INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team aces

Ambient Computing and Embedded Systems

Rennes - Bretagne Atlantique

THEME COM

Table of contents

1. Team

Head of project-team
Michel Banâtre [Research Director, INRIA]

Administrative assistant
Evelyne Livache

INRIA project staff
Ciaran Bryce [Research Director, INRIA]
Paul Couderc [Research Scientist, INRIA]

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

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

Project technical staff
Fabien Allard
Mathieu Becus
Ronan Menard [up to 31/10/2007]

Ph. D. student
Damien Martin-Gutteriez [ENS grant]
Mickaël LeBaillif [Inria grant]
Xavier Le Bourdon [MESR grant]
Pierre Duquesne [INRIA grant]
Azza Jediddi [INRIA grant]
Cedric Mosch [Inria grant, up to 30/4/2007]
Arnaud Guiton [Inria grant, up to 31/12/2007]
Mazen Tlais [Inria grant, up to 31/12/2007]

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). Today, around 80% of the French adult 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 [20], thus facilitating the deployment of a variety of information systems that control physical objects. The third phenomena is the increasing variety of wireless networks available for personal and embedded devices, e.g., Bluetooth, Wifi, GPRS, etc.

The combination of these three phenomena has permitted the emergence of context-aware person-centric applications and collaborative personal environments. These services complement a person's physical ability to interact with his environment. They are tailored to the needs, preferences and location of each person carrying a device, and are continually available. Services range from critical, e.g., remote health monitoring [23], 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* [31], and several significant research challenges remain. First, to facilitate mobility, ambient computing services should require minimal device manipulation by the device owner. It is crucial that the computing device operate as an extension of the person rather than as a tool. Second, there must be a way of modeling 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 three angles:

- *Programming Models for Ambient Computing*. We have looked at ways of modeling the physical environment in the virtual environment of programs in order to facilitate ambient application development. The goal is to be able to write programs that address and navigate through objects in the physical world as elegantly as a program traditionally manipulates a computer's main memory.
- *Quality of Ambient Service*. A user needs to be able to exploit ambient services as seamlessly as possible. In particular, he should be shielded from the effects of network breaks – something that can be quite common for wireless environments. We have been developing for ensuring continuous video streaming for mobile devices.
- *Operating System Support*. We are looking at portable operating system designs for personal devices that manage limited physical resources and irregular network connectivity.

In addition to technical work on ambient systems, ACES has been looking at a broader range of issues over the last year in anticipation of a new INRIA project. These issues relate to collaborative environments in general. One issue is *Privacy in ambient systems*; this is needed in order to ensure user-acceptability for services and to ensure that services operate legally. A second issue is *digital restrictions management* (DRM) that define and enforce usage controls on content exchanged between devices. These are important in ambient and personal computing settings for security in order to defend privacy, licensing and general economic interests. Notably, the TPM (Trusted Platform Module) can be used to help support DRM. Finally, Free and Open Source Software (F/OSS) is a movement that permits access to software source code, and, encompasses a software development paradigm that harnesses the efforts of a distributed community. We have participated in the formalization of this collaborative process in order to identify information system artifacts to support its operation.

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 [22] 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 [21], IBM's T-Space [34]) 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 (Spatial Information Systems) 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.

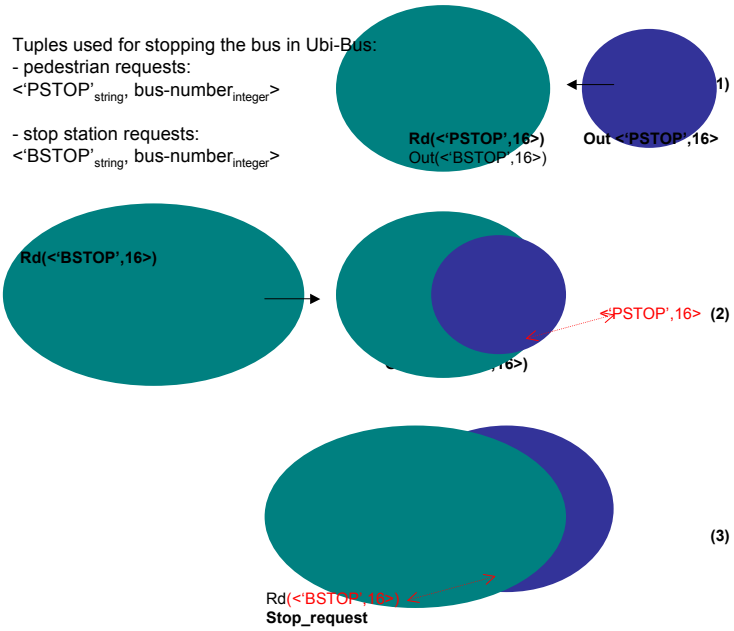

Figure 1. Physical Tuple Space

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.

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 [26]. 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 past few years have witnessed the rise of the cellular networks. These communication systems were designed with a philosophy of *any-time any-where* service. Users wish to receive and place calls at any location and without delay, to move while talking without interrupting their conversations. This requires ubiquitous coverage, which in turn requires significant infrastructure. A modern cellular system is installed with hundreds of base stations, at a cost of hundreds of millions of euros, in order that a communication link is always available. Such any-time any-where service provision becomes increasingly expensive and suffers from low bandwidth. Covering wide areas with high radio bandwidth requires complex equalization, due to signal attenuation, multi-path fade, and shadowing effects. Sophisticated radio engineering will lead to improved bandwidth, coverage, and mobile access, but this will be expensive, in terms of both capabilities and cost.

In this context, the ACES project has studied an alternate design for wireless networks where intermittent but very high speed is provided to the network through *Pico-cells*. The latter consists of a set of access points (APs), *i.e.* antennas around of which are defined radio cells with limited range (about 100 meters). Those antennas are discontinuously spread on the network area, thus providing a *many-time many-where* service. Actually, the idea of coverage discontinuity brings two major advantages. First, as it implies the use of a fewer number of access points, the architecture deployment will be cheaper. Second, radio cells disjunction hypothesis simplifies the radio frequency band management and avoids interference problems.

Even if this model simplifies network deployment, the connectivity intermittence induces important challenges in order to avoid service disruption. Thus, terminals have to take advantage of the high bandwidth when it is available. For delay tolerant data, a terminal stores data as it passes under a cell. Hence, it may consume the buffered data even when it passes through regions of poor network coverage. Many projects have studied very specific cases where the cells deployment is uniform and data is sent from servers to terminals (down-link). One example of this type of system is studied by WINLAB (Wireless Information Network Laboratory). In this project, cells are equally spaced and the data delivery algorithm is tested in a network with one dimensional system *i.e.* high ways.

Our approach is based on a more general case in which cells are distributed according to the envisaged traffic and data can be exchanged in both directions down-link (from servers to terminals) and up-link (from terminals to servers). The main challenge is to provide system mechanisms and efficient services addressing the specific constraints of this architecture: discontinuous coverage, user unconstrained mobility, high user density. 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

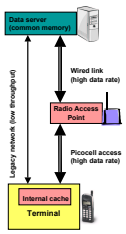

Figure 2. Pico-cell architecture model

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

Recent years have seen a huge proliferation of applications developed in the Java^{TM} programming language [17]. 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 had a previous collaboration 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 can run on any abstract Java bytecode processor platform.

The advantage of a Java operating system running on an abstract bytecode processor 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

[1] http://java.sun.com/j2me/

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 [30], JNode[2], OVM[3], Jalapeño [16], 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 any 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 [18] designed to run Algol-60 [32] and the SAR designed to support Algol-68. Other notable efforts include Hydra [33] from CMU and the iAPX432 from Intel [24]. 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 implemented independently of the Java layers: in microcode for hardware implementations of the abstract processor, in C or even Java for software implementations.

The structure of our Java operating system is illustrated in Figure 3. The native objects are placed at the same level as the abstract 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.

[2] http://www.jnode.org
[3] http://www.ovmj.org

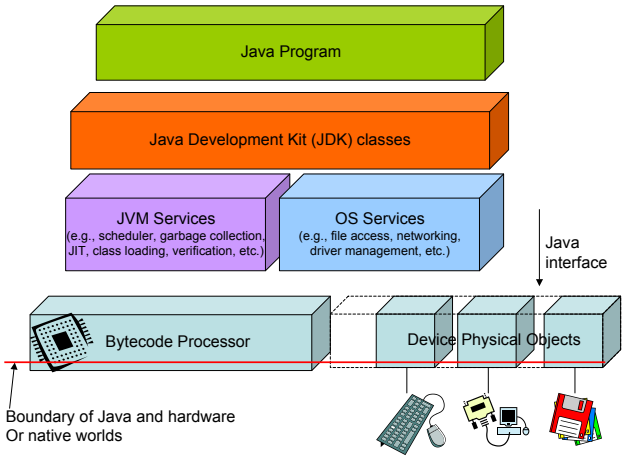

Figure 3. Structure of Java-based Operating System

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

3.4. Collaborative Computing Processes

Today's systems have entered the era of community computing – the antithesis of personal computing. While people do possess their own handheld and personal computers, they indelibly rely on the community of computers and users for content, code, and services. Users also contribute resources – this phenomenon underlying the open source movement [27], utility computing, peer-to-peer computing, B2C, B2B, etc. Nonetheless, each user is constrained in his use of content and services by community rules, these being expressed through licensing and IP contracts, the Law (e.g., Sarbanes-Oxley [28], Basle II) or organizational rules. Thus, systems built today must be aware of digital restrictions management and support organizational requirements.

Community computing devices do not simply run applications – they participate in *processes*. A process is a goal-directed, inter-related set of activities. An example of a process is the operation of an on-line boutique service. The activities of this process include Web server maintenance, customer lists management, catalogue production, payment, etc. One challenge for process managers is to coordinate activities; for instance, information from the catalogue activity must be fed to the Web maintenance activity, feedback from the payment activity is needed for customer management. In effect, process efficiency depends on facilitating information flows between activities. A second important aspect of processes is the presence of legal and organizational issues, e.g., privacy (for customer data retention), Sarbanes-Oxley for data archival, and intelligent attribution of tasks to people within the organization for the efficient running of the boutique.

The objective of the this activity is to study abstractions for *process programming models* and their implementation. The role of a process programming model is to express activities. These activities involve people, their devices, computers as well as environment (embedded) computing devices. Compared to standard programming languages, process languages need to express concepts like principals (for people, organizations, etc.), roles (for security and organizational tasks), rules for content protection and security, events (for activity coordination) and process metering (for performance, security, etc.). Much work has been done on modeling

processes in business information systems (e.g., BPEL [19]), though these are heavily dependent on XML. We would like an approach that is closer to high-level programming languages so that we can harness safety and portability. The case study of our approach is Free and Open Source software (F/OSS) processes, as this has legal, social, economic implications, and is also a method in which we can conduct our own developments.

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 mobile terminals, access points, admission control, content provision servers, WiMax MAC layers, wireless transmission, protocol cache management (based on the IETF SCTP protocol) 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 Bécus, Paul Couderc [contact], Damien Martin-Gutteriez, Mickael LeBaillif, Xavier Le Bourdon

The ambient computing activity for this year include three main topics, that we describe below. We published books on this topic this year [1], [2].

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 was defended by Julien Pauty in April 2006.

We started new works related to mobile robotic, where environment tagging can be used a cooperation infrastructure for swarm of robots. An important aspect of these systems is that coordination cannot be ensured by a central control station; physical data structures can be defined by disposing RFID tags in the environment, actually implementing a distributed memory mapped in physical space. We are currently focused on proposing coordination primitives for mobile robots, using RFID (which provide a spatially referenced memory) and short range communication for synchronization [9]. This work is done in the context of the Roboswarm project.

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 introduced in the past 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.

Implementing this definition relies on sensing technologies to collect information from physical objects. We are now using RFID, in combination with other technologies such as Bluetooth and WLAN that we used in in the past. RFID provides the advantage to allow precise spatial sensing, hence allowing more accurate geometrical definition for context settings. These new architectures will be experimented in a collaboration with Aeroports de Paris : Ubi Board, a context sensitive display system that 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.

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.

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 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, which studies 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.

Similar problems are studied to improve the reliability of cooperating swarms of robots: task contexts can be lost on robot failures, and must be recovered for the success of the work assigned to a swarm [5]. Without the assumption of a permanent communication link to remote reliable storage, we consider two types of backup storage which can be used by the robots: other robots in range of communication, and *a spatially distributed memory* implemented by RFID coverage. Determining good tradeoff between the use of these two backup facilities, depending on the application profile, are some of the important problem to solves. This work is done in the context of the Roboswarm project.

Finally, we are active on the emerging problem of distributed capture, which is quickly gaining much interest [8]. The proliferation of personal communication deviced enabled with powerful multimedia capture capabilities offers the opportunity for distributed capture infrastructures. Distributed capture offers interesting advantages over local captures: it allows better spatial and temporal coverage, as well as quality improvements. The difficult problem to solve in this domain is overcoming inaccuries in temporal and spatial references when aggregating data coming from distributed sources (coming from the lack of common reference, clock drifts and sensor errors), which raises inconstiencies in the aggregated media.

5.3. Discontinous Mobile Networks

Participants: Fabien Allard, Azza Jedidi, Ronan Ménard, Mazen Tlais, Frédéric Weis [contact]

Our first goal consists in designing a scalable streaming service for discontinuous networks. On the one hand, streaming servers are usually located in classical IP networks. Such networks are submitted to many constraints, like bandwidth variations. On the other hand, mobile terminals evolve in wireless discontinuous coverage areas, constituted of several access points. The first idea we propose in the context of such networks, is to design and introduce an intermediate equipment that will catch the video flows sent by the content server, and then efficiently distribute (*i.e.*schedule) them to mobile terminals in the network according to their radio conditions. This equipment is called the **Access Controller** (AC).

The AC acts as a cache which temporarily stores the part of the flow to be delivered. More precisely the AC is provided with many caches. Each of them is related with a single streaming request and contains the part of the flow to be sent to the terminal. A terminals also contains a cache to store some parts of the flow, in order to consume it while crossing out of coverage areas, and to avoid service disruption. And in the AC, a scheduling policy is implemented for classifying flows in priority queues, depending (1) on terminal positions in the radio cells and on (2) terminal cache contents.

5.3.1. Definition and evaluation of a discontinuous compliant transport protocol

The discontinuity of the coverage modifies the <<traditional traffic shape>> of a service. In a discontinuous context, the user throughput is null while the mobile user is out-of-cover. When the terminal arrives into a communication area, a burst occurs to provide all data delayed during it was out-of-cover. Then the mean throughput becomes constant, at the same value as the server can provide until the terminal goes out.

As a consequence, services which used to be implemented over an unreliable transport protocol, such as video streaming, must at least be built on a partial-reliable one to avoid congestion after a discontinuity. Indeed the burst of data may create a high packet loss rate due to congestions.

The <<cache based>> transport protocol used between the AC and the terminal must be <<reliable enough>> to ensure that required data are received by the remote host. For this reason, it must use mechanisms implemented in other transport protocols: delayed acknowledgement timer, retransmission timer, bandwidth estimation and flow control through a sliding window management.

Our approach is based on the SCTP[4]. SCTP is a TCP-like transport layer protocol, defined by the IETF Signalling Transport (SIGTRAN) working group. SCTP ensures reliable and in-sequence transport of messages with congestion control. Furthermore, SCTP is capable to multiplex independent flow of data and to use several network interfaces of an equipment to enhance the reliability of the communication. In our approach, SCTP works in conjunction with a scheduling policy for classifying flows in priority queues, depending on radio conditions and terminal cache contents [6].

This scheduling policy must adapt its transmission rate acording to the wireless link capacity. Many works were done about bandwidth evaluation using transport acknowledgments in the specific context of wireless communications. Our architecture is based on a *Westwood* like evaluation. The latter is computed according to the following formula:

$$Bp_i = \frac{19}{21} Bp_{i-1} + \frac{2}{21} \frac{\sum sizeof(Ack)}{\delta t}$$

where *sizeof(Ack)* is the amount of data that has been acknowledged during the evaluation period δt. A low-pass filter is used to average the evaluation in order to avoid drastic changes due to burst arrival of acknowledgements because of bufferised packets, or the lack of acknowledgements during one δt due to delayed acks (lower bandwidth), which could falsify the evaluation.

We have demonstrated through simulations that using a SCTP based protocol improves system scalability. We have evaluated the number of service disruptions according to users density. The result is that the network capacity (a fixed users density without service disruption) is improved of about 300 % (see figure 4 when a specific transport protocol is used between the AC and the terminals.

5.3.2. Enhancing the users' experience in the up-link flows

We have oriented our work, as in classical mobile networks, toward bidirectional flows. The problem of up-link flows management becomes more important because of advanced technologies of wireless networks, like, camera with capture functions. It implies that a user is able to manage any kind of data, like, photos and videos. The user wants that his data, stored in the MT cache, is often flushed to the wired network for multiple reasons: exchange of data or technical storage capacity.

We have explored the possibilities, mainly the access controllers' presence, to enhance the users' quality of experience in the up-link. Normally, the access network provider controls the deployment of access points, access controllers, and wired links between them. We have identified two main bottleneck in the path between mobile terminals and application servers. The first is related to radio communications and limited resources of access points. The second bottleneck concerns the IP backbone (Internet), since the access network provider does not have any control on this part of the network.

[4]Stream Control Transmission Protocol

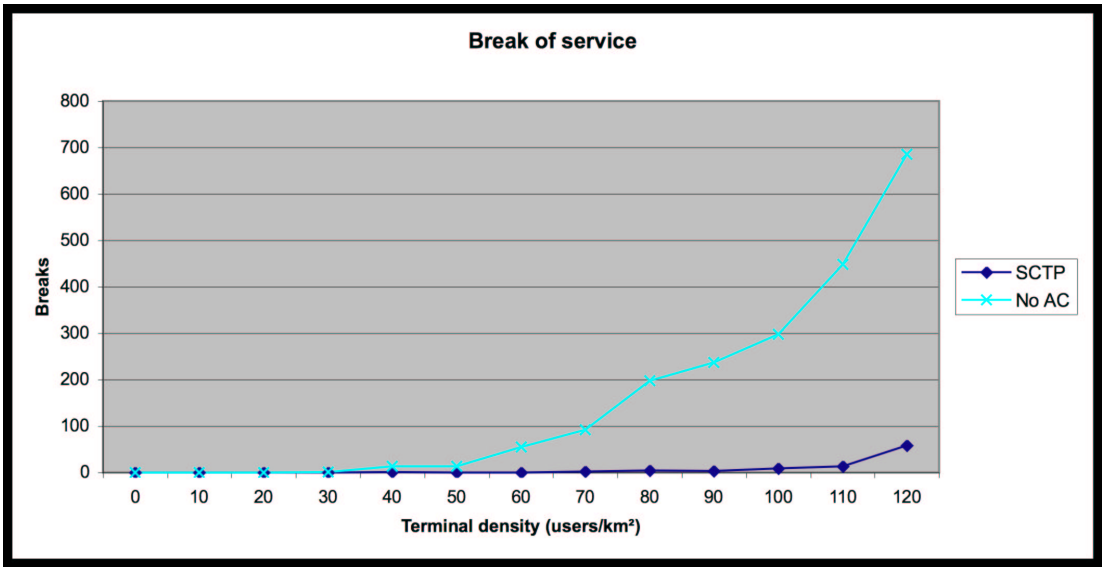

Figure 4. Evolution of service disruptions using SCTP and Westwood bandwidth evaluation

We have proposed to implement classes of service (CoSs) along the path between the MT and application server. The main goal of the CoSs is to associate a special behavior to the data, in order to enhance the users' quality of experience. More precisely, CoSs mechanism favor the upload of *popular data* versus *non popular data*. For doing so, we have modified dynamically the allocated bandwidth according to the *popularity index*. A model that supports the multi-ACs case has been proposed. This model is based on several messages that should be exchanged between MTs-ACs and ACs-application servers.

Our proposal has been validated by launching simulations. Results have shown that the CoSs have significantly improved the users' experience in terms of mean time needed to upload popular data, and mean bandwidth allocated to serve popular data. Nevertheless, this gain decreases when large data are uploaded and/or the rate of popular data is higher.

5.4. Java Operating Systems

Participants : Fabien Allard, Michel Banâtre [contact], Arnaud Guiton, Cédric Mosch, Jean-Paul Routeau, Claude Vittoria, Pierre Duquesne, Ciarán Bryce.

Our work on Java operating systems has been going on for four 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. Our group has also been collaborating on a Java Specification Request for resource management, and on a security model for Java multi-tasking environments with researchers from Sun Microsystems.

5.4.1. Pure Java Platforms

In the past two years, we 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 instructions that are micro-coded using the instruction set (C-ISA) supported by the associated hardware device.

There was 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.

One PhD student completed his thesis on the project this year. *Cédric Motsch* integrated a multi-tasking core (for running multiple applications) into the operating system and wrote a cross-application scheduler [3]. The main innovation in the platform is the introduction of application domains. A domain can encapsulate an application and its JDK and can be isolated from other domains. Restricted sharing between domains is nonetheless possible. *Arnaud Guiton* is currently studying memory management for the Java platform.

One of the new topics that we began studying this year is software patching. The code base of systems and applications are dynamic: they continue to change even after delivery of the software to clients. These changes are implemented through the application of *patches*. Though their primary necessity is to fix security holes and discovered bugs, patches are even used to add new functionality. Patching requires a formal framework that ensures that modifications can be safely made to running applications.

5.4.2. Resource Management

The group participated in a Java Specification Request for a resource management interface for the Java platform. Resources include CPU, heap memory, network bandwidth; a systems programmer can define further abstract resources, e.g., database connections, servlet invocations, etc. Monitoring resource consumption and imposing limits on consumption are key requirements for any platform. Until now, no such mechanism existed for the Java platform. The JSR was in progress for nearly two years, and its output is a Java API and reference implementation. The API specification has now entered a public review stage. In the context of JSR 284, we collaborated with representatives from Google, IBM, BEA Systems, Intel, Nokia, Siemens, Motorola and EPFL-Switzerland. This collaboration led to the final draft of the specification, published in 2007.

5.5. Information System Processes

Participants: Ciaran Bryce, Pierre Duquenes

5.5.1. DRM, Trusted Platforms and Privacy Issues

One of the community computing areas that we are looking at is privacy enforcement for ambient systems. We are currently collaborating in the PRIAM project, whose approach to privacy is that technology need not necessarily be an obstacle to privacy protection: if legal and social issues are considered from the outset, then technology can also be used to allow individuals to exercise their rights. We believe that the key issue is to devise techniques able to support ambitious privacy protection policies while allowing for the flexibility required in the ambient intelligence context. We illustrate our position using three technical requirements: (1) formal specification of privacy policies, (2) trust management and (3) auditability, which show both the challenges posed by ubiquitous computing and the opportunities to strengthen privacy. The Aces group is looking at how the TPM can be exploited to enforce privacy policies [7].

The TPM (*Trusted Platform Module*) [29] is an example of a general purpose hardware chip designed for trusted computing. This is the result of a design effort of the Trusted Computing Group – a consortium composed of major hardware and software manufacturers including AMD, Intel, ARM, Microsoft, Sun Microsystems, Ericsson, Nokia, and HP. The chip can be integrated into any device from a PDA to a Web server platform. TPMs are now shipped with PCs; 200 million TPM-enabled PCs will have been shipped by the end of 2007. We have been looking at how the TPM can be used to permit a device to validate the “acceptable behavior" of a partner device.

With respect to DRM, we are participating in the P2PImages project, whose goal is the design and implementation of a peer-to-peer infrastructure for the legal exchange of content. The ACES group is working in the security sub-project; we are interested in the expression and trustworthy implementation of usage constraint expressions.

5.5.2. Free and Open Source Software

Free and Open Source Software (F/OSS) is one of the great facts of software development of the past few years. In this approach, a community of people with common interests collaborate to develop new ideas, models and, *in fine*, produce freely available software. The software is distributed with its source code which can be freely modified by any developer; the modifications made by this developer are, in turn, made available to the community.

In a F/OSS project, the interests of the community push design requirements and software licenses regulate intellectual property rights. A F/OSS community is responsible for all aspects of software development, from requirements to coding, testing, and even manual compilation and translation. Thus, a F/OSS community is self-organizing; compared to the proprietary software model used by some companies, there is no hierarchal organization of the community. For this reason, F/OSS is said to be organized like a *bazaar*, as opposed to the *cathedral* model of proprietary software development [27].

Increasing the size of a F/OSS project community brings great potential to a project – more ideas, code and developers. Nonetheless, this increase can be accompanied by a management challenge due to the fragmented and distributed nature of actors and activities, dependencies between projects and the highly dynamic environment of F/OSS. For instance, as of 30 October 2006, the standard Mandriva Linux distribution contains 10566 packages, with an average package size of 2717431 bytes. These package involve more than one hundred different licenses [25]. Gaim [13], one of the most active projects on SourceForge, had an average of 533 daily read transactions in August 2006, and 17 daily write transactions per day for an average of 101 files updated per day. These figures suggest that managing a large F/OSS project is already a challenge:

- Possibility of locating competence in the community. Managing an activity entails harnessing the competence of community members. Competent and potentially interested members must be located. Apart from news-groups and mailing lists, there is no way of actively locating potential activity collaborators.
- Information about activities and participants need to be made available to the community. For instance, a user seeking to install a package must be immediately informed of any detected configuration error. Similarly, coding activities must be aware of information from testing; development activities need to be informed of information on community profiles produced by community management activities.

F/OSS is an example of a virtual process. This is a set of activities (e.g., coding, testing, community management, etc. in F/OSS), resources (e.g., packages, configurations), roles (e.g., developers, project committers). We participated in the design of a *Process Reference Model* (PRM) that formalizes the F/OSS process [11], [10]. The model permits meta-data, or *attributes*, to be bound to artifacts to simplify the classification and localization of artifacts. Attributes express information such as defect reports, patches, configuration requirements, topics. Attributes facilitate coordination between activities since all information required by an activity (e.g., patch development) that is produced by another activity (e.g., testing) is expressed in the artifact attributes (e.g., defect report). The process model can also permit different F/OSS projects to be compared and contrasted based on the activities each undertake. Existing project methodologies and supporting tools [14], [15] tend to be limited to project artifacts, ignore the organizational aspects of the process or ignore process interactions. We contend that F/OSS processes can only be improved by addressing process issues.

6. Contracts and Grants with Industry

6.1. National contracts

6.1.1. Aéroport de Paris

- Title: Conception et mise au point de pilotes de services aéroportuaires reposant sur les technologies de l'informatique diffuse développées par l'INRIA- UR Rennes.
- Partner: *Aéroport de Paris*
- Starting: 01/11/2006, ending : 15/02/2007

6.1.2. TV mobile <<sans limite>> (TVMSL)

- Partner : Alcatel Mobile Broadcast, Alcatel Alenia Space Alcatel, DiBcom, Sagem Communication, TeamCast, Radio Frequency Systems, UDcast, CEA-Léti, Inria-Rennes, CNRS-L2S
- Starting: 01/11/206, ending : 30/04/2009

The objective of this contract is to design and evaluate inovative caching mechanisms to distribute H.264 flows in future hybrid DVB-H+ infrastructure (satellite + terrestrial).

6.1.3. 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 : 31/10/2007

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.

6.1.4. 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. European Project: Roboswarm

- Title: Robot Swarms
- Proposal/contract no: 045255
- Tallinna Tehnikaulikool- Estonie, ELIKO Tehnoloogia Arenduskeskus- Estonie, Institut National de Recherche en Informatique et en Automatique- UR Rennes, TEKNILLINEN KORKEAKOULU- Finlande, OULUN YLIOPISTO- Finlande, FUNDACION FATRONIK- Espagne, KTH - Kungliga Tekniska Hogskolan Royal Institute of Technology- Sweden, IDMIND-ENGENHARIA DE SISTEMAS, LDA -Portugal, Universita degli Studi di Genova -Italie
- Starting: November 2006, ending: April 2009

Roboswarm is an EU project that started in November 2006 in which ACES is a participant. The goal of the project is to develop an open knowledge environment for self-configurable, low-cost and robust robot swarms usable in everyday applications. Advances in the state-of-the art of networked robotics are proposed through introduction of a local and global knowledge base for ad hoc communication within a low-cost swarm of autonomous robots operating in the surrounding smart IT infrastructure. The ACES group was invited into this consortium due to its experience with ad hoc (ambient) network environments.

7.1.2. Smartmuseum

- Title: Smartmuseum
- Partners: Competence Centre of Electronics-, Info- and Communication Technologies, ELIKO (Estonia), Helsinki University of Technology TKK (Finland) Kungliga Tekniska Högskolan KTH (Sweden), Webgate JSC, (Bulgaria), Heritage Malta, (Malta), Institute and Museum of the History of Science, (Italy), Apprise, (Estonia).
- Starting: end of 2007; ending: December 2010

The general objective of the SMARTMUSEUM project is developing solution and IT services for user interest dependent (profiled) access to digitalized cultural information that is relevant in particular physical location. The activities of the project are addressing personalised approach to cultural exploration, including cultural tourism. The future smart museum IT infrastructure and services, which are capable of increasing bidirectional interaction between multilingual European citizens and cultural heritage objects taking full benefit of the multi source digitalized cultural information. By doing this, priorities are set on: Improving structured and user competence dependent access to the vast repository of cultural heritage, Improving the meaning and individual experience people receive from cultural and scientific resources, Bringing personalized cultural experience closer to non-expert community, Making real reuse of experiences related with cultural heritage access for variety of interest groups.

7.1.3. 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.

Michel Banâtre is the scientific leader of the Work Package 2, which is the "more research oriented" work package of the RESIST NoE

7.2. French initiative for research in security and informatics

7.2.1. Region: P2PImages

- Title: Privacy in Ambient Computing Systems
- Partners: ENST Bretagne, France Télécom, CREM, IRISA/INRIA, IPdiva, Mitsubishi Electric, THOMSON.
- Starting: October 2007 to October 2009

The goal of P2PImages is to investigate the design of platforms for legal content exchange in peer-to-peer environments. The goal of Aces in this project is to study digital rights management issues.

7.2.2. ARC: PRIAM

- Title: P2PImages.
- Partners:INRIA (Aces, Ares, PopArt), University Jean Monnet (Saint-Etienne), University of Twente (Netherlands).
- Starting: January 2007 to December 2008

The PRIAM project addresses the privacy issues in our ambient world in a transversal and multidisciplinary way, favoring the exchange of ideas between lawyers and experts from the information and communication technology.

7.2.3. ACI: Mosaic

- Title: Mobile System Availability Integrity and Confidentiality
- Partners: Institut Eurecom, Institut de Recherche en Informatique et Systèmes Aleatoires (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 International Workshop on Privacy and Security in Agent-based Collaborative Environments, PSACE 2007. (C. Bryce)
- PC member of 2008 ACM Symposium on Applied Computing, Track Security. (C. Bryce)

- PC member of the 9ème rencontres francophones sur les aspects algorithmiques des télécommunications, ALGOTEL 2007. (F. Weis)
- The 18th Annual IEEE Internal Symposium on Personal, Indoor and Mobile Radio Communications, (PIMRC 2007) (M. Banatre)
- The First International Workshop on Specialized Ad Hoc Networks and Systems (SAHNS 2007) (M Banatre)
- The 2nd European Conference on Smart Sensing and Context (EuroSSC 2007) (M. Banatre)

8.1.2. Conferences, meetings and tutorial organization

F. Weis is organization chair of the french Ubimob 2008 conference, Saint-Malo, France.

M Banatre is scientific organizer of a invited session on Ubiquitous Applications in Autonomics Oct. 2007. He was also scientfic co-organizer of the US France Young Engineering Scientists Symposium : "Environment and Sensor Networks" Washington, Oct. 2007

8.1.3. Organizing and reviewing activities

Ciarán Bryce is a member of the expert scientific committee for the ANR (*Agence national de recherche*) Global Security program.

8.2. National and international working groups

F. Weis is participating in the GdR I3 (Information - Interaction - Intelligence), group Mobility and Ubiquitous Computing.

C. Bryce is participating on behalf of INRIA in a liaison committee of the Trusted Computing Group that is organized by the French National Defense department.

C. Bryce is the International Affairs correspondent for INRIA at INRIA-Rennes; he is also a member of INRIA COST on International affairs.

8.3. Teaching activities

- Ifsic
  - Responsibility of the lecture on Ambient Computing and 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 Ambient Computing and Distributed Operating Systems (final year of masters) computer science department (M. Banâtre).
  - 6 hours of lectures on Computer Security to final year engineering students (C. Bryce).
- 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 2007 (M. Banâtre).
- Universities of Geneva, Lausanne and Neuchâtel, Continuous Education Degree on Information Security, February - April 2007. (C. Bryce).

8.4. Internship supervision

We have supervised the following internships in 2006:

- Muhammad Abed (IFSIC Masters student). Subject: A Survey of Electronic Voting.
- Nadia Brahmi (University of Bordeaux Master student). Subject: Collaborative data capture in a distributed context.
- Wallid Drizet (University of Bordeaux Masters student). Subject: simulation of up-link flows in a discontinuous network.
- Salma Ben Sassi (University of Bordeaux Masters student). Subject: Access Controler discovery in a discontinuous network.

8.5. Industrial transfers

Throughout 2007, 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 as a part of ACES group planned to create a start-up company called *UbiSphere*, it get the INRIA incubator status in October 2006. This start up will be definitively created in January 2008. During 2007, due to its future business, ACES/UbiSphere has continued to develop numerous contacts/partnerships with major companies/ end-users around ambient computing technologies and the solutions provided by ACES. For example we have design and implement pilots for Aéroport de Paris (ADP), Bouygues (e-Lab) and the European Parliament (Brussels and Strasbourg).

9. Bibliography

Year Publications

Books and Monographs

[1] M. BANÂTRE, C. BRYCE, P. COUDERC, F. WEIS. *Informatique diffuse : des concepts à la réalité*, Hermes Publishing, 2007.

[2] M. BANÂTRE, A. OLLERO, A. WOLISZ, P. J. MARRON. *Cooperating embedded systems and wireless sensor networks*, Johm Wiley / Hermes Publishing, 2007.

Doctoral dissertations and Habilitation theses

[3] C. MOTSCH. *Mécanismes de gestion des flots d'exécution dans un système d'exploitation écrit en langage Java*, Ph. D. Thesis, October 2007.

Articles in refereed journals and book chapters

[4] A. OLLERO, A. WOLISZ, M. BANÂTRE. *An introduction to cooperating objects and wireless sensor networks*, John Wiley / Hermes, December 2007.

[5] S. SANTINI, K. ROEMER, P. COUDERC, P. MARRON, D. MINDER, T. VOIGT, A. VITALETTI. *System Architectures and Programming Models*, John Wiley / Hermes, December 2007.

[6] M. TLAIS, F. WEIS. *Distributed communication model in hierarchical infostations systems*, in "Concurrency and Computation: Practice and Experience", vol. 19, June 2007, p. 1183-1192.

Publications in Conferences and Workshops

[7] C. BRYCE, M. DEKKER, S. ETALLE, D. LE METAYER, S. MINIER. *Ubiquitous Privacy Protection*, in "Proc. of the 1st IEEE International Workshop on Privacy in Ubiquitous Systems, Salzburg, Austria", August 2007.

[8] X. LE BOURDON, P. COUDERC. *A protocol for distributed multimedia capture for personal communicating devices*, ACM, Roma, Italy, October 2007.

[9] J. PAUTY, P. COUDERC, M. BANÂTRE, Y. BERBERS. *Geo-Linda: a Geometry Aware Distributed Tuple Space*, in "IEEE 21st International Conference on Advanced Networking and Applications (AINA '07), Canada", IEEE, May 2007, p. 370–377.

[10] M. PAWLAK, C. BRYCE. *A Reference Model for F/OSS Process Management*, in "Free and Open Source Software Developer's European Meeting", February 2007.

[11] M. PAWLAK, C. BRYCE. *A Reference Model for F/OSS Process Management*, in "Open Source Development, Adoption and Innovation", Springer, May 2007.

[12] M. TLAIS, F. WEIS. *Data Exchanging in Hierarchical Infostation Systems*, in "2nd International Workshop on Advanced Distributed and Parallel Network Applications (ADPNA 2007), Xian, China", September 2007.

References in notes

[13] *Gaim Linux/UNIX instant messenger client*, October 2006, http://gaim.sourceforge.net/.

[14] *Method for Qualification and Selection of Open Source software (QSOS) Project*, June 2006, http://www.qsos.org/.

[15] *Sourceforge*, June 2006, http://sourceforge.net/.

[16] 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.

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

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

[19] L. CHEN, B. WASSERMANN, W. EMMERICH, H. FOSTER. *Web service orchestration with BPEL*, in "28th International Conference on Software Engineering (ICSE 2006), Shanghai, China, May 20-28, 2006", L. J. OSTERWEIL, H. D. ROMBACH, M. L. SOFFA (editors), ACM, 2006, p. 1071–1072, http://doi.acm.org/10.1145/1134511.

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

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

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

[23] 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.

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

[25] MANDRIVA. *Mandriva Linux package statistics*, Oct 2006, http://mandriva.edos-project.org/xwiki/bin/view/Packages/PackageStatistics.

[26] 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.

[27] E. S. RAYMOND. *The Cathedral and the Bazaar*, August 1998, http://www.openresources.com/documents/cathedral-bazaar/.

[28] P. SARBANES, G. OXLEY, ET AL. *Sarbanes-Oxley Act of 2002*, 2002.

[29] TRUSTED COMPUTING GROUP. *TPM Main Specification*, Main Specification, n^{o} Version 1.2 rev. 85, Trusted Computing Group, February 2005, http://www.trustedcomputinggroup.org.

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

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

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

[33] 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.

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