INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team aces

Ambient Computing and Embedded Systems

Rennes

THEME COM

Activity Report

2005

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]

Project technical staff
Fabien Allard [since 10/10/2005]
Mathieu Becus
Antoine Luu
Cyril Ray [up to 30/09/2005]
Julien Sevin [up to 15/06/2005]

Ph. D. student
Carole Bonan [Inria grant, up to 31/12/2005]
Damien Martin-Gutteriez [ENS grant, since 01/09/2005]
Mickaël LeBaillif [Inria grant, since 10/10/2005]
Xavier Le Bourdon [MESR grant, since 01/10/2005]
Julien Pauty [MESR grant, up to 30/09/2005]
Claude Vittoria [Inria grant]
Cedric Mosch [Inria grant]
Arnaud Guiton [Inria grant]
Mazen Tlais [Inria grant]

2. Overall Objectives

2.1. Overall Objectives

Three key phenomena have been changing the nature of computing over the last few years. The first is the popularity of portable devices such as mobile telephones and Personal Digital Assistants (PDAs). More than 50% of the French population possess their own mobile phone and there is a large variety of smartphones on the market that integrate PDA functionality. The second phenomenon is the large number of *embedded systems*; these are everyday devices that have their own processor and memory. Estimates suggest that more than 98% of the world's processor's are in embedded systems [14], thus facilitating the deployment of a variety of information systems that control physical objects. The third phenomena is the increasing variety of wireless networks deployed in personal and embedded devices, e.g., Bluetooth, Wifi, GPRS, etc.

The combination of these three phenomena has permitted the emergence of environment-aware **person-centric** applications and services. These services complement a person's physical ability to interact with his environment. They are tailored to a the needs, preferences and location of each person carrying a device, and are continually available. Services range from critical, e.g., remote health monitoring [17], to utility, e.g., navigational help, etc. to value-added, e.g., virtual museum guides, smart home, etc.

The domain of person-centric computing is known in research circles as *ambient computing* [20]. The domain poses significant research challenges. First, to facilitate mobility, ambient computing services should require minimal device manipulation by the device owner. It is crucial that the device operate as an extension of the person rather than as a tool. Second, there must be a way of modelling the physical environment so that applications can seamlessly import data from the environment and modify the environment when possible. Third, applications must be able to adapt to the rather limited storage and processing capabilities of mobile devices, as well as to variable and intermittent wireless network coverage.

The ACES (Ambient Computing and Embedded Systems) group is addressing research from two angles:

- *Programming Models*: we are studying ways of modelling the physical environment in the virtual environment of programs in order to facilitate ambient application development.
- *Operating System Support*: we are looking at portable operating system designs for personal devices that manage limited physical resources and irregular network connectivity.

Throughout 2005, in accordance with the goals of the institute, the ACES group spent a great deal of time and effort on seeking to transfer its research results on ambient computing to potential users. We sought out and negotiated with representatives of end-users in advertising and the urban construction industries. Our motivation stems from our observation that producing innovative research results, even those protected by patents, is no longer sufficient for a modern research team. It is essential to convince industry that solutions are robust, scalable and most importantly, address a problem that real users are faced with. This research approach necessitates the development of several prototypes that are tested in real environments. It also necessitates a continuous technology watch to ensure the validity of submitted patents, as well as verification of existing patents and research reports. Although this activity is, traditionally, unusual for a research team, it becomes inevitable if results are to have a real impact in the ambient computing applications currently being deployed. Our technology transfer efforts have been successful and, within a short time, we expect to sign a major contract with one of the main actors in the advertising board industry.

This document overviews our activities in more detail. The section *Scientific Foundations* gives some background to our work in person-centric computing. The section *Application Domains* describes the importance of our research agenda through the presentation of several applications, some of which are being developed in our group. The group's recent results are presented in the section *New Results*.

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. Ambient computing and embedded systems are the foundations of person-centric computing. Our group is concentrating on *programming models* and *operating system support*; these are two essential and complementary aspects of ambient computing.

The purpose of a programming model is to represent information as data, and to provide a computational framework for data processing. The challenge for ambient and embedded computing is to seamlessly merge information from the physical and virtual worlds, so that programs can act upon and influence the physical world around them.

The goal of our research on operating system support is to define platforms that enable programs to run on resource-limited devices, where wireless network connection fluctuates. In particular, we are looking at environments built around Java^{TM} technology, in order to maximise application safety and portability.

3.2. Programming Models

The goal of ambient computing is to seamlessly merge virtual and real environments. A real environment is composed of objects from the physical world, e.g., people, places, machines. A virtual environment is any information system, e.g., the Web. The integration of these environments must permit people and their information systems to implicitly interact with their surrounding environment.

Ambient computing applications are able to evaluate the state of the real world through sensing technologies. This information can include the position of a person (caught with a localisation system like GPS), the weather (captured using specialised sensors), etc. Sensing technologies enable applications to automatically update digital information about events or entities in the physical world. Further, interfaces can be used to act on the physical world based on information processed in the digital environment. For example, the windows of a car can be automatically closed when it is raining.

This real-world and virtual-world integration must permit people to implicitly interact with their surrounding environment. This means that manual device manipulation must be minimal since this constrains person mobility. In any case, the relative small size of personal devices can make them awkward to manipulate. In the near future, interaction must be possible without people being aware of the presence of neighbouring processors.

3.2.1. Programming Context

Information systems require tools to *capture* data in its physical environment, and then to *interpret*, or process, this data. A context denotes all information that is pertinent to a person-centric application. There are three classes of context information:

- The *digital context* defines all parameters related to the hardware and software configuration of the device. Examples include the presence (or absence) of a network, the available bandwidth, the connected peripherals (printer, screen), storage capacity, CPU power, available executables, etc.
- The *personal context* defines all parameters related to the identity, preferences and location of the person who owns the device. This context is important for deciding the type of information that a personal device needs to acquire at any given moment.
- The *physical context* relates to the person's environment; this includes climatic condition, noise level, luminosity, as well as date and time.

All three forms of context are fundamental to person-centric computing. Consider for instance a virtual museum guide service that is offered via a PDA. Each visitor has his own PDA that permits him to receive and visualise information about surrounding artworks. In this application, the *pertinent* context of the person is made up of the artworks situated near the person, the artworks that interest him as well as the degree of specialisation of the information, i.e., if the person is an art expert, he will desire more detail than the occasional museum visitor.

There are two approaches to organising data in a real to virtual world mapping: a so-called *logical* approach and a *physical* approach. The logical approach is the traditional way, and involves storing all data relevant to the physical world on a service platform such as a centralised database. Context information is sent to a person in response to a request containing the person's location co-ordinates and preferences. In the example of the virtual museum guide, a person's device transmits its location to the server, which replies with descriptions of neighbouring artworks.

The main drawbacks of this approach are scalability and complexity. Scalability is a problem since we are evolving towards a world with billions of embedded devices; complexity is a problem since the majority of physical objects are unrelated, and no management body can cater for the integration of their data into a service platform. Further, the model of the physical world must be up to date, so the more dynamic a system is, the more updates are needed. The services platform quickly becomes a potential bottleneck if it must deliver services to all people.

The physical approach does not rely on a digital model of the physical world. The service is computed wherever the person is located. This is done by spreading data onto the devices in the physical environment; there are a sufficient number of embedded systems with wireless transceivers around to support this approach. Each device manages and stores the data of its associated object. In this way, data are physically linked to objects, and there is no need to update a positional database when physical objects move since the data *physically* moves with them.

With the physical approach, computations are done on the personal and available embedded devices. Devices interact when they are within communication range. The interactions constitute delivery of service to the person. Returning to the museum example, data is directly embedded in a painting's frame. When the visitor's guide meets (connects) to a painting's devices, it receives the information about the painting and displays it.

3.2.2. Spatial Information Systems

One of the major research efforts in ACES over the last few years has been the definition of the Spread programming model to cater for spacial context. The model is derived from the Linda [16] tuple-space model. Each information item is a *tuple*, which is a sequence of typed data items. For example, `<10, 'Peter', -3.14>` is a tuple where the first element is the integer 10, the second is the string '"Peter" and the third is the real value -3.14. Information is addressed using patterns that match one or a set of tuples present in the tuple-space. An example pattern that matches the previous tuple is `<int, 'Peter', float>`. The tuple-space model has the advantage of allowing devices that meet for the first time to exchange data since there is no notion of names or addresses.

Data items are not only addressed by their type, but also by the physical space in which they reside. The size of the space is determined by the strength of the radio signal of the device. The important difference between Spread and other tuple-space systems (e.g., Sun's JavaSpaces [15], IBM's T-Space [23]) is that when a program issues a matching request, only the tuples filling the *physical space* of the requesting program are tested for matching. Thus, though SIS applications are highly distributed by nature, they only rely on localised communications; they do not require access to a global communication infrastructure. Figure 1 shows an example of a physical tuple space, made of tuples arranged in the space and occupying different spaces.

As an example of the power of this model, consider two of the applications that we have developed using it.

- *Ubi-bus* is a spatial information application whose role is to help blind and partially blind people use public transport. When taking a bus, a blind person uses his PDA to signal his intention to a device embedded in the bus stop; this device then contacts the bus on the person's behalf. This application illustrates how data is distributed over the objects of the physical world, and generally, how devices complement human means of communication.
- *Ubi-board* is a spatial information application designed for public electronic billboards. Travel hotspots like airports and major train stations have an international customer base, so bill-board announcements need to be made in several languages. In Ubi-bus, a billboard has an embedded device. When a person comes within communication range of the billboard, his device sends a request to the billboard asking it to print the message in the language of the person. In the case where several travellers are in proximity of the billboard, the board sends a translation of its information message to each person. The Ubi-board application illustrates personal context in use, i.e., the choice of natural language, and also how actions can be provoked in the physical world without explicit intervention by the person.

3.3. Operating System Support

The role of an operating system is to offer an environment for program execution. It offers programs access to essential services, such as communication and storage, and thus indirectly to network and disk. Person-centric computing is characterised by small portable devices that can be awkward for a person to manipulate. The operating system needs to manage generally limited resource, in particularly in relation to the network. We first give some background to the problem of resource management – given its importance in person-centric computing – before coming back to other operating system aspects.

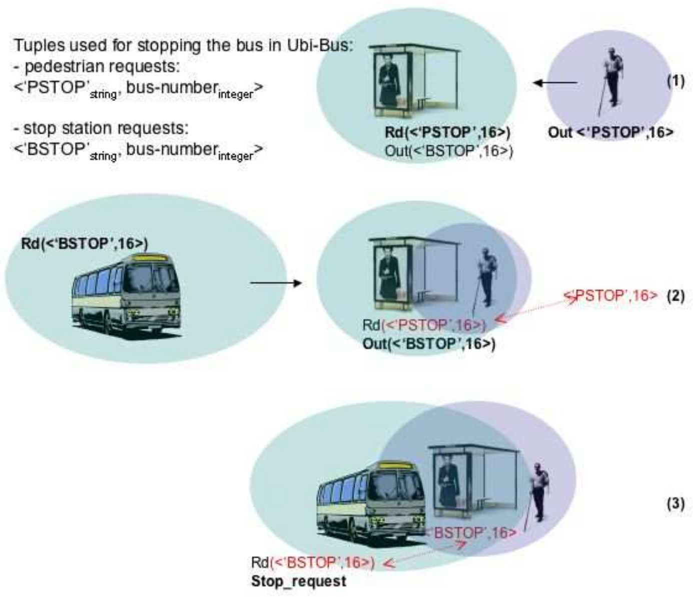

Figure 1. Physical Tuple Space

3.3.1. Service Adaptation

Mobile networks are becoming increasingly heterogeneous. Global coverage is now well provided by 2G and 2.5G cellular systems, and 3G networks (UTMS) are being deployed in some densely populated areas. Nonetheless, high data rates (several Mb/s) will not be available everywhere in the near future, so the delivery of large amounts of information to people on the move will remain limited and expensive.

3.3.1.1. Data Adaptation

Wireless network coverage in mobile systems can be highly variable for reasons of coverage as well as for reasons related to subscription business models. This variability suggests that an application must be able to adapt to network related changes. An application must also be able to adapt to particular resource limitations of devices it communicates with at any time.

Ultimately, adaptation manifests itself in the quality and nature of data exchanged between devices. One solution to adapting data is to transform its representation in a way that best suits the digital context of host devices.

For example, depending on the actual behaviour of the system (network bandwidth, load, etc.), it implies 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 the same picture). These representations differ only in quality. For instance, when transmitting video to a mobile device, the number of frames per second can be reduced to cater for variable bandwidth, and the size of each frame can be significantly reduced to cater for the fact that the screen is quite small. Another example is Web browsing; a mobile device might decide to filter applets and images from downloaded pages in order to reduce the device CPU time needed to process the page.

Data adaptability requires collaboration between the operating system and the applications [18]. As the previous examples illustrate, the type of data adaptation relies on the semantics of the application, so only it can decide how to adapt to resource changes. It is the role of the operating system to survey changes in resources, and to inform the application of important changes so that adaptation can take place in time.

An unstable connection can temporarily isolate a mobile entity from the rest of the system. When this happens, the proposed service cannot be properly delivered. In order to hide disconnections from a person, pre-fetching techniques can be used: documents are pre-loaded in a local cache of the mobile device during high connectivity periods. The application therefore remains operational when a temporal disconnection occurs as all needed documents have been stored locally in the cache of the device. The data loaded into the caches are selected based on the personal context, incorporating a person's preferences, as well as on hints provided by the user and on information that is automatically tracked (for example a log of all previously accessed documents).

3.3.1.2. The Pico-cell Architecture

The ACES group has designed a new network architecture to address the issue of data adaptation in mobile networks. This architecture is based on the deployment of high data rate *Pico cells*. The main challenge is to implement continuous data access for mobile people, in spite of the discontinuous coverage offered by the architecture. We strongly contend that efficient solutions can be provided, using data caching techniques and mobility management mechanisms that anticipate a person's move.

Our first contribution in this direction is an architecture model for representing the different components of a mobile heterogeneous network (mobile terminals, access controllers, legacy network, etc.). We mainly focus on the problems of resource sharing and the use of caching and prefetching mechanisms. The model is based on three communication links, c.f. Figure 2:

- The first link is provided by the legacy network, and is used by the mobile device 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 device.

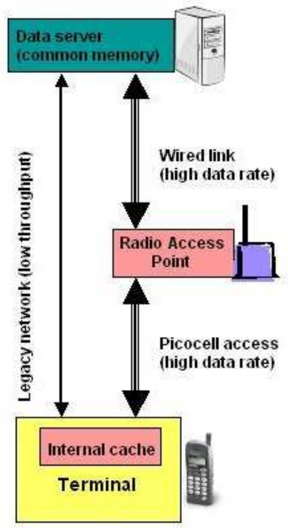

Figure 2. Pico-cell architecture model

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

Through this architecture model, we underline the analogy between heterogeneous mobile networks and multiprocessor architectures (for example the mobile device can be considered as a processor). This approach allows us 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 (See section on *Software*).

3.3.2. Operating Systems for Small Devices

Many of the requirements for operating systems in ambient computing and embedded system environments are the same as those for operating systems for desktop computers. For instance, an operating system must ensure that programs are protected from each other and that each obtains a fair share of platform resources. Another aim of operating systems is to facilitate application portability – the ability to execute an application on different hardware platforms without having to modify its code. One of the few new requirements for small device operating systems is the ability to fetch and load programs directly from the network, rather than just from disk, since devices may not have a disk integrated and/or may be managed remotely.

Recent years have seen a huge proliferation of applications developed in the Java^{TM} programming language [12]. The advantage of Java is that the language is strongly typed, so programs are protected from the memory manipulation errors that are frequent with C/C++ applications. Another reason for its success is that Java environments now possess a rich set of libraries that are convenient for a wide range of systems programming tasks. A final, and important, reason for the prevalence of Java is that the Java compiler transforms programs to a standardised set of high-level assembly-like instructions called *bytecode*. Java environments incorporate a virtual machine that interprets bytecode. The advantage of this is portability: any platform that runs a Java environment may run a compiled Java program, without modification. Java environments now exists for mobile phones, pagers, PDAs and a wide range of embedded and wireless devices[1].

3.3.2.1. Java Operating Systems.

The ACES research group has been collaborating with *Texas Instruments* on the design of a heterogeneous multiprocessor architecture composed of a dedicated Java bytecode processor (JSM) and ARM processor. This work led us to design an operating system that is completely written in the Java programming language, and that will run over the JSM.

The advantage of a Java operating system running on a JSM is portability and safety. Currently, Java environments permit applications to interact with non-Java code, which is generally referred to as *native* code. Native code is used to implement operating system services, drivers as well as legacy applications. The problem with native code is that it creates an application dependency on third-party non-Java code. This challenges application portability since an application on one platform is no longer guaranteed to run on another platform. Further, native code is not subject to the type-safety checks of the Java language; this code can therefore compromise the security of the whole application.

There have been several R & D efforts to develop Java operating systems and Java environments written in the Java language, e.g., JX [19], JNode[2], OVM[3], Jalapeño [11], etc. These systems still contain native code units. JNode and Jalapeño compile the system's code base to native code for a host platform processor and OS pair in an effort to maximise run-time performance. However, this approach means that no use can be made of an available JSM bytecode processor, and the system requires extra means to protect the Java environment from all imported native code.

The idea of a machine that directly implements a language is not new, with many attempts to build such architectures in the 1970s, e.g., the Burroughs machine [13] designed to run Algol-60 [21] and the SAR designed to support Algol-68. Other notable efforts include Hydra [22] from CMU and the iAPX432 from Intel [10]. The goal of these projects was to run a programming language directly on a hardware architecture. This approach provides the obvious benefits of having the programming language semantics directly implemented by the hardware. However, these systems had three important drawbacks. First, the implementation of communication channels to interface between the programming language and the underlying hardware was complicated. Second, performance was generally poor. Third, the programming languages chosen lacked the popularity needed for the machines to reach a critical mass. The Java language does not suffer from the popularity drawback, and the availability of JSM processors that execute bytecode is helping to address the performance issue. The remaining challenge is program-to-hardware communication, and is an issue being addressed in the ACES research group.

3.3.2.2. Native Objects and the Java Abstract Machine.

A portable Java operating system environment is one where all programs, services and operating system functions are compiled to bytecode. The only non-Java, or complementary instruction set architecture (C-ISA), instructions are those that manipulate attached hardware resources, e.g., keyboard, screen, communication ports, etc. The ACES group developed an operating system model in which devices are represented as Java objects whose methods can only be invoked by the operating system code (written in Java). These objects are known as *native objects*. Their methods are coded in C-ISA instructions, thus representing the underlying hardware in the Java world.

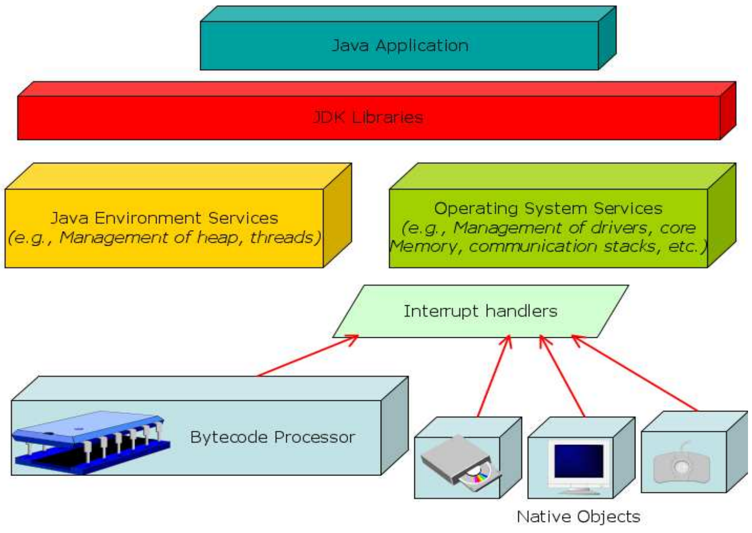
Java Application
JDK Libraries
Java Environment Services
(e.g., Management of heap, threads)
Operating System Services
(e.g., Management of drivers, core Memory, communication stacks, etc.)
Interrupt handlers
Bytecode Processor
Native Objects

Figure 3. Structure of Java-based Operating System

The structure of our Java operating system is illustrated in Figure 3. The native objects are placed at the same level as the bytecode processor. To date, we have developed native objects for a keyboard, video card and UART serial bus. As with any processor, the bytecode processor supports the deployment of interrupt handlers so that the operating system services can manage devices. The next layer is composed of two parts. A Java services part implements all functionality needed to implement Java programs that is is not provided by the bytecode; this includes garbage collection of unused objects, thread scheduling and code verification of classes loaded into the system. The operating system part contains the device drivers and other OS functionality written in Java; for instance, we ported a TCP/IP stack and LCUDI graphics library to Java for integration in this part. The remaining two layers are common to all Java environments. The presence of the Java development kit (JDK) means that Java applications can run unaware of the fact that a bytecode processor is running below them. In particular, user code and libraries cannot invoke methods on native objects.

Though our Java operating system is designed to run over a physical bytecode processor, it can just as easily run over a software emulation of this processor in native code. In this respect the bytecode processor is an example of an *abstract machine*. Implementing this abstract machine is simpler than implementing a traditional Java virtual machine since only the bytecode interpretation component needs to be implemented instead of a whole series of thread and memory management services. Further, this implementation is hidden from all application and system service developers who can in no way interfere with the correct functioning of the abstract machine, since their code is written in Java. Thus, a native implementation of the abstract machine does not suffer from the portability and security shortfalls that we mentioned above for existing Java environments.

We return to our work on Java operating systems in Section *New Results* where we detail the recent innovative elements of our work .

[1]http://java.sun.com/j2me/
[2]http://www.jnode.org
[3]http://www.ovmj.org

4. Software

4.1. 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.

4.2. Simulation Platform for Networks with Discontinuous Coverage

The ACES group developed a network simulator for pico-cell architectures that is used to analyse flow distribution in networks. This architecture is described in the *New Results* section of this report. The simulator is entirely coded in the Java programming language and uses the DESMO-J discrete event model. The simulator models all entities required to study discontinuous coverage network behaviour, including terminals, access points, admission control, content provision servers, MAC layers, wireless transmission, protocol cache management as well as different (person) device mobility models. The main emphasis of our development has been to tailor the simulator to measuring performance in large-scale networks – the size of towns where there are hundreds of devices per square kilometers – over periods of a few hours.

4.3. Ubi-Board

Ubi-Board is a splitted display system, capable of adapting contextually the content and the display surface accordingly to the population in the immediate vincinity of a display panel. The system uses spatial programming in order to determine dominant profiles, such as dominant language which allows the selection of the appropriate content to display on the main panel. As the main panel changes, the system synchronously pushes the appropriate content version to users not belonging to the dominant profile, to be displayed on their personal display (typically cellphones).

5. New Results

5.1. Introduction

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

- Programming models
- Discontinuous mobile networks
- Java Operating Systems

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

5.2. Programming models

Participants: Michel Banâtre, Mathieu Becus, Paul Couderc [contact], Damien Martin-Gutteriez, Mickael LeBaillif, Xavier Le Bourdon, Julien Pauty

The ambient computing activity for this year include three main topics, that we describe in the following.

5.2.1. Geometry and pervasive computing

The role of geometry is important in pervasive computing as we can derive implicit data organization from spatial properties, and implicit data processing from mobility. Spatial programming is a computation model using this approach, allowing simple programming of pervasive computing application distributed over a set of physical objects. A Ph. D was completed on this topic, and will be defended by Julien Pauty in January 2006.

A recent result of this work is a solution for the problem of atomic operation in the context of spontaneous communications, such as for example the atomic token passing between two mobile devices. This problem has no general solution, due to the inherently unreliable nature of wireless communication and the limited communication time implied by device mobility. However, partial solutions to the problem are proposed, involving a modified two phase commit protocol where engagement is restricted by a geometrical constraint. This constraint reduce the zone in which the protocol can start, in order to allow the completion before the devices be out of reach. We have shown that although it is not possible to guarantee failure free operation, acceptable reliability can be achieved for certain applications.

5.2.2. Context sensitive systems

The notion of context as it is used in most ubiquitous computing work is strongly different of the notion of context used in information system. Typically, context is understood by the ubiquitous computing community as a set of information charaterizing a physical situation. The notion of context in information systems corresponds to the relative location of a given information item, in respect to a more global set of information. We tried to provide a unified definition of context, based on the definition used in information system, but also covering physical aspects. In this definition, we consider the physical space as an implicit structure for information associated to physical objects or region. This definition provides a support for organizing and navigating an information collectionimplicitely from the physical space.

As an illustration, we developed a pervasive photo collection system, allowing in-context photo publication (as they are shot), as well as navigating context-relevant photo from the environment. This application was published in [3].

Regarding context-awareness, we were also strongly involved in a collaboration with an outdoor furniture industrial partner, to experiment real life application of our SPREAD platform and its applications, especially in the context of the bluetooth technology which raises important difficulties due to high latency discovery time. We enhanced our architectures, in particular by providing support for multiple bluetooth units in order to distribute discovery and connexion tasks. We are still investigating these issues, and other architectures to overcome them.

We also develop a splitted display system, Ubi Board, to demonstrate a useful application of dynamic context description by a set of physical objects. In this system, the mobile phone of each user self describes profile information (such as "language"). A smart public display panel then determines the "dominant" profile in its viciny, accordingly to the current majority. This is used to switch the content of the display in the appropriate language (of the majority), but at the same time sending localized version of the content to mobile of each user in a minority. This application is the object of a patent request.

5.2.3. Cooperating objects

The third topic is the cooperation of autonomous objects. With today proliferation of mobile devices enabled with communication and multimedia data capture capabilities, large quantity of data production and processing directly takes place on the move, or in situation. Heterogenous communication interface and node architecture, interactions with unknown nodes (hence, under limited trust), data reliability, global processing over a set of nodes are some of the challenging issues raised by cooperating objects. Wireless sensor networks have similar issues, in addition to exacerbated energy and resources constraints.

We were involved in the Embedded WiSents European cooperation [4], as a study leader regarding programming abstractions and system architecture for cooperating objects.

Key to the successful and widespread deployment of cooperating objects and sensor network technologies is the provision of appropriate programming abstractions and the establishment of efficient system architectures able to deal with the complexity of such systems. Programming abstractions shield the programmer from the “system specific details” and allow the developer to think in terms of the concrete application problem rather then in terms of the system. This is also true for traditional distributed systems, where numerous software frameworks and middleware architectures are crucial to perform an integrated computing task. Such

[4]http://www.embedded-wisents.org/

frameworks and middleware are based on programming models such as distributed objects or events. These conventional and successful distributed programming abstractions can, however, not be simply applied to cooperating objects or sensor networks, due to the differences existing among the latter and the former systems.

We are also involved in another cooperation, the french ACI Mosaic [5], which study the problem of cooperative backup of mobile devices. The first objective is to define an automatic data back-up and recovery service based on mutual cooperation between mobile devices with no prior trust relationships. Such a service aims to ensure continuous availability of critical data managed by mobile devices that are particularly prone to energy depletion, physical damage, loss or theft. The basic idea is to allow a mobile device to exploit accessible peer devices to manage backups of its critical data. The implementation of such a service by cooperation between devices with no prior trust relationship is far from trivial since new threats are introduced: (a) selfish devices may refuse to cooperate, (b) backup repository devices may themselves fail or attack the confidentiality or integrity of the backup data; (c) rogue devices may seek to deny service to peer devices by flooding them with fake backup requests; etc.

Dealing with these threats is the second objective of the project. We are studying mechanisms for managing trust in cooperative services between mutually suspicious devices. Of particular interest are mechanisms based on reputation (for a priori confidence-rating and a posteriori accountability) and rewards (for cooperation incitation). In the sparse ephemeral networks of devices considered, these mechanisms can rely neither on accessibility to trusted third parties nor on connectivity of a majority of the considered population of devices. Self-carried reputation and rewards are therefore of prime interest.

In this study, we are focused on the energy issues of the backup protocol, and on analysing the data production and use pattern of typical mobile applications.

5.3. Discontinous Mobile Networks

Participants: Carole Bonan, Antoine Luu, Julien Sevin, Mazen Tlais, Frédéric Weis [contact]

During this past year, we designed an architecture capable of supporting video streaming over a pico-cell type infrastructure. The core idea is to insert a new component, called an *access controller*, that acts as a connector between the IP infrastructure and high-bandwidth cells. The access controller is able to efficiently schedule (distribute) streams to different mobile nodes. Recently, we have extended this mechanism so that video streams are distinguished from less time-critical applications, e.g., Web, file transfer, etc.

5.3.1. Components of a Discontinous Mobile Networks

Streamed data must be received by a user device terminal in a continuous manner in a way that masks the jitter inherent in best-effort networks (such as the IP networks). The Internet's RTP (Real-Time Protocol) is an example of a protocol designed with this goal. Its implementation makes use of data caches on the network as well as time-stamps in the headers of transmitted data.

A key assumption in our work is that streaming data on discontinous mobile networks poses similar problems to that of RTP on IP. The device can only receive data at intermittent periods, so must exploit its periods of connection to load data in advance so that the stream contents may be viewed in a fluid manner. We thus decided to introduce caches into the network infrastructure. One cache is placed on the device itself. Another cache is placed on the network, somewhere that is "near" to the device. The goal of this cache is to keep data close to where mobile users connect to the network. There are two possible places for the cache: either in the content server or in the network's wireless access points.

1. Managing caches on the *content server* actually requires important changes to existing streaming protocols. Moreover, since a content server is placed somewhere on the fixed IP network, the forward transmission of cache contents is subject to jitter. Further, this solution does not make use of any available storage space available in wireless hotspots that a device might be carried into.

[5] http://www.laas.fr/mosaic

2. When the cache is placed in the *access point*, the cache thus operates on the outskirts of the fixed network infrastructure. Having the cache at this point has the advantage that fixed network jitter is removed from end-to-end streaming. On the other hand, this approach requires knowledge of, or at least models for, user mobility since changing the access point involves changing the cache location. This can be hard to manage.

We extended the second solution by introducing an *access controller* into the architecture. Its role is to load a cache in advance and to efficiently direct streaming content to the cache in the user's current access point. The access controller interacts with the content server and transmits the stream to the set of access points that a person is likely to use. In the future, the access controller will adapt content for a device based on the physical properties, and constraints, of the device. Figure 4 illustrate the main protocol layers for the content server and access controller.

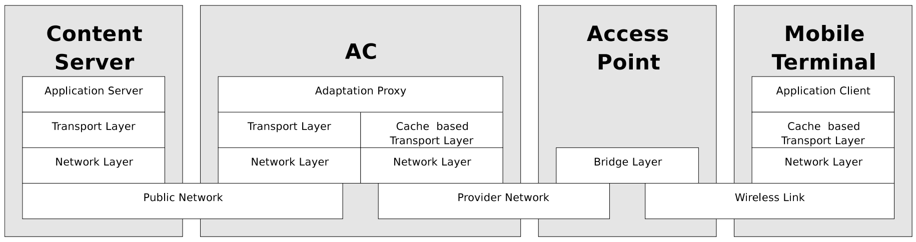

Figure 4. Protocol elements of access controller and content server exchange.

5.3.2. Access Controller Architecture

Figure 5 shows a functional segmentation of the internal architecture of an access controller. The flow adaptation service transforms the fixed-network side's flow to a cache compliant flow. It removes the transport or application layer timing dependency, and prefetches data blocks in order to benefit from good radio condition availability. The transformed stream is then stored, ready to be sent on demand.

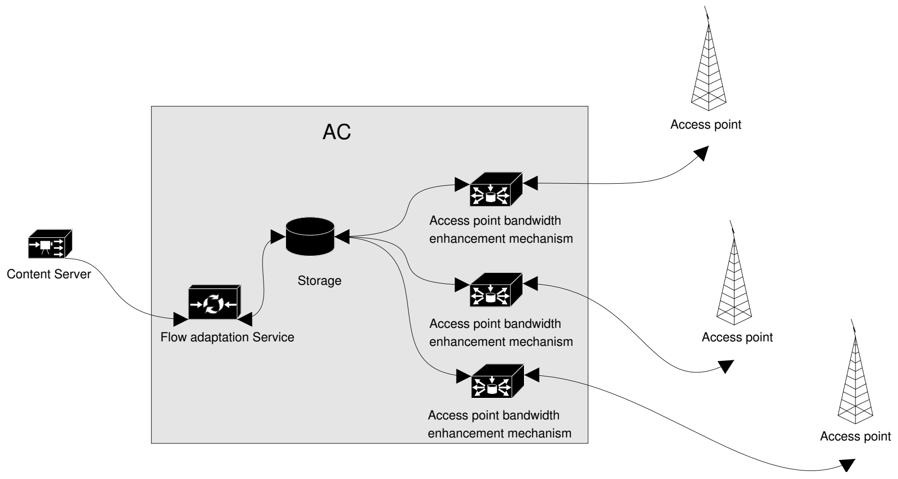
AC
Content Server
Flow adaptation Service
Storage
Access point bandwidth enhancement mechanism
Access point bandwidth enhancement mechanism
Access point bandwidth enhancement mechanism
Access point
Access point
Access point

Figure 5. Access controller architecture.

More specifically, each “bandwidth enhancement mechanism” allows the access controller to individually manage the communication link with each access point; this part is presented in Figure 6. When a device enters into the access point’s coverage, their streams are registered with a scheduler. The “access point bandwidth and flow control” module is the heart of the bandwidth enhancement mechanism. It supplies data, which must be sent to a device, and avoids congestion at the access point through a flow control mechanism. The scheduler selects data to be sent. We have studied several scheduling policies combining (1) the internal cache state of the mobile device, and (2) a data rate probing mechanism of the communication link.

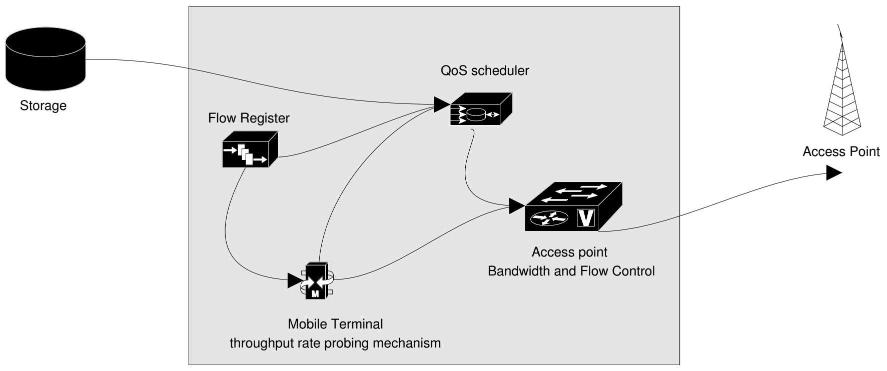

Figure 6. Bandwidth Enhancement of an Access Controller.

When a person wonders out of range of a network, it is not possible to know whether he will come back within a short time interval. For the scheduling policy, we have fixed this parameter to the longest time out of cover the service is assured (*ToC* = data size / streaming service data rate). Thus, if a user is out of coverage for a time longer than *ToC*, service disruptions may appear. In our approach, the scheduler manages two queues. The highest priority queue contains streams for devices with “internal cache duration” lower than *ToC*. The second queue contains the other streams. To distribute data, the scheduler checks if some flows are registered in the first queue; it then distributes packets of each flow using a round robin on the queue. Otherwise, it does the same thing for the second queue. We have evaluated this mechanism for several network topologies and variable users densities, to see whether the scheduler ensures that devices are fed enough to survive network loss. In the case where there is some remaining available bandwidth between the access controller and the device, the scheduler is able to provide “extra data” that might be consumed later when the device is attached to an access point under high load.

5.4. Java Operating Systems

Participants : Fabien Allard, Michel Banâtre [contact], Arnaud Guiton, Cédric Mosch, Cyril Ray, Jean-Paul Routeau, Claude Vittoria

Our work on Java operating systems has been going on for three years now. At the end of 2004, we had completed the design of a Java operating system for the underlying Java processor and proposed the idea of *native objects* to encapsulate hardware devices. We had started to implement the system over a Linux platform and, in this context, implemented the TCP/IP stack in the Java language.

Over this past year (2005), we have principally concentrated on the specification of our Java abstract machine around the bytecode processor. In particular, we formalised the integration of native objects and interrupt handling into the machine. This leads to the layered operating system model that is illustrated in Figure 3. In this model, the methods of native objects are coded as a special bytecode instruction that is micro-coded using the instruction set (C-ISA) supported by the associated hardware device.

There has been a considerable amount of implementation work on the project. We started coding a software implementation of the bytecode processor based on the K virtual machine (KVM); this yielded what we name the KVM BYTECODE PROCESSOR (KBP). Native objects were developed for a frame-buffer based screen and a keyboard. Device drivers were written in the Java programming language for a UART serial bus, the screen and keyboard devices, and for an infrared (IRDA) device. At the Java service layer (c.f., Figure 3), we developed a thread scheduler in Java for the CLDC variant of the Java (J2ME) environment and at the operating system services layer, we developed a protocol stack for the GPRS mobile phone protocol and integrated this with the IP stack that we developed last year. Finally, we experimented with the classloader mechanism to allow independent applications to be concurrently run over our Java operating system.

Two PhD students are currently working on the our Java operating system project. *Cédric Motsch* is studying the integration of multi-tasking (for running multiple applications) over the operating system and wrote a survey of this domain. *Arnaud Guiton* is currently writing a survey of object-based operating systems as well as JVM memory management. Our work on the design and implementation of the GPRS stack and on multi-application support were done in the context of French Master theses.

6. Contracts and Grants with Industry

6.1. National contracts

6.1.1. Texas Instruments

- Number: 198C2730031303202
- Title: Real time Java Distributed Processing Environment
- Related Research activity: see section 5.4
- 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.

6.1.2. Alcatel

- Number: 101C078
- Title: Advanced service delivery for 4G discontinuous networks
- Related Research activity: see section 5.3
- Partner: *Alcatel*
- Founding: *Alcatel*
- Starting: 01/09/2002, ending : 30/10/2004
- Extension starting: 01/11/2004, ending : 31/10/2006

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.

6.1.3. JCDecaux

- Number: Inria 401
- Title: Tuple-ware,
- Related research activity: see section 5.2
- Partner: *JCDecaux*
- Founding: *JCdecaux*
- Starting: 01/11/2004, ending 31/05/2005

The goal of this contract was to implement a pilot (named Tuple-ware), in order to help the JCDecaux company to evaluate the potential of ambient computing services based on ACES technologies in the area of street furniture and billboard.

7. Other Grants and Activities

7.1. European actions

7.1.1. 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), Asociacion de Investigacion y CooperacionIndustrial de Andalucoa (Spain), Institut National de Recherche en Informatique et 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.

7.1.2. NoE Resist

- Title: Resilence and Survability for IST
- Head: LAAS
- Starting: beginning of 2006

The NoE ReSIST (Resilence and Survability for IST) will focus on the following four objectives in addressing the scalability of dependability and security via resilience:

- Integration of teams of researchers so that the fundamental topics concerning scalably resilient ubiquitous systems are addressed by a critical mass of co-operative, multi-disciplinary research.
- Identification, in an international context, of the key research directions induced on the supporting ubiquitous systems by the requirement for trust and confidence in AmI.
- Production of significant research results that pave the way for scalably resilient ubiquitous systems.
- Promotion and propagation of a resilience culture in university curricula and in engineering best practices.

7.2. French initiative for research in security and informatics

7.2.1. 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.

8. Dissemination

8.1. Animation of the scientific community

8.1.1. Program committees

- PC member of the first International workshop on systems and networking for smart objects (SANSO 2005), (M. Banâtre)
- PC member of Second European Workshop on Wireless Sensor Networks (EWSN 2005), February 2005 (M. Banâtre)
- PC member of Third European Workshop on Wireless Sensor Networks (EWSN 2006), February 2006 (M. Banâtre)
- 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 and P. Couderc)

8.1.2. Organizing and reviewing activities

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 (end of 2005).

8.2. National and international working groups

Participation to the GdR I3 (Information - Interaction - Intelligence), group Mobility and Ubiquitous Computing (F. Weis)

8.3. Teaching activities

- Ifsic
  - Responsability of the optional lecture on Operating Systems in masters in Computer Science (M. Banâtre, P. Couderc, F. Weis),
  - Responsibility of the lecture on Distributed Operating Systems in "Diic 3 ARC" (final year of masters) (M. Banâtre, P. Couderc and 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),
- 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, October 2005 (M. Banâtre).

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

8.4. Internship supervision

We have supervised the following internships in 2005:

- Fabien Allard (Engineer student, ENSEIRB Bordeaux)
- Julien Lancia (Engineer student, ENSEIRB Bordeaux)
- Mickaël LeBaillif (Engineer student, ENSEIRB Bordeaux)
- Xavier Le Bourdon (masters student, University of Rennes)
- Damien Martin-Gutteriez (masters student, University of Rennes)
- Ronan Ménard (masters student, University of Rennes)

8.5. 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:

- Mise en oeuvre des interactions de proximité ans le cadre de l'ubiquité numérique : l'exemple des Sysèmes d'Information Spontané GDR I3, groupe de travail "Mobilité et Ubiquité", CNAM Paris, January 2005 (F. Weis)
- Michel Banâtre and Paul Couderc have organized an INRIA session day on "Ubiquitous Computing and sensor networks", with the participation of scientific experts in these two research areas (P. Bonnet - DIKU, H. Saikonen - NOKIA, K. Rothermel - University of Stuttgart, D. Rouffet - ALCATEL RI, D. Simplot - LIFL, P. Couderc - INRIA), March 2005

8.6. Thesis committees

- M. Martin Valera Rico, University of Rennes, November 2005, *Evaluation pseudo-subjective de la qualité d'un flux multimédia et ses applications au contrôle*, "examinateur" (M. Banaâtre).

8.7. Patents

- M. Banâtre, P. Couderc, M. Becus, « *Installation pour la diffusion contextuelle d'informations en mode à la fois collectif et personnel.* » European patent application request file on April 2005, number: 0503678

8.8. Industrial transfers

As already detailed in section 2.1, we have strong activities in this area, related with ambient computing and spatial programming. But we have not yet succeeded. So we cannot give more details in this current report.

9. Bibliography

Articles in refereed journals and book chapters

[1] M. BANÂTRE. *Encyclopédie sur l'informatique et les systèmes d'information*, chap. Une introduction aux architectures et aux systèmes distribués, Vuibert, April 2006.

[2] A. TROEL, F. WEIS, M. BANÂTRE. *Prise en compte du mouvement dans les systèmes de communication sans fil*, in "Techniques et Sciences Informatiques", vol. 24, n^o 1, 2005, p. 65–94.

Publications in Conferences and Workshops

[3] J. PAUTY, P. COUDERC, M. BANÂTRE. *Using Context to Navigate through a Photo Collection*, in "Proc. of the International Symposium on Mobile Human-Computer Interactions (Mobile HCI '05), Salzburg, Austria", September 2005.

[4] J. PAUTY, P. COUDERC, M. BANÂTRE. *Atomic Token Passing in the Context of Spontaneous Communications*, in "Workshop on Applications and Services in Wireless Networks (ASWN05), Paris, France", June 2005.

[5] J. PAUTY, P. COUDERC, M. BANÂTRE. *Spatial Programming: Using the Physical World as a Computing System*, in "UbiPhysics workshop / Ubicomp 2005, Tokyo, Japan", September 2005.

[6] M. TLAIS, F. WEIS, C. BONAN. *Mobility prediction in 4G D-Cov Networks*, in "Proc. of the 1st IEEE International Workshop on Performance Analysis and Enhancement of Wireless Networks (PAEMN'06), Vienna, Austria", April 2006.

Internal Reports

[7] M. BANÂTRE, A. GUITON, C. MOTSCH, C. RAY, J.-P. ROUTEAU, C. VITTORIA. *Java Operating System*, Technical report, n^o 1769, IRISA TR, December 2005.

[8] J. PAUTY, P. COUDERC, M. BANÂTRE. *Atomic Token Passing in the Context of Spontaneous Communications*, Technical report, n^o RR-5445, INRIA, January 2005, http://www.inria.fr/rrrt/rr-5445.html.

[9] J. PAUTY, P. COUDERC, M. BANÂTRE. *Using Context to Combine Virtual and Physical Navigation*, Technical report, n^o RR-5496, INRIA, February 2005, http://www.inria.fr/rrrt/rr-5496.html.

Bibliography in notes

[10] (INTEL CORP.). *Introduction to the iAPX 432 Architecture*, Intel Corporation, Santa Clara, CA, 1981.

[11] B. ALPERN, C. R. ATTANASIO, J. J. BARTON, A. COCCHI, S. F. HUMMEL, D. LIEBER, T. NGO, M. MERGEN, J. C. SHEPHERD, S. SMITH. *Implementing Jalapeño in Java*, in "Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications", 1999, p. 314–324.

[12] K. ARNOLD, J. GOSLING. *The Java Programming Language*, The Java Series, Addison-Wesley, 1996.

[13] BURROUGHS. *Burroughs B6700 Information Processing Systems Reference Manual*, 1972.

[14] D. ESTRIN, R. GOVINDAN, J. HEIDEMANN. *Embedding the Internet*, in "Communications of the ACM", vol. 43, n^o 5, May 2000, p. 39–41.

[15] E. FREEMAN, S. HUPFER, K. ARNOLD. *JavaSpaces Principles, Patterns, and Practice*, Addison-Wesley, Reading, MA, USA, 1999.

[16] D. GELERNTER. *Generative Communication in Linda*, in "TOPLAS", vol. 7, n^o 1, jan 1985.

[17] K. HAMEED. *The Application of Mobile Computing and Technology to Health Care Services*, in "Telematics and Informatics", vol. 20, n^o 2, 2003, p. 99–106, http://dx.doi.org/10.1016/S0736-5853(02)00018-7.

[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] C. WAWERSICH, J. KLEINDER, M. FELSER, M. GOLM. *The JX Operating System*, April 22 2002, http://citeseer.ist.psu.edu/561091.html.

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

[21] M. WOODGER. *An introduction to ALGOL 60*, in "The Computer Journal", vol. 3, July 1960, p. 67–75.

[22] W. WULF, E. COHEN, W. CORWIN, A. JONES, R. LEVIN, F. POLLACK. *HYDRA: The Kernel of a Multiprocessor Operating System*, in "Communications of the ACM, CACM", vol. 17, n^o 6, June 1974, p. 337–345.

[23] P. WYCKOFF, S. MCLAUGHRY, T. LEHMAN, D. FORD. *T Spaces*, in "IBM Systems Journal", vol. 37, n^o 3, 1998, p. 454–474.