

Overview of the Moby Dick project

Gerard J.M. Smit, Paul J.M. Havinga, Sape J. Mullender, Arne Helme^{*}
Gunnar Hartvigsen, Terje Fallmyr, Tage Stabell-Kulø[†]
Alberto Bartoli, Gianluca Dini, Luigi Rizzo, Marco Avvenuti[‡]

Abstract--The Moby Dick project focuses on developing theories, architectures and applications for a new generation of hand-held computers. The combination of an intelligent information system and a location system enables many new types of applications, such as admission control, digital chequebook, paging, and an automatic diary that keeps track of where you were and with whom. The design challenges lie primarily in the creation of a single architecture that allows the integration of security functions, externally offered services, personality, and communication. In the architecture, Quality of Service (QoS) is no longer a networking issue alone, but a framework to model integration and integrated management of all the system services and applications in the Pocket Companion.

Index terms-- mobile computing, security, energy management, hybrid networks, data consistency

I. INTRODUCTION

In the Moby Dick project [10] we develop and define the architecture of a new generation of hand-held computers, the so-called *Pocket Companion*. With this Pocket Companion we shall be able to make telephone calls, make payments and we can use it to store all the information we now carry in our briefcases. Several factors have hampered the advance of the use of electronic information in everyday life. One of the most important has been that the information cannot always be made available where it is needed; electronic information was never very mobile. This is now about to change. Digital wireless telephony can now be used to connect very small portable computers to the internet so that, wherever one is, one's data can be accessed and interaction with others is possible. Mobile computing, assisted by wireless networking, is an essential step in making electronic document exchange, electronic communication, and electronic commerce replace at least some portion of their non-electronic counterparts. This

development will have to be accompanied by much better infrastructures for securing the privacy of one's personal data and the integrity and authenticity of financial and other transactions.

The combination of networking and mobility will engender new applications and services. Not only does it provide a means for users to stay in touch while on the move and to receive notifications of important events, it also gives people a whole new way to interact with the infrastructure of large public institutions, such as airports, supermarkets, or even whole cities. This interaction can be used for information about services, access to them and transactions with them. Standing in line for ticket or teller windows may become a thing of the past. Instead offices and public places will be equipped with access points, through which hand-held computer users will be able to communicate with the existing infrastructure.

A. *The Pocket Companion*

The technologies of PDA, digital cellular phone and, when combined and integrated well, have the potential of replacing all of the things people have to carry around with them by one small device, a Pocket Companion. This device is a small portable computer with a smart card and communications device that can replace a.o. cash, cheque book, passport, keys, diary, phone/pager and maps. The infrastructure should allow the user of the Pocket Companion to use e-mail and to access information services and data networks. The hardware basis of the Pocket Companion can be similar to the Personal Digital Assistants (PDA) on the market today, augmented with a security module and a wireless-network interface.

What makes a Pocket Companion different from a desktop machine? First of all its size and weight are small: a pocket companion has to fit into one's shirt pocket. Secondly, it is a truly personal machine: it might, for instance, contain sensitive information or store the owner's private keys or electronic money. And finally, it is a resource-poor device: there is a limited amount of energy in its batteries, there is only a small amount of memory, the CPU

^{*} University of Twente, dept. Computer Science, P.O. Box 217, 7500 AE Enschede, the Netherlands

[†] University of Tromsø, Norway

[‡] University of Pisa, Italy

processing power is restricted and the communication bandwidth is moderate. The Pocket Companion is more than just a small machine to be used by one person at a time like the traditional organizers. The Pocket Companion extends the notion of a so called 'desktop companion'. A desktop companion is a hand-held machine that is designed to give roaming users access to their business data and applications while on the road (like a Windows CE system). Desktop companions are designed and optimized for compatibility and communication with the user's desktop machine(s), e.g. via modem, infrared or a docking station.

The Pocket Companion will run applications typically found in desktop companions, but it is designed to run other applications using external public services as well. A Pocket Companion interacts with the environment and so is part of an open distributed system. It needs to communicate with - possibly hostile - external services under varying communication and operating conditions, and not only to its desktop 'master'. When Pocket Companions are incorporated into a global distributed system, their owners must be confident that the system is trustworthy.

II. PROBLEM AREAS IN MOBY DICK

The design challenges of the Moby Dick project lie primarily in the creation of a *single architecture* that allows the integration of security functions, externally offered services, personality, and communication. In this paper we will address: QoS, security, networking, data consistency and energy management.

A. The QoS framework

Applications will be used in a variety of computing environments. Some of these are distributed, and many make use of internet services. It used to be the case that applications were designed for particular computing environments, typically personal computers. But soon applications will have to run in environments that differ dramatically in processor performance, user-interface, communication performance and communication cost. Such applications will have to adapt their behaviour to the environment in which they run. Operating systems will have to provide assistance for this adaptation, which has become known as Quality of Service. The term stems from the notion that the quality of service an application can deliver depends on the resources that can be made available to it.

Originally, QoS was used in the context of network communication resources in wide-area networks. Later it was also applied to systems resources needed for multimedia applications. In Moby Dick we have extended this notion to applications in mobile-computing environments [3].

The architecture integrates QoS management into every software module, and all modules are responsible for the collection of the QoS management information they require. In the design of a module, it is important to express both the resources it needs from other modules and the adaption that is required based on what resources the module actually gets. The design of software modules for the Pocket Companion therefore focuses on cooperation and adaptation issues rather than performance.

B. Personal computing and security

When computers become more involved in people's personal and business activities security i.e. confidentiality, privacy, authentication, authenticity and non-repudiation become important concerns. Judicious application of cryptography can satisfy these concerns, provided systems provide a secure environment for users in which the appropriate cryptographic algorithms can do their work without any risk of compromising (losing) keys or confidential data. We believe that in order to use the full potential of these truly personal machines, the owner must trust his machine and he must be in full control over its machine, its information flow, and who can access it. In our view the security environment must be separated from the general-purpose computing environment. The Moby Dick security architecture allows secure signing of arbitrary contracts between mutually authenticated principals. The architecture focuses particularly on the human/computer interface. People's digital signature will not be placed without their consent and only on the document they can see. Signature keys are managed such that their owners cannot divulge them accidentally, or be lured into doing so by a malicious expert; this helps to assure non-repudiation [7,8].

We have devised a hardware architecture where the Trusted Computing Base (TCB) is limited to a security module and supported hardware (i.e., explicitly excluding software). The architecture has been designed specifically to support signing of contracts in a (possible) hostile environment, a feature we believe will be crucial for a Pocket

Companion. Execution of foreign code on your personal computer is not a new phenomenon, but additional work and experimenting is required to make it secure. We have built an experimental security framework for executing foreign programs, called helpers, on a Pocket Companion [12]. Helper programs have a profile associated with it, that specifies what files and resources will be accessed, the way they are accessed, and the capabilities of the helper. This mechanism uses a two phase approach: it checks the profile in order to make the decision whether to run the application or not, and after that it monitors the behaviour of an application.

C. Hybrid networks

Future mobile information systems will be built upon heterogeneous wireless (possibly overlapping) networks, extending traditional wired networks to hosts moving over a wide area. Distributed applications in mobile-computing environments must be prepared to deal with partitions - a mobile computer will, at times, not be able to communicate - or changes in the communication infrastructure. Although it should be possible to access all one's information via one wireless link, we must assume that most users will access their data from the type of network that is available or most convenient at any particular time. Mobile computers need to be able to move seamlessly from one communication medium to another, for example from a GSM network to an in-door network, without rebooting or restarting applications. The system must even be prepared to deal with complete disconnection from the network. Even then applications might continue, but are notified when *inconsistencies* occur. We have built a prototype implementation that allows all applications to keep their connections, while the computer switches from one network to another. The implementation of the *Software Network* enables multihomed mobile computers to switch dynamically and seamlessly from one network to another [2].

D. Data consistency

Wireless networks will become faster and more reliable than they are today, but it is unlikely that they will become as reliable as today's local-area networks. Mobile computers, therefore, will be disconnected from the network from time to time and applications will have to deal with this situation [11]. Since user data will be accessed

from different locations, sometimes concurrently (a diary may, simultaneously, be updated by a manager, her secretary and her husband), the underlying system must support distribution. However, unlike in conventional distributed systems, distributed applications in mobile-computing environments must be much better prepared to deal with partitions [1].

To give users reasonable performance over slow networks and reasonable service during network outages, storage systems must be distributed and do extensive caching. To give optimum service, distributed storage systems must be aware of the bandwidth and communication costs of the links that connect their parts. Storage systems will be a vital guardian of consistency for most applications. Partitions are possible and are expected primarily in wireless networks; when a partition occurs, file replicas (cached files) are accessible, but applications are notified when a possibility exists that inconsistencies occur. Applications must be prepared to deal with inconsistency. Simple applications, such as editors, can delegate the decision to edit a file under threat of inconsistency to the user. More complicated applications, such as a distributed diary application must implement the protocols to deal with inconsistencies and resolve them later, when the partition has been fixed.

The behaviour of an application may have to change as a consequence of network outages. The QoS manager must be aware, when making an appointment, that there is such an outage and that updates from the secretary may not have been able to reach it for a day. The user interface of a diary application must reflect such information.

E. Energy management

The Pocket Companion is a hand-held device that is resource-poor, i.e. small amount of memory, limited battery life, low processing power, and connected with the environment via a network with variable connectivity. The increasing levels of performance and integration that is required will be accompanied by increasing levels of energy consumption. Without significant energy reduction techniques and energy saving architectures, battery life constraints will limit the capabilities of a Pocket Companion [5]. Because battery life is limited and battery weight is an important factor for the size and the weight of the Pocket Companion, energy

management plays a crucial role in the architecture.

The power management system is based on three issues: system decomposition, integration of QoS management and communication. First of all, because the system architecture is decomposed of dedicated application specific subsystems that are connected with each other, energy consumption is reduced. Secondly, one of the key aspects of our QoS approach is to move power management policy decisions to the user and coordination of operations into the operating system. The operating system will control the power states of devices in the system and share this information with applications and users. Each module has its own - dedicated - *local* power management. Only the module is able to, and has the knowledge to implement the necessary power management fine-tuning of the internal functions. However, the overall power management control of the modules is done by the operating system and the user. So also here the user is in the 'control loop'. Finally, as the wireless network interface of a mobile computer consumes a significant fraction of the total power, we put considerable effort in reducing energy consumption of communication interfaces [4,6]. There are several ways to achieve this: by system decomposition, using hybrid networking, and by applying power aware MAC protocols. We have developed a MAC protocol that is suitable for real-time multimedia applications and that has low-power characteristics [9].

III. STATUS AND FUTURE DIRECTIONS

In the first phase of the project we have implemented a small set of applications to evaluate our solutions to a number of key problems and reveal the potential of the system. These demonstrators have given us a better understanding of the real world parameters connected to the research problems in the Moby Dick project. Currently we are building a testbed using off-the-shelf technology, that is several modules connected via a system-scale network. Since we envision that future hardware will be smaller with much more functionality, we anticipate on this by doing experiments with equipment that has similar behaviour - except for size. The motivation for using off-the-shelf technology, even though it does not match our idea of hand-held devices, is that it will allow us to use existing development tools (e.g. compilers, debuggers, windowing systems,

transmission software). This is essential to keep the size of the project reasonably small and to enable people to focus on the essence of the problems related to systems research and application design for Pocket Companions.

IV. REFERENCES

All publication of the Moby Dick project can also be found on <http://www.cs.utwente.nl/~havinga/mobydick.html>.

- [1] A. Bartoli: "Group-based Multicast and Dynamic Membership in Wireless Networks with Incomplete Spatial Coverage", ACM/Baltzer Journal on Mobile Networks and Applications, 1997
- [2] D. Brattli: "System support for hybrid networks", deliverable II.3.2 of the Moby Dick project, 1997.
- [3] T. Fallmyr, T. Stabell-Kulø, "QoS applied to security in mobile computing", submitted for publication
- [4] P.J.M. Havinga, G.J.M. Smit: "Minimizing energy consumption for handheld computers in Moby Dick", proceedings Euromicro 97, short contributions, pp 196-201, September 1997
- [5] P.J.M. Havinga, G.J.M. Smit: "Low Power system Design techniques for mobile computers", CTIT technical report 97-32, December 1997.
- [6] P.J.M. Havinga, G.J.M. Smit: "Minimizing energy consumption for wireless computers in Moby Dick", Proceedings 1997 IEEE International Conference on Personal Wireless Communication, December 1997
- [7] A. Helme, T. Stabell-Kulø: "Security functions for a file repository", ACM Operating Systems Review, 31(2):3-8, 1997
- [8] A. Helme: "A system for secure user-controlled electronic transactions", Ph. D. dissertation, University of Twente, 1997, ISBN 90-423-0011-6)
- [9] Linnenbank, G.R.J. et al.: "A request-TDMA multiple-access scheme for wireless multimedia networks", Proceedings Third Workshop on Mobile Multimedia Communications (MoMuC-3), 1996.
- [10] Mullender S.J., Corsini P., Hartvigsen G. "Moby Dick - The Mobile Digital Companion", LTR 20422, Annex I - Project Programme, December 1995 (see also <http://www.cs.utwente.nl/~havinga/pp.html>).
- [11] L. Rizzo: "Effective Erasure Codes for Reliable Computer Communication Protocols", ACM Computer Communication Review, Vol.27, n.2, Apr.97, pp.24-36
- [12] G.J.M. Smit, P.J.M. Havinga, D. van Os: "The Harpoon Security System for Helper Programs on a Pocket Companion", Proceedings Euromicro 97, September 1997, ISBN 0-8186-8129-2, pp 231-238