris, february 2004 (Isabelle Puaut and Alexis Arnaud)
- *WCET analysis and memory hierarchies*, RIS working group on "nouvelles architectures et technologies de processeurs et sûreté de fonctionnement", LAAS, Toulouse, september 2004 (I. Puaut)

9.6. Thesis committees

- Xavier Vera, Malardälen university, Vasteräs, Sweden, january 2004, *Cache and compiler interaction: how to analyze, optimize and time cache behavior*, as an *opponent* (I. Puaut).
- Olivier Charra, Université Joseph Fourier, Grenoble), may 2004, *Conception de noyaux de systèmes embarqués reconfigurables*, "rapporteur" (I. Puaut).
- Jerome Legrand, Université de Bretagne Occidentale, Brest, december 2004, *Contribution à l'ordonnancement des systèmes temps réel comprenant des tampons*, "rapporteur" (I. Puaut).
- Mikaël Briday, Université de Nantes, décembre 2004, *Validation par simulation fine d'une architecture opérationnelle*, "rapporteur" (I. Puaut).
- Guillaume Lussier, INSA of Toulouse, september 2004, *Test guidé par la preuve - Application à la vérification d'algorithmes de tolérance aux fautes*, "rapporteur" (I. Puaut).
- Hamoudi Kalla, INPG, Grenoble, décembre 2004, *Génération automatique de distributions/ordonnancements temps-réel, fiables et tolérants aux fautes*, examinateur (I. Puaut).

9.7. Reviewing activities

- External reviewer for a 4-year long research project *Live Updates for Embedded Systems* submitted to the Fund for scientific research of Flanders, Belgium, march 2004 (I. Puaut)

9.8. Patents

- R. Skabra, G. Watts, F. Weis, M. Banâtre « *Dynamic Network Interface Selection in Heterogeneous Networks* » European patent application request filed on July 1st 2004, number: 04 291 679.1
- M. Banâtre, P. Couderc, M. Becus, « *Dispositif transactionnel à prétraitement.* » European patent application request file on 15th of January 2004, number: 0400352
- M. Banâtre, P. Couderc, (Film by C. Blonz) « *Ubi-Q.* » An INRIA film about Ubi-Q (2004).

9.9. Industrial transfers

This year, our main industrial transfer is about the "dynamic network interface selection in heterogeneous networks" (patent on software). This work has been transfered to Alcatel. At the same time, one of the engineer (G. Watts) who evolved in this activity has just joined Alcatel R&D.

10. Bibliography

Doctoral dissertations and Habilitation theses

[1] P. SINGH. *Environement de tolérance aux fautes pour terminaux mobiles*, Ph. D. Thesis, Université de Rennes I, November 2004.

[2] D. TOUZET. *Interrogation continue des systèmes d'information de proximité*, Ph. D. Thesis, Université de Rennes I, March 2004.

[3] A. TROËL. *Prise en compte de la mobilité dans les interactions de proximité entre terminaux à profils hétérogènes*, Ph. D. Thesis, Université de Rennes I, March 2004.

Articles in referred journals and book chapters

[4] J. PAUTY, P. COUDERC, M. BANÂTRE. *Architectures de systèmes pour l'informatique diffuse*, in "TSI", to appear, 2005.

[5] D. TOUZET, F. WEIS, M. BANÂTRE. *Architectures pour l'ubiquité numériques*, in "Techniques et Sciences Informatiques", vol. 23, n° 4, 2004, p. 439–478.

Publications in Conferences and Workshops

[6] M. BANÂTRE, P. COUDERC, J. PAUTY, M. BECUS. *Ubibus: Ubiquitous Computing to Help Blind People in Public Transport.*, in "MobileHCI04", september 2004, p. 310-314.

[7] L. DAVID, I. PUAUT. *Static Determination of Probabilistic Execution Times*, in "Proc. of the 16th Euromicro Conference on Real-Time Systems, Catania, Sicily, Italy", June 2004, p. 223-230.

[8] M. KILLIJIAN, D. POWELL, M. BANÂTRE, P. COUDERC, Y. ROUDIER. *Collaborative Backup for Dependable Mobile Applications*, in "Proc. of 2nd International Workshop on Middleware for Pervasive and Ad-Hoc Computing, Middleware 2004, Toronto, Ontario, Canada", October 2004.

[9] G. LAPALME, M. BANÂTRE, P. COUDERC. *Audioguide dynamique*, in "Journée ATALA "La Génération de LN", ENST Paris", December 2004.

[10] H. MAILLARD, C. BAZIN, G. JAUPITRE, R. SKRABA, F. WEIS. *Enhanced Multimedia Messaging Service (MMS) delivery over heterogeneous networks*, in "Proc. of the 9th International Conference on Intelligence in service delivery Networks (ICIN'2004), Bordeaux, France", October 2004.

[11] J. PAUTY, G. CABILLIC. *A Checkpoints Mechanism for MobileJava Applications*, in "Proc. of the Parallel and Distributed Computing and Networks (PDCN'04), Innsbruck, Austria", February 2004.

[12] I. PUAUT, A. ARNAUD, D. DECOTIGNY. *Analyse de performance de méthodes de verrouillage statique de caches dans les systèmes temps-réel strict*, in "Proc. of the 12th International Conference on Real-Time Systems (RTS'04)", Mars 2004.

Internal Reports

[13] P. COUDERC, M. BECUS. *An architecture for Fast bluetooth connections in the context of spontaneous communication*, to appear, Technical report, INRIA, 2004.

Bibliography in notes

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

[15] 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.

[16] A. COLIN, I. PUAUT. *Worst Case Execution Time Analysis for a Processor with Branch Prediction*, in "Real-time systems journal", vol. 18, n° 2-3, May 2000, p. 249–274.

[17] F. MUELLER. *Timing Analysis for Instruction Caches*, in "Real-time systems journal", vol. 18, n° 2, May 2000, p. 217–247.

[18] 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.

[19] P. PUSCHNER, A. BURNS. *A Review of Worst-Case Execution-Time Analysis*, in "Real-time systems journal", Guest Editorial, vol. 18, n° 2-3, May 2000, p. 115-128.

[20] P. PUSCHNER, A. V. SCHEDL. *Computing Maximum Task Execution Times – A Graph Based Approach*, in "Proc. of IEEE 1997 Real-Time Systems Symposium", vol. 13, Kluwer Academic Publishers, 1997, p. 67–91.

[21] J. STANKOVIC, M. SPURI, M. D. NATALE, G. BUTTAZZO. *Implications of Classical Scheduling Results For Real-Time Systems*, in "IEEE Computer", vol. 28, n° 6, June 1995, p. 16–25.

[22] J. STANKOVIC. *Strategic directions in real-time and embedded systems*, in "ACM Computing Surveys", vol. 28, n° 4, December 1996, p. 751–763.

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

[24] M. WEISER. *Some Computer Science Issues in Ubiquitous Computing*, in "Communication of the ACM", vol. (7)36, 1993, p. 75-83.

[25] N. ZHANG, A. BURNS, M. NICHOLSON. *Pipelined Processors and Worst Case Execution Times*, in "Real-time systems journal", vol. 5, n° 4, October 1993, p. 319–343.

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