

The Pocket Companion's architecture

Paul J.M. Havinga, Gerard J.M. Smit

Abstract -- The Pocket Companion is a small personal portable computer with wireless communication facilities. The typical use of the Pocket Companion induces a number of requirements concerning security, performance, energy consumption, communication and size. The energy consumption due to the increasing demand for performance and functionality will be the limiting factor for its capabilities. Therefore reducing energy consumption plays a crucial role in the architecture. Communication, and particularly wireless communication, is essential for the system to support electronic transactions. Such a system requires a good security infrastructure not only for safeguarding personal data, but also to allow safe transactions. Index terms -- hand-held computer, energy consumption, security, communication

I. INTRODUCTION

The Moby Dick project (Esprit Long Term Research 20422) [8] develops and defines the architecture of a new generation of hand-held computers, the so-called *Pocket Companion*. It is a small personal portable computer and wireless communications device that can replace cash, cheque book, passport, keys, diary, phone, pager, maps and possibly briefcases as well. Typical applications of a Pocket Companion are diary, e-mail, note-taking and electronic payments.

The Pocket Companion is resource-poor, i.e. small amount of memory, limited battery life, low processing power, and connected with the environment via a (wireless) network with variable connectivity. The Pocket Companion is more than just a small machine to be used by one person at a time like the traditional organizers and desktop assistants. The Pocket Companion extends the notion of a 'desktop companion'. The Pocket Companion will run applications typically found in desktop companions, but it will run other applications using external public services as well.

A. Problem statement

The typical use of the Pocket Companion impels a number of requirements on the system architecture of the Pocket Companion

concerning security, performance, energy consumption and communication. In current architectures these items have been dealt often as separate or add-on items. Although these issues could be treated as separate problems, we believe that they are interrelated and can only be solved optimal when they are integrated in one single architecture.

- *Performance and energy consumption*

To support multimedia functionality for the intended applications of the Pocket Companion the system needs to have real-time properties. 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. More extensive and continuous use of network services will only aggravate this problem since communication consumes relatively much energy [7,10].

- *Communication*

Communication, and particularly wireless communication, is essential for the system to allow electronic transactions. Wireless links differ from wired networks in many aspects that have impact on the system design level. These include high and variable latency, low bandwidth, incomplete spatial coverage, energy consumption and communication cost.

- *Security*

A Pocket Companion that interacts with foreign services - under full user control and according to the owner's notion of trust and security - requires a good *security infrastructure*. The privacy of the owner's personal sensitive data has to be guaranteed and the integrity and the authenticity of transactions with the environment has to be ensured. In the Moby Dick architecture security is an integrated part of the system architecture.

An important requirement for a Pocket Companion is flexibility and programmability. In our vision, Pocket Companions will need to provide the *flexibility* to handle a variety of (multimedia) services and standards (like different video decompression schemes and security mechanisms) and the *adaptability* to accommodate to its current environment for

the changing conditions in communication connectivity, required level of security, and available resources.

II. THE POCKET COMPANION'S SYSTEM ARCHITECTURE

Over the past two decades the semiconductor technology has been continuously improved and has led to ever smaller dimensions of transistors, higher packaging density, faster circuits, and lower power dissipation. Such advances provide an effective area increase of about an order of magnitude that can be used for several goals, e.g. to increase performance, to add functionality, but also to reduce energy consumption.

The goal of the Pocket Companion's architecture is to optimize the *overall energy-performance* of the system, and not performance alone. The technology is used to *decrease energy consumption* and to *increase functionality* to provide services such as compression and decompression, network access, multimedia, and security functions.

A. Architecture

The difficulty in achieving all requirements into one architecture stems from the inherent trade-offs between flexibility and energy consumption, and also between performance and energy consumption. Flexibility requires generalized computation and communication structures, that can be used to implement different kinds of algorithms. While conventional architectures (like used in current laptops) can be programmed to perform virtually any computational task, they achieve this at the cost of high energy consumption. The Pocket Companion has a rather unconventional architecture that saves energy by using system decomposition at different levels of the architecture and exploiting locality of reference with dedicated, optimized modules. Security incurs another trade-off: in current systems a high security level can only be reached at high costs and require much performance (or patience). Our approach is based on dedicated functionality and the extensive use of power reduction techniques at all levels of system design [3]. The system has a number of premises:

- An architecture with a general purpose processor surrounded by a set of heterogeneous programmable modules, each providing an energy efficient implementation of dedicated tasks [1].

- A reconfigurable internal communication network that exploits locality of reference and eliminates wasteful data copies.
- A system design that avoids wasteful activity: e.g. by the use of *autonomous* modules that can be powered down individually and are data driven.
- A wireless communication system designed for low energy consumption by the use of *intelligent* network interfaces that deal efficiently with a mobile environment, by using a power aware network protocol, and by using an energy aware MAC protocol.
- An architecture for *signing arbitrary contracts*. The architecture can divide the system in a general computing processor and a trusted computing base (TCB) that is not user-programmable.

Figure 1 gives a schematic overview of the Pocket Companion architecture. In it, we

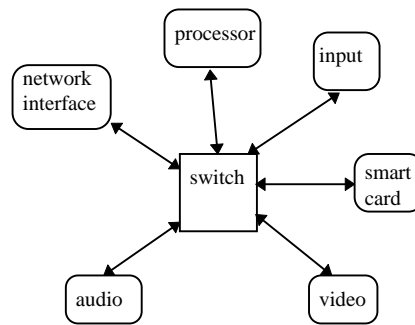


Fig. 1: Pocket Companion system architecture

distinguish a *switch* with a *security module* surrounded by several modules. The switch interconnects the modules and provides a reliable path for communication between modules. As in switching networks, the use of a multi-path topology will enable parallel data flows between different pairs of modules and thus will increase the performance. Modules are autonomous and can communicate without involvement of the main processor. Each module has its own functionality and responsibility. The modules are optimized for their function in terms of performance and energy consumption and have their own *local power management*.

B. Energy consumption and performance

The design flow of a system constitutes of various levels of abstraction. When a system is designed with the emphasis on power optimization as a performance goal, then the design must embody optimization at all levels

of the design flow. In general there are three levels on which energy reduction can be incorporated. The *system level*, the *architecture level*, and the *technological level*. As the most effective design decisions stem from the architectural and system level, a careful design at these levels can reduce the power consumption considerable. For multimedia application in particular there is a substantial reduction in energy consumption possible as the computational complexity is high, they have a regular and spatially local computation, and the communication between modules is significant. Improving the energy efficiency by exploiting *locality of reference* and using *efficient application-specific modules* therefore has a substantial impact on a system like the Pocket Companion.

C. Networking

The wireless network interface consumes a significant fraction of the total power [11]. Therefore we put considerable effort in reducing the energy consumption of the network interface.

- *MAC protocol*

We use a TDMA protocol to coordinate delivery of data to receivers [6]. The basic objective is that the protocol tries to minimize all actions of the network interface, i.e. minimize 'on-time' of the transmitter as well as the receiver. A base station is responsible for traffic scheduling. Mobiles with scheduled traffic are indicated in a list, which allows mobiles without traffic to reduce power rapidly. As switching between states (i.e. off, idle, receiving, transmitting) consumes time and energy, the number of state-transitions is minimized. By scheduling bulk data transfers, an inactive terminal is allowed to doze and power off the receiver.

Measurements have shown [11] that on typical applications like a web-browser or e-mail, the energy consumed when the interface is on and idle is more than the cost of transceiving packets. This is because the interface is generally longer idle than actually communicating. Therefore we also consider to use a very low power *transceiver for the signalling only*. This transceiver will be used for the MAC protocol and operates in parallel with the actual data-stream with an other transceiver on the same interface. This data-stream transceiver has more bandwidth and consumes more energy, but will be turned on only when there is actually data to be transmitted, and is not used for 'useless' signalling.

- *Error control*

Wireless networks have a much higher error rate than the normal wired networks. In the presence of a high packet error rate and periods of intermittent connectivity characteristics of wireless links, some network protocols (such as TCP) may overreact to packet losses, mistaking them for congestion. The limitations of TCP can be overcome by a more adequate congestion control during packet errors [2]. In Moby Dick several methods will be examined and compared. Forward Error Correction (FEC) will be used to improve the performance and to reduce energy consumption, not only at the data link level, but also in higher levels of the protocol stack [9].

- *Decomposition*

In normal systems much of the LAN protocol stack is implemented on the main processor. Thus, the network interface and the main processor must always be 'on' for the LAN to be active. Decomposition of the network protocol stack and a careful analysis the data flow in the system can reduce the energy consumption. For example, the network module can handle most of the lower levels of the protocol stack and can move the data directly to its destination, thereby allowing the main processor to sleep for extended periods of time without affecting system performance or functionality. Furthermore, part of the network protocol can be handled on another machine, e.g. the base station. For example, the mobile units can use a private lightweight protocol rather than a protocol like TCP/IP resulting in a net savings of energy.

D. Security

The heart of the security system is the Trusted Computing Base (TCB). This is the part of the system the owner of a Pocket Companion has to trust. The TCB consists of an input device (e.g. keyboard), the display, a stable storage to log transactions, a smart card, and the switch that connects these modules. A crucial property of this architecture is that the TCB is *not* user-programmable and built by a reliable manufacturer. If it does not have this property, then if the application on the main processor is corrupted, it can for example falsify data on the display, it can squirrel away the password or PIN code, and can sign other unauthorized messages as well.

A typical example of using the security function is the signing of a contract during some transaction [4]. Normally - that is when no contract signing is going on -, display and

input device are available to the local processor as its input/output device. But, when a contract has to be signed, the security module and switch disconnects display and keyboard from the main processor and takes them over for the secure signing purpose. The main processor can no longer modify the display or read the keyboard. A smart card contains, protects and applies a signature key. The card also stores the certificates that authenticates the key. When the contract is signed the main processor regains the ownership of the keyboard and display.

III. CURRENT STATUS

Currently we are designing and building a testbed for the Pocket Companion using existing hardware components and subsystems. The hardware basis of the Pocket Companion is an interconnection switch. All data in the system is based on the size of an ATM cell (48 bytes data). This not only allows us to easily interconnect with an ATM environment, but the size is also adequate: it is small enough to buffer several cells in the switch and have small latency, and large enough to keep the overhead small.

The prototype of the switch is built using a Xilinx FPGA and six small microcontrollers. The switch is capable to connect six modules. It is a tiny ATM switch as it is able to transfer ATM cells to a destination according to its VCI. The basic functions of the microcontrollers is to perform routing, to establish a connection and to interface with the connected modules. The design methodology used is data-driven and based on asynchronous methods, combined with synchronous parts. Each part of the switch that is not used at some time does not receive any data and clock changes, thus minimizing energy consumption.

The attached modules can be similar to today's PDAs, notebook computers or 'handheld PCs' augmented with a security module and one or several wireless network devices. Currently we are using the WaveLAN modem as network interface to experiment with MAC protocols and error correction. With this testbed we can easily evaluate (parts of) the architecture concerning energy consumption, security, Quality of Service and communication. We gradually improve the architecture to the final architecture.

We also experiment with system software, as hardware architecture and system software are related. The operating system being used is *Inferno* from Lucent Technologies [5]. This

system is quite well suited for experimenting with systems like the Pocket Companion.

IV. REFERENCES

- [1] Abnous A, Rabaey J.: "Ultra-Low-Power Domain-Specific Multimedia Processors," Proceedings of the IEEE VLSI Signal Processing Workshop, San Francisco, October 1996.
- [2] Balakrishnan H., et al.: "A comparison of mechanisms for improving TCP performance over wireless links", Proc. ACM SIGCOMM'96, Stanford, CA, USA, August 1996.
- [3] Havinga P.J.M., Smit G.J.M.: "Low power system design techniques for mobile computers", CTIT technical report 97-32, 1997
- [4] Helme A., Mullender S.J.: "An architecture you can trust", internal report University of Twente, March 1997.
- [5] Information on Inferno is available on <http://cruel.com/vanni/>
- [6] 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.
- [7] W. Mangione-Smith, et al.: "A low power architecture for wireless multimedia systems: lessons learned from building a power hog", Proceedings of the international symposium on low power electronics and design, August 1996
- [8] 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>).
- [9] Rizzo L.: "On the feasibility of software FEC", internal report University of Pisa, 1997, available at: <http://www.ieta.unipi.it/~luigi/fec.html>.
- [10] Sheng S., Chandrakasan A., Brodersen R.W.: "A Portable Multimedia Terminal", IEEE Communications Magazine, pp. 64-75, vol. 30, no. 12, Dec., 1992.
- [11] M. Stemm, et al. "Reducing power consumption of network interfaces in hand-held devices", Proceedings MoMuC-3, 1996.