Providing an Embedded Software Environment for Wireless PDAs

Valérie Issarny[1], Michel Banâtre[2], Frédéric Weis[2], Gilbert Cabillic[2], Paul Couderc[2], Teresa Higuera[1], Frédéric Parain[2]

[(1)]INRIA-Rocquencourt, Domaine de Voluceau, BP 105, 78153 Le Chesnay Cédex, France

[(2)]INRIA-IRISA, Campus de Beaulieu, 35032 Rennes Cédex, France

{First.Lastname}@{inria,irisa}.fr

Position Paper

Abstract. The use of wireless PDAs is foreseen to outrun the one of PCs in the near future. However, for this to actually happen, adequate software environments must be devised in order to allow the execution of various types of applications. This paper introduces the base features of such an environment, which is a customizable JVM-based middleware. In particular, the middleware platform embeds services for appropriate resource management and for supporting novel PDA-oriented applications.

1 Introduction

The use of wireless Personal Digital Assistant (PDA) devices is foreseen to outrun the one of PCs in the near future. However, for this to actually happen, there is still the need to devise adequate software and hardware platforms. The use of PDAs should be as convenient as the one of PCs and in particular must not overly restrict the applications that are supported. Considering the ongoing effort towards providing convenient hardware platforms in industry, this paper focuses on design issues for an embedded software environment aimed at wireless PDAs, as examined within the Solidor research group[1], at INRIA-IRISA and INRIA-Rocquencourt. Two major concerns drive the design of the target software environment:

(*i*) The environment must accommodate the embedded constraints associated with PDAs. In particular, it is mandatory to finely tune the management of energy consumption and of memory.

(*ii*) The environment must enable the execution of the applications traditionally supported on the desktop, including soft real-time multimedia applications. In addition, the easy carry-on of PDAs enables provisioning new applications extending the capabilities of mobile phones. Such applications will promote the exchange of information among people as they happen to be in a nearby communication environment (e.g. based on geographical proximity and/or on the commonality of the users' interests).

Addressing the two above requirements then lies in combining solutions to the following issues:

- Devising resource management policies that are closely coupled with the resulting impact upon energy consumption.
- Delegating tasks to proxy services so as to benefit from an increased quality of service while minimizing resource consumption on the PDAs.
- Offering an open software environment, which enables both extending the set of applications that can be supported on the PDA and adapting the software environment as new applications emerge.

The literature already provides us with a number of base solutions, upon which we can build to meet the aforementioned objectives. Focusing on work from the operating system community, relevant work on the issue of energy saving for mobile computing has been examined in [3]; the proposed solution lies in energy-aware adaptation of mobile applications through collaboration between the applications and the operating system. Compared to this work, we are interested in examining coupled management of resource consumption with energy saving, including proposing a scheduler taking into account both real-time constraints and the

[1]`http://www.inria.fr/Equipes/SOLIDOR-eng.html`

available energy budget for task scheduling. Regarding the delegation of tasks to remote services, this issue has been examined from the perspective of providing a front-end proxy service, which carries out a number of computations prior to deliver the application to the thin client (e.g. see [10, 4]). We are interested in a broader cooperation between the proxy services and the PDAs so as to allow the remote execution of services whenever possible in order to minimize resource and related energy consumption on the PDA. Finally, providing an open environment subsumes adhering to some standard so as to get portable code, which can possibly be dynamically downloaded. The ideal candidate for this is Java, which appears as a major player in the area of embedded software environment. Although, Java has some shortcomings regarding the target device, these shall be solved in the near future in the light of ongoing work on extending Java to meet the requirements appertained to embedded real-time software [6] and to information appliances[2]. We are currently examining such extensions, focusing in particular on adequate solutions to energy-aware scheduling of real-time tasks and memory management.

This paper is organized as follows. Section 2 sketches the requirements that we consider as key concerns for provisioning the embedded software environment; it then presents the resulting base design decisions, which lie in providing a customizable middleware platform comprising adequate services. Section 3 discusses the middleware services on which we are currently concentrating, i.e., those for respectively handling resource scarcity and enabling novel types of applications. Finally, Section 4 offers some conclusions.

2 Base Design of the Embedded Software Environment

This section first gives an overview of the main requirements that we have identified for a software environment aimed at wireless PDAs. It then presents the resulting base design decisions for the environment.

2.1 Requirements

Requirements for a software environment aimed at wireless PDAs are of two kinds depending on whether they relate to the PDAs' intrinsic features and in particular their limited resource budget, or to the applications that need be supported for the wide acceptance of PDAs by users. We examine in turns those two sets of requirements from which we derive base guidelines for the design of the software environment.

Although the progress in hardware technology enables provisioning wireless PDAs that will be increasingly powerful, PDAs will always have less resource capabilities than conventional PCs. Most importantly, the success of PDAs will primarily lie in combining low-cost, small-size, low-weight, low-power, and long-autonomy features, which subsumes adequate hardware technology as currently examined by hardware vendors. However, such a technology must be complemented by adequate software solutions in order to not restrict the applications that can be run, possibly concurrently. Needed solutions integrate resource management for enforcing energy saving, and for coping with the limited memory resource budget.

Although wireless PDAs are not intended to fully replace conventional laptops, these are intended to complement them in a convenient way as they are far more convenient to carry. The success of PDAs will then outrun the one of PCs if PDAs enable running most of the applications that are traditionally supported on the desktop. PDAs further open opportunities for devising novel applications exploiting the fact that PDAs will always be carried by the users just like mobile phones. Such applications build upon the ubiquitous computing philosophy, which does no longer require any specific surrounding infrastructure except the possession of wireless PDAs. The main applications for the PDAs are typically those run during a trip. In this context, applications will be Internet-based for accessing both discrete and continuous multimedia data. Hence, the major constraint imposed by foreseen applications is to have adequate support for the management of soft real-time tasks. Wireless PDAs offer at least the capabilities of a mobile phone, i.e. phone communication whose software support lies in the management of hard real-time tasks. PDAs further give the opportunity of novel applications in the spirit of wearable and ubiquitous computing. Considering that users will always carry their PDA, this enables mobile users to exchange information while they happen

[2] `http://java.sun.com/j2me/`.

to physically encounter. An example of such application is given in [7], which introduces the notion of profile-based cooperation as a way to support awareness and informal communication between mobile users who are in close physical proximity. We are further investigating another kind of application that is also based on physical proximity. This is called *Spontaneous Information System* and consists of dynamically setting up a distributed system among PDAs that are physically close and only during so [2]. Supporting these novel applications requires provisioning adequate services relating to identifying the population of users that are physically close and trusted enough for the sake of information exchange.

2.2 Base Software Architecture

The previous subsection has given an overview of the requirements for the embedded software environment aimed at wireless PDAs. These may be addressed through the provision of a middleware platform, comprising services for resource management and services dedicated to applications. The main issues in offering services of the former category lie in the scheduling of real-time tasks enabling energy saving, and in customized memory management, especially considering the use of Java. For services dedicated to applications, we are in particular interested in services aimed at novel applications that depend on proximity-based interactions. In addition to the above, the environment should be coupled with remote services for delegating tasks to proxy servers in order to diminish resource consumption on the PDA whenever possible (e.g. see [10, 4]). Such services are foreseen to become commonplace. For instance, it suffices to consider the WAP[3] (Wireless Application Protocol) architecture specification, which is already exploited by service providers for the delivery of Web data to wireless PDAs.

Following the identified requirements for the software environment, we propose a JVM-based middleware (see in Figure 1) where the choice of building our environment on Java results from our concern of offering an open environment. The environment relies on the base underlying infrastructure comprising the hardware, the operating system, and base wireless communication protocols. As already raised, there is a number of hardware vendors investigating the design of convenient hardware platforms for next generation PDAs. In the specific case of our project, we will be experimenting with a multiprocessor hardware platform[4]. However, it is one of our design objectives to propose a software environment that is open enough to be used over various hardware platforms. In the same way, providing OS environments for PDAs, possibly running Java applications, is examined by a number of OS vendors (e.g. the Symbian's EPOC). Here again, we are devising our environment so as to be independent from the underlying OS. Regarding experimentation, we are currently using the WindRiver VxWorks real-time OS. Independence with the underlying infrastructure is achieved through the *wrapper* layer, which offers the API as used by the software environment. Let us notice that one of our major design concerns for the definition of the API is to leave place for optimization whenever possible. Hence, the API will embed a number of optional functions. In particular, we consider underlying operating system support for identifying the actual energy budget available. Let us further raise here that the handling of phone communication needs not be addressed within the software environment. However, the resulting usage of hardware resources must be accounted for by the scheduler. This issue is addressed by pre-reserving resources needed for handling phone communication, hence handling it as a top-level priority task. The base JVM of our environment is a JVM that we have been developing from scratch so as to be able to offer an open, middleware environment which can easily be customized with respect to the services that it embeds. The base JVM is now operational, offering the necessary functionalities for the execution of Java applications. Notice that unlike traditional JVMs, the applications run concurrently within a single JVM instance, in a way similar to the Java OS from Utah [11]. This has thus led us to integrate the necessary protection mechanisms within the base JVM.

3 Middleware Services

Considering the services offered by the middleware platform so as to enable the optimal usage of PDAs, it is clear that a large number of services may be envisioned. In the current course of our project, we

[3]`http://www.wap.net`
[4]`http://www.irisa.fr/solidor/work/scratchy.html`

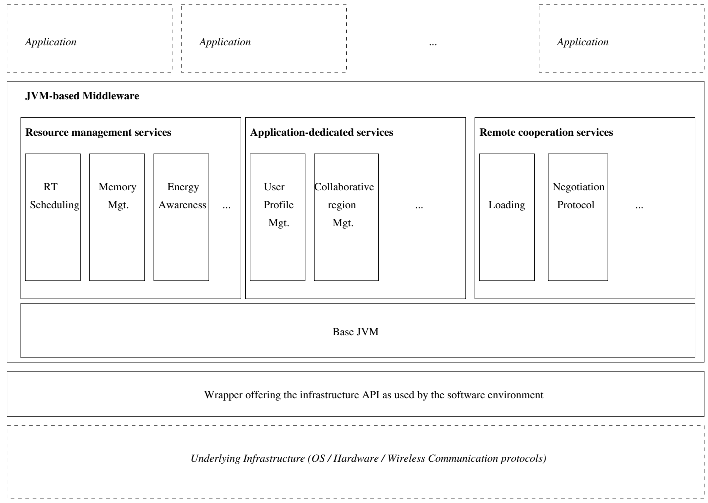

Figure 1: Architecture of the embedded software environment

are concentrating on the services for resource management, and on the services for enabling *Spontaneous Information Systems*, i.e., setting up a trusted collaborative region among users that are physically close together and only during so. These two categories of services are further discussed below.

3.1 Services for Resource Management

Regarding the scheduling of tasks within the environment, we have to address scheduling of soft real-time tasks while accounting for the associated energy consumption. The construction of real-time applications using the Java environment is already a problem (e.g. see [6]). Basically, the construction of soft real-time applications must enable identifying the adequate scheduling of embedded tasks so as to meet associated temporal constraints like deadline and ready execution time. Let us recall here that soft real-time systems differ from hard real-time ones in that deadlines can be missed. In particular, this enables provisioning a scheduler based on the empirical analysis of real-time task duration rather than on the worst-case executions. Real-time tasks making up an application can be scheduled according to either a priority-based scheme or a deadline-based scheme where the assignment of priorities in the first scheme is done according to the tasks' periods, deadlines and durations. Hence, the Java programmer must be provided with some means to express the real-time constraints associated with tasks, which typically embeds the associated periods, deadlines, and durations. In addition, the programmer must be careful regarding the usage of the Java synchronization and exception handling mechanisms. We address the above issues by providing the programmer with a number of Java classes. These serve describing a real-time application in terms of embedded real-time tasks (i.e., temporal constraints associated with the task together with the Java code to be executed upon each period) and shared resources (i.e. any resource that may be accessed by more than one real-time task). This model enables not using Java threads and synchronization mechanisms, which pose problems for the construction of real-time tasks. The proposed description of real-time applications enables deriving a scheduling graph for the real-time tasks according to their temporal constraints and to their access to shared resources. Hence, upon the arrival of a new application, the scheduling graph of the application is to be combined with the overall scheduling graph that specifies the scheduling of tasks for all the applications that are run concurrently. The application may be ultimately rejected if the scheduler cannot compute a global scheduling graph enabling to meet the temporal constraints of all the applications. In addition to embedding the temporal constraints of applications, the specification of applications further comes along with the associated energy consumption. This thus enables the scheduler to either accept or reject an incoming application according to the resulting energy consumption[5]. We are currently implementing the scheduling

[5]Our approach regarding energy-aware scheduling further lies in exploiting the underlying heterogeneous multiprocessor hardware architecture, which enables choosing the most adequate processor for executing a task with respect to both temporal

service, and further investigating solutions to the issues of exception handling (e.g. see [9]), and of fair handling of non real-time applications (e.g. see [8])

Another key issue for resource management lies in memory management, especially given the use of Java and its implicit memory reclamation mechanism. In particular, the Garbage Collector (GC) must be carefully designed to be compliant with real-time constraints [1]. We are currently designing a memory management strategy, which builds on a number of existing work. First, we rely on a distributed memory management scheme where each running application is assigned a given portion of memory upon execution. A dedicated GC is then run within each application according to the application's memory usage profile. By default, any real-time application will be combined with the GC proposed in [12]. Finally, a global real-time GC is run for reclaiming the objects created by the environment as well as those shared among applications. In addition, while there exist garbage collection algorithms that offer the necessary features for being compliant with real-time constraints, the issue of scheduling the GC-related tasks without interfering with the scheduling of real-time tasks remain. Given the profiling of the application's memory usage and the knowledge of eligible algorithms, we are currently investigating ways to couple the GC with the application by introducing the GC as a real-time task [5].

The execution of applications further relies on a negotiation protocol as commonly used when running multimedia applications. The handling of real-time constraints together with the limited capability of the PDA require to make sure that there is enough resource available to execute a new application. In the case where there is not enough resource left, it is common to have a negotiation protocol taking place between the application and the system where the application lowers its resource requirements by changing the resulting quality of service offered to the user (typically, changing from color to black-and-white for video display). We have not yet examined in detail this issue, we are currently studying it where we are interested in addressing dynamic negotiation in order to benefit from resources left by applications that terminate.

The scheduling and memory management services that we are investigating subsume some precise knowledge about the applications' behavior. However, since we are addressing soft real-time constraints, the applications' analyses may rely on profiling rather than on precise static analysis tools evaluating resource consumption in the worst case. We are currently designing profiling tools for Java so as to assess the application's behavior in terms of energy consumption, execution time, and memory usage. Such tools are not to be used on the PDA. Instead they will be offered in the development environment and the analysis results will be provided together with the application when delivered to the PDA for execution. Notice that this approach is similar to the virtual machine distribution proposed in [10] regarding the front-end handling of applications.

3.2 Services Dedicated to the Spontaneous Information System Application

In a Spontaneous Information System (SIS), the information system is distributed over the PDAs themselves. This requires from a device to be able to dynamically discover the other PDAs physically close, and the information they can eventually provide. JINI[6] addresses this issue for a slowly evolving environment, the problem of exploiting the volatile network connections of highly mobile devices (used to set up a SIS) is not considered. In particular, JINI requires the availability of a centralized lookup service. Regarding the SIS context (a set of autonomous and mobile PDAs), we have to design an information discovery mechanism without any centralized entity. At the same time, we must also guarantee information confidentiality according to a user defined policy. To reach these goals, the following issues have to be considered carefully: (*i*) designing of a lookup service satisfying the autonomy property of each PDA, and (*ii*) defining a progressive information disclosing to satisfy user confidentiality constraints.

Another key issue for SIS occurs after the information discovery phase. Users are mobile, thus unexpected disconnections are frequent (because of the limited communication range) and communication time can not be bounded. Regarding these constraints, the issue is to design an efficient communication scheme in spite of user's mobility. An approach we are currently investigating is to take into account, at the physical level, the parameters that characterize the position of a remote entity. This thus will enable to estimate

constraints and resulting energy consumption.

[6] `http://www.sun.com/jini`

periodically the "remaining communication time" between two mobile nodes. In addition we also consider the information representation within a SIS. Our objective is to design mechanisms to transfer structured information between SIS entities and stabilize them despite the highly volatile environment (user mobility and limited coverage of the communication interface). These mechanisms have to be defined in order to take benefit of the above "communication time estimation" built from environment awareness. To this end, we study carefully the following issues: (*i*) devising an "appropriate" representation of the information, and (*ii*) providing an atomic information transfer with selective degradation of data according to the estimated communication time.

4 Conclusion

Wireless PDAS are foreseen to become prominent computing devices in the near feature by combining the easy carry-on feature of mobiles phones with the processing capacity of current PCS. However, offering such PDAS to users still requires proposing adequate software and hardware environments. This paper has introduced a software environment for wireless PDAS, which lies in a customizable JVM-based middleware. Our current research interest is on devising the needed middleware services to allow the execution of various applications as diverse as multimedia applications and novel applications enabling to set up a distributed information system over the PDAS of users that are physically close. The base middleware infrastructure is now operational. We are now concentrating on the design and implementation of a subset of middleware services, i.e., services for resource management and in particular those for energy-aware real-time scheduling and memory management, and services for setting up spontaneous information systems.

References

[1] H.G. Baker. The Treadmill: Real-Time Garbage Collection Without Motion Sickness. In *Workshop on Garbage Collection in Object-Oriented Systems*. OOPSLA'91, 1991.

[2] M. Banâtre and F. Weis. A new paradigm for mobile communication systems. Information Society Technologies Conference (IST'99), November 1999.

[3] J. Flinn and M. Satyanarayanan. Energy-aware adaptation of mobile applications. In *Proceedings of SOSP'99*, pages 48–61, December 1999.

[4] A. Fox, S. D. Gribble, and Y. Chawathe. Adapting to network and client variation using active proxies: Lessons and perspectives. In *Special Issue of IEEE Personal Communications on Adaptation*, August 1998.

[5] R. Henriksson. Scheduling Real Time Garbage Collection. Technical report, Also published in Proceedings of NWPER'94, Nordic Workshop on Programing Environment Research, Lund, Sweden. ftp://www.dna.lth.se, June 1994.

[6] T. Higuera, V. Issarny, M. Banâtre, G. Cabillic, J-P. Lesot, and F. Parain. Java embedded real-time systems: An overview of existing solutions. In *Proceedings of ISORC 2000 – The 3rd IEEE International Symposium on Object-oriented Real-time Distributed Computing*, pages 392–399, March 2000.

[7] G. Kortuem, Z. Segall, and T. G. Cowan Thompson. Close encounters: Supporting mobile collaboration through interchange of user profiles. In *International Symposium on Handheld and Ubiquitous Computing (HUC'99)*, pages 171–185, 1999.

[8] J. Nieh and M. S. Lam. The design, implementation and evaluation of SMART: A scheduler for multimedia applications. In *Proceedings of SOSP'97*, pages 202–216, December 1999.

[9] K. Nilsen, S. Mitra, S. Sankaranarayanan, and V. Thanuvan. Asynchronous Java exception handling in a real-time context. In *IEEE Workshop on Programming Languages for Real-Time Industrial Applications*, December 1998.

[10] E. Gun Sirer, R. Grimm, A. J. Gregory, and B. N. Bershad. Design and implementation of a distributed virtual machine for networked computers. In *Proceedings of SOSP'99*, pages 202–216, December 1999.

[11] P. Tullmann and J. Lepreau. Nested Java Processes: OS Structure for Mobile Code. In *Eighth ACM SIGOPS European Workshop*. http://www.cs.utah.edu/projects/flux, September 1998.

[12] P.R. Wilson and M.S. Johnstone. Real-Time Non-Copying Garbage Collection. In *Workshop on Garbage Collection and Memory Management*. OOPSLA, 1993.