MOBILE MULTIMEDIA SYSTEMS

Paul J.M. Havinga

ISBN 90-365-1406-1

MOBILE MULTIMEDIA SYSTEMS

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof.dr. F.A. van Vught,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 11 februari 2000 te 16.45 uur.

door

Paul Johannes Mattheus Havinga

geboren op 1 januari 1962
te Groningen

Dit proefschrift is goedgekeurd door

prof. dr. S.J. Mullender, promotor, en
dr. ir. G.J.M. Smit, assistent-promotor.

Preface

This Ph.D. thesis is the result of research in the field of mobile multimedia computing. The research was conducted as part of the MOBY DICK project carried out at the faculty of Computer Science of the University of Twente in the Netherlands. As a project-assistant working for the group Systems Programming and Computer Architecture (later renamed to Embedded Systems) I have considered it a great privilege to have the chance to obtain my Ph.D. degree.

Although mine is the only name printed on the cover of this thesis, this work would not have been possible without my colleague and assistant-advisor Gerard Smit. Many ideas presented in this thesis have resulted from joint discussions. Working towards a Ph.D. was very pleasant and seemed very natural due to his valuable coaching and collaboration throughout all the years I worked with him at the University.

I would also thank the other members of the committee for their valuable comments and suggestions on the thesis, and especially my advisor Sape Mullender for his advice and support. Furthermore, I thank Luigi Rizzo from the University of Pisa for giving me the opportunity to work in a very pleasant environment on the topic of error control. I would like to thank all my helpful colleagues, especially Ties Bos for the many inspiring discussions on various topics and Pierre Jansen for keeping me away from practical-work obligations during the last year.

I would like to thank my sons for keeping me busy with all things but my thesis, and showing me that you can do more with a computer, like watching Teletubbies. Finally, but most important, I would like to thank Josephine for supporting and encouraging me through the years, and letting me know that life is more than work alone.

Paul Havinga
Hengelo, January 2000

Table of contents

CHAPTER 2
DESIGN TECHNIQUES FOR ENERGY-EFFICIENT AND LOW-POWER SYSTEMS

CHAPTER 3
THE DESIGN OF A SYSTEM ARCHITECTURE FOR MOBILE MULTIMEDIA COMPUTERS

CHAPTER 4
THE OCTOPUS SWITCH

CHAPTER 5
ENERGY EFFICIENT WIRELESS COMMUNICATION

Abstract

Abstract

Recent advances in wireless networking technology and the exponential development of semiconductor technology have engendered a new paradigm of computing, called *personal mobile computing*. In this paradigm, the basic personal computing and communication device will be an integrated, battery-operated device, small enough to carry with you all the time. This device will be used as a replacement of many items the modern human-being carries around. However, the technological challenges to establishing this paradigm are non-trivial. In particular, these devices have limited battery resources, will handle diverse data types, and will operate in environments that are insecure, time varying, and unplanned. As the mobiles must remain usable in a wide variety of environments, they must be flexible enough to accommodate a variety of multimedia services and communication capabilities and adapt to various operating conditions in an (energy) efficient way.

The opportunities given by the continuous miniaturisation of micro-electronics are employed in the architecture of the *Mobile Digital Companion* to solve some of the problems that mobile multimedia computers encounter. We have shown that it is not sufficient to simply continue advancing our chip architectures and technologies as just more of the same: building microprocessors and devices that are simply more complicated versions of the kind built today.

The characteristics and requirements of such future handheld computers, influences many levels of the design process. Key issue in this are *energy efficiency* and *Quality of Service* (QoS). There is a vital relationship between hardware architecture, operating system architecture and applications, where each benefits from the others. Achieving high energy efficiency requires first of all the elimination of the waste that typically dominates the energy consumption in general-purpose processors. The second main principle used is to have a high locality of reference. The philosophy is that all

operations that are required on the data should be done at the place where it the most efficient, thereby also minimising the transport of data through the system.

The approach made to achieve such a system is to use autonomous, adaptable components, interconnected by a switch rather than by a bus, and to offload as much work as possible from the CPU to programmable modules that are placed in the data streams. Thus, data is delivered exactly to where it is needed, work is carried out where the data passes through – bypassing the 'main' memory –, modules are autonomously entering an energy-conservation mode and adapt themselves to the current state of the resources and the requirements of the user.

Of particular importance to the system architecture is the communication network that connects the modules. The system architecture of the *Mobile Digital Companion* is connection centric, which means that the media type of the traffic drives the data flow in the system using connections. The interconnect of the architecture is based on a switch, called *Octopus*, which interconnects a general-purpose processor, programmable (multimedia) devices (modules), and a wireless network interface. The switch is built analogous to some of the concepts that have been used in the field of ATM switching fabrics. All connections are identified with a connection identifier which is used to identify the type of data, and to determine the module destination address. This identifier provides the mechanism to support lightweight protocols that provide data-specific transport services that are associated with a certain QoS. This approach gives the system the possibility to control the QoS of a task down to the communication infrastructure.

The wireless network is another important aspect of a mobile multimedia system. We argue that energy-awareness must be applied in almost all layers of the network protocol stack. To achieve maximal performance and energy efficiency, *adaptability* is important, as wireless networks are dynamic in nature. We present an architecture of a highly adaptive network interface and a novel MAC protocol (E^2MaC) that provides support for diverse traffic types and QoS while achieving a good energy efficiency of the wireless interface of the mobile.

Samenvatting

Recente vooruitgang in technologie voor draadloze netwerken en de exponentiële ontwikkeling van semi-conductor technologie hebben een nieuw toepassingsgebied voortgebracht, gebaseerd op persoonlijke mobiele systemen. In dit toepassingsgebied zal de persoonlijke computer en het communicatieapparaat een geïntegreerde, batterij gevoed, apparaat zijn, dat klein genoeg is om altijd bij je te dragen. Het apparaat zal gebruikt worden ter vervanging van vele dingen die moderne mensen bij zich dragen. Echter, de technologische uitdagingen om dit toepassingsgebied te verwezenlijken zijn niet triviaal. In het bijzonder zullen deze apparaten een beperkte hoeveelheid batterij-energie hebben, diverse typen data behandelen, en werken in omgevingen die niet veilig zijn, variëren in tijd, en niet van tevoren voorspelbaar zijn. Omdat de apparaten bruikbaar moeten blijven in velerlei omgevingen, dienen ze flexibel genoeg te zijn om geschikt te zijn voor een groot aantal multimedia diensten en communicatie-mogelijkheden, en zich op een (energie) efficiënte manier kunnen aanpassen aan de verschillende omstandigheden.

De kansen die geboden worden door de continue miniaturisering van de micro-elektronica worden in de architectuur van de *Mobile Digital Companion* benut om een aantal problemen op te lossen waar mobiele multimedia computers op stuiten. We hebben aangetoond dat het niet afdoende is om simpelweg op dezelfde weg door te gaan met het verbeteren van de chip architecturen en technologieën als slechts meer van hetzelfde: het bouwen van microprocessoren en apparaten die slechts gecompliceerde versies zijn van het soort dat al bestaat.

De karakteristieken en eisen van toekomstige kleine ('handheld') computers, beïnvloedt diverse lagen in het ontwerpproces. Sleutelthema's hierin zijn *energie-efficiëntie* en *kwaliteit van geboden diensten*. Er bestaat een vitale relatie tussen hardware-architectuur, besturingssysteem en toepassingen, waarin elk component voordeel kan hebben van de anderen. Het bereiken van een hoge energie-efficiëntie vereist om te beginnen het vermijden van alle verspilling die in het algemeen domineert bij algemeen toepasbare processoren. Het tweede belangrijke principe is om een hoge mate van lokaliteit te bewerkstelligen. De gedachte hierachter is, dat de bewerkingen die nodig zijn op de data, plaats moeten vinden waar dat het meest efficiënt kan, waarbij tevens het datatransport door het systeem dient te worden geminimaliseerd.

De aanpak die gemaakt is om zo'n systeem te bereiken, is om autonome, aanpasbare componenten te gebruiken, die verbonden zijn door een dynamische schakelaar in plaats van een gemeenschappelijke verbinding (bus), en om zoveel mogelijk werk over te dragen van de algemene processor naar programmeerbare modulen die in de datastroom zijn geplaatst. Dus, data wordt afgeleverd precies daar waar het nodig is, werk wordt gedaan daar waar de data langs komt – zonder gebruik te maken van het hoofdgeheugen –, modulen komen autonoom in een energiezuinige modus en passen zichzelf aan de huidige toestand van de beschikbare middelen en de wensen van de gebruiker aan.

Een belangrijk aspect voor de systeemarchitectuur is het communicatienetwerk dat de modulen verbindt. De systeemarchitectuur van de *Mobile Digital Companion* is verbindings-georiënteerd, wat inhoudt dat het mediatype van het verkeer de datastroom in het systeem bepaalt, gebruik makende van verbindingen. De verbindingsstructuur is gebaseerd op een schakelaar, genaamd *Octopus*, die de verbindingen legt tussen de algemene processor, programmeerbare (multimedia) modulen, en het draadloze netwerk. De schakelaar is gebouwd naar analogie van sommige concepten die gebruikt worden in het veld van ATM-netwerk schakelsystemen. Alle verbindingen worden geïdentificeerd met een verbindings-identificatie die het type van de verbinding en de bestemmingsmodule bepaalt. Met deze identificatie kunnen lichtgewicht protocollen gemaakt worden die dataspecifieke transportdiensten leveren, en geassocieerd zijn met een bepaalde kwaliteit. Door deze aanpak heeft het systeem de mogelijkheid om de kwaliteit van een taak te besturen tot op het communicatiemedium.

Het draadloze netwerk is een ander belangrijk aspect in een mobiel multimedia systeem. We beargumenteren dat bijna alle lagen van het netwerk communicatieprotocol energiebewust moeten zijn. Om het maximale uit een systeem te halen en om energie-efficiënt te zijn, is aanpasbaarheid van belang, met name omdat draadloze netwerken van nature erg dynamisch zijn. We presenteren een architectuur van een sterk aanpasbaar netwerk interface, en een nieuw MAC protocol (E^2MaC) dat diverse verkeerstypen en kwaliteiten ondersteunt, terwijl het een goede energie-efficiëntie geeft voor de draadloze interface van het mobiele systeem.

1

Introduction

Recent advances in wireless networking technology and the exponential development of semiconductor technology have engendered a new paradigm of computing, called ***personal mobile computing*** *or* ***ubiquitous computing****. Users carrying portable devices will have access to a shared infrastructure independent of their physical location. The technological challenges to establishing this paradigm of computing are non-trivial, however. The research in this thesis is about designing such a mobile multimedia system. Personal mobile computing offers a vision of the future with a much richer and more exciting set of architecture research challenges than extrapolations of the current desktop architectures. In particular, these devices will have limited battery resources, will handle diverse data types, and will operate in environments that are insecure, dynamic and which vary significantly in time and location.*

1.1 Personal mobile computing

In recent years, technology drivers changed significantly. High-end systems used to direct the evolution of computer architectures and systems. Now low-end systems drive technology, due to their large volume and attainable profits. Advances in technology enable portable computers to be equipped with wireless interfaces, allowing networked communication even while mobile. Whereas today's notebook computers and personal digital assistants (PDAs) are self contained, tomorrow's networked mobile computers are part of a greater computing infrastructure.

Two trends – multimedia applications and mobile computing – will lead to a new application domain and market in the near future. *Personal mobile computing* (often also referred to as ubiquitous *computing* [27]) will play a significant role in driving technology in the next decade. In this paradigm, the basic personal computing and communication device will be an integrated, battery-operated device, small enough to carry along all the time. This device will be used as a replacement of many items the modern human-being carries around. It will incorporate various functions like a pager,

cellular phone, laptop computer, diary, digital camera, video game, calculator and remote control. An important issue will be the user interface: the interaction with its owner. The device will support multimedia tasks like speech recognition, video and audio.

Wireless networking greatly enhances the utility of a personal computing device. It provides mobile users with versatile communication, and permits continuous access to services and resources of the land-based network. A wireless infrastructure capable of supporting packet data and multimedia services in addition to voice will bootstrap on the success of the Internet, and in turn drive novel networked applications and services.

However, the technological challenges to establishing this paradigm of personal mobile computing are non-trivial. In particular, these devices have limited battery resources, will handle diverse data types, and will operate in environments that are insecure, unplanned, and show different characteristics in time.

In the next section we will describe the problems to be solved when designing an architecture for such a *mobile multimedia system*. In Section 1.3 we describe our approach to solve these problems, followed by a brief introduction to current systems and research on mobile multimedia devices in Section 1.4. We conclude in Section 1.5 with an overview of the thesis.

1.2 Problem statement

This dissertation is concerned about how mobile multimedia systems should be designed. The main focus is on those issues pertinent to the system design level, i.e. the area of the hardware system-designer and systems programming-designer. We will not delve into the lower level details of the VLSI realisation of the mobile system itself, nor into the higher levels of the operating system and applications.

In this section we survey the principal challenges faced in the system design of an architecture for a mobile multimedia computing device. The issues described herein divide cleanly into three items, each stemming from an essential property of mobile computing. Section 1.2.1 discusses the consequences of the required functionality on the *system architecture*, such as adaptability, multimedia functionality, wireless networking, and energy efficiency. Section 1.2.2 considers the implications of using *wireless communication* for multimedia traffic, for example susceptibility to errors and disconnections, (low) bandwidth availability, and variable network conditions. Section 1.2.3 investigates the pressure that the *energy efficiency* places on the design of a mobile system.

1.2.1 System architecture

Today, the choice of wireless devices is largely limited to simple wireless phones on the one hand, to complex and bulky laptops with wireless communication capability on the other. While these devices serve their purposes, they are neither the most integrated nor

the most general: their functionality is often limited, they can operate for just a short time, and they are incapable of fully exploiting the emerging integrated wireless networks.

Even while current devices have the ability to communicate and process data, they are and by large primarily either data processing devices *or* communication devices. Simply shrinking the processing devices and communication devices, and packaging them together does not alleviate the architectural bottlenecks of integrated mobile multimedia devices [19]. The real challenge is to design a device where data processing and communication share equal importance.

Future wireless devices must meet five major requirements: high performance for multimedia functions, energy efficiency, small size, low design complexity, and a very intuitive and small user interface.

Multimedia functionality is a driving force for many research challenges. For example, due to the size constraints on a portable computer, the user interface must be small. This is a main reason that pens have become the standard input devices for PDAs. The shortage of area on a mobile device can cause us to trade buttons in favour of recognising the user's intention from analog input devices such as handwriting, gesture [4] and voice. Speech generation and recognition seem an ideal user interface since they require no surface area and allow hands-free and eye-free operation. However, general-purpose speech input and output places substantial storage and processing demands on a mobile device. Other research investigates the use of head-mounted virtual reality displays [26]. Main problems to be solved are the required processing power or communication bandwidth and the required weight and size (i.e. a small and light headgear).

A key challenge of mobile computing is that many attributes of the environment vary dynamically. Mobile devices face many different types of variability in their environment. Therefore, they need to be able to operate in environments that can change drastically in short term as well as long term in available resources and available services. Some short-term variations can be handled by adaptive communication protocols that vary their parameters according to the current condition. Other, more long-term variations generally require a much larger degree of adaptation. Merely algorithmic adaptations are not sufficient, but rather an entirely new set of protocols and/or algorithms may be required. For example, mobile users may encounter a complete different wireless communication infrastructure when walking from their office to the street. They might require another air interface, other network protocols, and so forth. A possible solution is to have a mobile device with a reconfigurable architecture so that it can adapt its operation to the current environment and operating condition.

Reconfigurability also has another more economic motivation: it will be important to have a fast track from sparkling ideas to the final design. If the design process takes too long, the return on investment will be less. It would further be desirable for a wireless terminal to have architectural reconfigurability whereby its capabilities may be modified by downloading new functions from network servers. Such reconfigurability would also

help in field upgrading as new communication protocols or standards are deployed, and in implementing bug fixes [19]. This also asks for a flexible architecture with a reasonable amount of programmability [20]. One of the key issues in the design of portable multimedia systems is to find a good balance between flexibility and high-processing power on one side, and area and energy-efficiency of the implementation on the other side.

1.2.2 Wireless communication

Mobile computers require wireless network access, although sometimes they may physically attach to the network for a better or cheaper connection. Wireless communication is much more difficult to achieve than wired communication because the surrounding environment interacts with the signal, blocking signal paths and introducing noise and echoes. As a result wireless connections have a lower quality than wired connections: lower bandwidth, less connection stability, higher error rates, and, moreover, with a highly varying quality. These factors can in turn increase communication latency due to retransmissions, can give largely varying throughput, and incur a high energy consumption. Three key problems in networked wireless multimedia systems are 1) the need to maintain quality of service (throughput, delay, bit error rate, etc) over time-varying channels, 2) to operate with limited energy resources, and 3) to operate in a heterogeneous environment.

Quality of Service – Considerations of energy efficiency are fundamentally influenced by the trade-off between energy consumption and achievable Quality of Service (QoS). To deal with the dynamic variations in networking and computing resources gracefully, both the mobile computing environment and the applications that operate in such an environment need to adapt their behaviour depending on the available resources including the batteries.

Energy-efficiency – The wireless network interface of a mobile computer consumes a significant fraction of the total energy of a mobile computer. More extensive and continuous use of network services will aggravate this problem. Energy efficiency can be improved at various layers of the communication protocol stack. Adaptability of the protocols is a key issue is achieving this.

Heterogeneity – In contrast to most stationary computers, mobile computers encounter more heterogeneous network connections. As they leave the range of one network transceiver they switch to another. In different places they may experience different network qualities. There may be places where they can access multiple transceivers, or even may concurrently use wired access. The interface may also need to change access protocols for different networks, for example when switching from wireless LAN coverage in an office to cellular coverage in a city. This heterogeneity makes mobile computing more complex than traditional networking.

In this thesis we concentrate on the problems related to Quality of Service and energy-efficiency.

1.2.3 Energy efficiency

Although the subject of low-power consumption of integrated circuits (ICs) is drawing considerable attention ("cool chips are hot"), this interest is only of recent date. There are several motivations for energy-efficient design. Perhaps the most visible driving source is the success and growth of the portable consumer electronic market. Today's desktop computers are not intended to be carried, so their design is liberal in their use of space, weight, energy consumption, noise, cabling, and heat dissipation. In contrast, the designer of a hand-held mobile computer should strive for the properties of a wristwatch: light, small, durable and long battery life.

Batteries are the largest single source of weight in a portable computer. Minimising energy consumption can improve portability by reducing battery weight and lengthening the life of a charge. Moreover, the functionality of the mobile computer is limited by the required energy consumption for communication and computation. Unfortunately, the rate at which battery performance improves (in terms of available energy per unit size or weight) is fairly slow, despite the great interest generated by the booming wireless business. Aside from major breakthroughs it is doubtful that significant reduction of battery size and weight can be expected in the near future [21]. It has been generally expected that the battery technology alone will not solve the low-power problem. It therefore makes sense to look for alternative strategies for energy savings and energy management. The emerge of various applications and the need to support them in a wireless setting may open new possibilities for energy-saving strategies.

Remarkably, high performance computing systems also drives the low power needs. The power dissipation of high performance microprocessors is now already several dozen Watts, comparable to that of a hand-held soldering iron [25]. The *cost* associated with packaging and cooling such devices is becoming prohibitive.

In addition to cost, there is the issue of *reliability*. High power systems tend to increase the silicon temperature, and high temperature tends to exacerbate several silicon failure mechanisms. Every 10°C increase in operating temperature roughly doubles a component's failure rate [17].

Another major demand for low-power systems comes from *environmental concerns*. Computers are the fastest-growing electricity loads in the commercial sector [25]. Since electricity generation is a major source of pollution, inefficient energy usage in computing equipment indirectly contributes to environmental pollution.

Finally, a fraction of the consumed energy is radiated into space, possibly affecting other electronic equipment (Electro-Magnetic Compatibility, or EMC) [2].

The way out is *energy efficiency:* doing more work with the same amount of energy. Traditionally, energy efficiency has been focussed on low-power techniques for VLSI design. However, the key to energy efficiency in future mobile multimedia devices will be at the higher levels: energy-efficient system architectures, energy-efficient communication protocols, energy-cognisant operating system and applications, and a well designed partitioning of functions between wireless device and services on the network.

A major challenge in achieving this will be that many attributes of the system environment can vary drastically by several orders of magnitude over the short and long term. Key to these issues will be *adaptability*. Research has shown that continually adapting the system and protocols can significantly improve the energy efficiency while maintaining a satisfactory level of performance [21]. Adapting to the variability is the shared responsibility of many layers in the system design of the mobile device, including the applications.

1.2.4 Hypothesis

Energy efficiency and Quality of Service will be very important for mobile multimedia systems. In this dissertation the following two hypothesises are made:

1. The key to energy efficiency will be achieved in the design of the higher layers of the system, its system architecture, its functionality, its operating system, and the entire network. Of special importance in this are the communication channels.
2. Quality of Service is an important mechanism for mobile multimedia systems not only to give users an adequate level of service, but also as a tool to achieve an energy efficient system.

1.3 Approach

The research presented in this thesis addresses the design of an architecture for a mobile multimedia handheld computer that can cope with the requirements and difficulties mentioned above. The main focus is on the specification of a system architecture supporting the required functions for future handheld devices.

The approach made in our research was to study *practical* solutions to the inherent problems of handheld multimedia terminals. In this field too often, system architectures, protocols, and applications are developed with a theoretical background only and with a limited scope covering one horizontal layer in a system. In contrast, this research is characterised by a strategy that traverses *vertically* through various layers of the system architecture of a multimedia hand-held system and is driven by energy-efficient design considerations.

While low-level circuit and logic techniques have been well established for improving energy efficiency, they do not hold promise for much additional gain. As the issue of energy efficiency becomes even more pervasive, the battle to use the bare minimum of energy will be fought on multiple fronts: semiconductor technology, circuit design, design automation tools, system architecture, operating system, and application design. The key to energy efficiency in future mobile systems will be designing higher layers of the mobile system, its system architecture, its functionality, its operating system, and indeed the entire network, with energy efficiency in mind.

In its most abstract form, a networked computer system has two sources of energy drain required for operation:

- *Communication*, due to energy spent by the wireless interface and due to the internal traffic between various parts of the system, and
- *Computation*, due to processing for applications, the operating system, and tasks required during communication.

Broadly speaking, minimising energy consumption is a task that will require minimising the contributions of communication and computation, making the appropriate trade-offs between the two. For example, reducing the amount of transmitted data may be beneficial. On the other hand, the computation cost (e.g. to compress the data being sent) might be high, and in the extreme it might be such that it would be better to just send the raw data.

In this thesis we will concentrate on the *communication channels* rather than the computational elements. The communication channels contribute a significant amount of the total energy consumption of a typical mobile system. This property also holds for multimedia applications, even though these applications typically require a significant computational effort as well. This is for a significant part due to the limitations of most current hardware and operating systems that are unable to differentiate between various traffic streams. A general theme in this thesis is to reduce the amount of communication and avoid 'useless' and inefficient computation, which consequently reduces energy dissipation and increases performance of the system.

Specific contributions of the research described in this thesis are the design of an energy-efficient architecture for mobile multimedia systems and a reconfigurable connection switch, as well as the design of crucial wireless network functions (i.e. MAC protocol, adaptable network interface, and a model for adaptable error-correction) that are energy efficient and can support multimedia traffic.

1.3.1 System architecture

The traditional architecture of a mobile is centered around a general-purpose processor with local memory and a shared-bus that connects peripherals to the CPU. However, in such an architecture several problem areas can be identified. Main problem areas are energy consumption, performance, and Quality of Service guarantees.

A large fraction of system time and power budget in a shared bus architecture is devoted to bus transactions. Busses are significant sources of power dissipation due to high switching activities and large capacitance. This architecture requires frequent traversal of multimedia streams over the bus and through the layers of the operating system software, and possibly also through to a network protocol stack which is composed of transport, network, link and medium access (MAC), and physical layer protocols. Typical functions in the network protocol stack include routing, congestion control, error control, resource reservation, scheduling, etc. Instead of arithmetic functions like additions and multiplications, the primary importance in the system is processing of the protocols.

Current systems based on a shared bus architecture are able to deliver the required performance for various multimedia applications not only by using the rapid advance in technology, but also by careful design and use of the interface modules. The process to achieve this requires a huge amount of effort of both the hardware designer of the I/O interfaces and the system designer. There are many subtle device issues that can influence the overall I/O performance of a system. Minor changes in the hardware or software configuration can have severe consequences for the performance of the system. The reason for these problems are often caused by the interconnect and the interconnection protocols. Since a shared bus cannot give QoS guarantees, a single device or application can reduce the throughput that is available for all devices.

By designing a *connection-centric* architecture that moves processing power closer to the data stream, it is possible to solve these problems. The whole system is based on connections between modules. Each connection is associated with a certain QoS. This approach is especially well suited for continuous media data (e.g. audio, video, etc.), where the processing is actually of a very specialised nature (e.g. signal processing, compression, encryption, etc.) and needs to be carried out in real-time. In contrast to memory-centric (or CPU-centric) systems, a connection-centric system is decomposed out of application-specific modules. In such a system the data traffic is reduced, mainly because unnecessary data copies are removed. For example, in a system where a stream of video data is to be displayed on a screen, the data can be copied directly to the screen memory, without going through the main processor. The CPU is thus moved out of the data flow datapath, although it still participates in the control flow. The role of the CPU is reduced to a controller that initialises the system and handles complex protocol processing that are most easily implemented in software.

The approach used in our research to achieve a system as described above is to have autonomous, reconfigurable modules such as network, video and audio devices, interconnected by a switch rather than by a bus, and to offload as much as work as possible from the CPU to modules placed in the data streams. Thus, communication between components is not broadcast over a bus but delivered exactly where it is needed, work is carried out where the data passes through, bypassing the memory. To limit the communication overhead and the required buffering, the granularity of the tasks on the devices is rather coarse, and the application is partitioned in large blocks. The programmability of each module is more fine-grained and is controlled by the individual autonomous module.

The interconnect of the architecture is based on a reconfigurable communication network switch, called *Octopus*, which interconnects a general-purpose processor, (multimedia) devices, and a wireless network interface. Conceptually, the architecture is analogous to a self-routing packet switch. The connection-oriented approach using fixed sized cells and the asynchronous multiplexing are key factors. This not only eliminates the need to transfer a large number of address bits per access, it also gives the system the possibility to control the QoS of a task down to the communication infrastructure. All connections are identified with a connection identifier, which is used to identify the type of data, and the module destination address. This identifier provides the mechanism to support lightweight protocols that provide data-specific transport services that are

associated with a certain QoS. This is an important requirement since in a QoS architecture all system components, hardware as well as software, have to be covered end-to-end along the way from the source to the destination.

1.3.2 Wireless communication

Another important aspect in mobile multimedia systems is wireless communication protocols that provide multimedia services to mobile users. Multimedia applications are characterised by their various media streams. Each stream can have different quality of service requirements. Depending on the service class and QoS of a connection a different policy can be applied by the communication protocol in order to minimise energy consumption. For example, by avoiding error-control overhead for connections that do not need it and by never transmitting stale data, efficiency is improved. This combination of limited bandwidth, high error rates, and delay-sensitive data requires tight integration of all subsystems in the device, including aggressive optimisation of the protocols that suits the intended application. The protocols must be robust in the presence of errors and they must be able to differentiate between classes of data, giving each class the exact service it requires.

The access to the wireless channel is controlled by data link protocols. Many protocols for wireless networks are basically adaptations of protocols used in wired networks, and ignore energy issues. A first step in improving the energy efficiency of the wireless network protocols is to *eliminate useless activity* of the wireless interface. There are various reasons for this useless activity. It has been shown that for typical applications like a web-browser or e-mail, the energy consumed while the interface is ‘on’ and idle is more than the cost of actually receiving packets. That is because most applications have little demanding traffic needs, and hence the transceiver is *idling* most of the time. Furthermore, in a typical wireless broadcast environment, the receiver has to be powered ‘on’ at all times to be able to receive messages from the base station, resulting in significant energy consumption. The receiver subsystem typically receives all packets and forwards only the packets destined for this mobile. Another cause is due to the *inactivity threshold*, which is the time before a transceiver will go in the ‘off’ or ‘standby’ state after a period of inactivity, which causes the receiver to be in an energy consuming mode needlessly for a significant time. Significant time and energy is further spent by the mobile in switching from transmit to receive modes, and vice-versa.

The next step is to *reduce the amount of data*, which must be pushed through the channel. This goal can be reached in a number of ways. One is to reduce the *overhead of a protocol* which influences the energy requirements due to the amount of ‘useless’ control data and the required computation for protocol handling. Another step is to avoid *collisions* that typically may occur in broadcast networks. This causes the data to become useless and the energy needed to transport that data to be lost. The high *error rate* that is typical for wireless links is another source of energy consumption for several reasons. First, when the data is not correctly received the energy that was needed to transport and process that data is spoiled. Second, energy is used for error control mechanisms. This includes energy spent in the physical radio transmission process, as well as energy spent in computation, such as signal processing and error control at the

transmitter and the receiver. Finally, because in wireless communication the error rate varies dynamically over time and space, a fixed-point error control mechanism that is designed to be able to correct errors that hardly occur, spoils energy and bandwidth. If the application is error-resilient, trying to withstand all possible errors spoils even more energy for needless error control. Reducing the amount of data is also an application-layer issue. For example, the application might change the compression rate or possibly reduce the data resolution. Instead of sending an entire large full-colour image, one can send black-and-white half-size images with lossy compression.

The goals of low energy consumption and the required support for multiple traffic types lead to the communication system described in this thesis that is based on reservation and scheduling strategies. For each connection a different set of parameters concerning scheduling, flow control and error control is applied. The wireless network is composed of several base-stations that each handle a single radio cell possibly covering several mobile stations. The base-station controls access on the wireless channel based on communication requests for connections of the mobiles by dividing bandwidth into transmission slots. The key to providing QoS for these connections and the energy efficiency of the mobiles will be the scheduling algorithm that assigns the bandwidth. The premise is that the base-station has virtually no processing and energy limitations, and will perform actions in courtesy of the mobile. The main principles are: avoid unsuccessful actions by avoiding collisions and by providing provisions for adaptive error control, minimise the number of transitions by scheduling traffic in larger packets, synchronise the mobile and the base-station which allows the mobile to power-on precisely when needed, and migrate as much as possible work to the base-station.

1.4 Related work

The growing popularity of mobile systems has spawned much interest and research by industry and universities in both computer science and electrical engineering. Most of the current research, however, often tackles just one horizontal layer of the design. Although this research is valuable, and must be applied whenever suitable, we will provide here merely a brief overview of those systems and current research that look into the problem of designing a mobile multimedia device in an integrated fashion. We do not include the wireless devices that are on the market today, because, as discussed above, they are somewhere in the spectrum between portable computers with a wireless interface and a wireless phone. They fail to address the specific requirements and fail to exploit the possibilities offered by multimedia communication. The related research of the specific issues of the mobile computer system architecture (like interconnection structures, medium access and data link protocols) is reviewed in the corresponding chapters in this thesis.

Currently, there is a broad consensus that the existing mobile devices are by far not capable of supporting the required multimedia functionality. Some reasons are: processing power, energy consumption, communication bandwidth requirements, etc.

About the solution to solve this problem there is much less consensus, however. Within the notion of mobile computing, there is considerable latitude regarding the role of the portable device. Is it a terminal or an independent, stand-alone computer? How many purposes shall the device serve? Many different architectural choices are possible, each with a different partitioning of functions between the wireless device and remote servers. These design choices greatly affect the issues mentioned in this chapter.

Several architectures have been proposed that address mobile multimedia computing. Only few systems address energy reduction. Systems like the *InfoPad* [24] and *ParcTab* [6] are designed to take advantage of high-speed wireless networking to reduce the amount of computation required on the portable. These systems are portable terminals and take advantage of the processing power of remote compute servers. No local computation, except for appropriate coding/decoding of the I/O data, is done at the pad. Such devices are known as *thin clients*, since the client itself does little work. This approach simplifies the design and reduces power consumption for the processing components, but significantly increases the network usage and thus potentially increases energy consumption because the network interface is energy expensive. These systems also rely on the availability of a high bandwidth network connectivity and cannot be used when not connected.

In the Ubiquitous Communications project (*Ubicom*) [26] at the Delft University of Technology the clients also depend heavily on the wireless communication network. This project aims at developing a campus-wide system for wireless communication that is capable of supporting multimedia applications. The target is a visual geographic information system that uses augmented reality techniques to display information on a mobile user's headset; information is super-imposed on the user's view using a retinal scanning display. To minimise energy consumption in the mobile unit, the main processing power is located in the backbone network.

The *Merlin* project of the University of California at Los Angeles (UCLA) [21][18] is developing mobile computing and wireless communication technologies with the focus on creating a wireless I/O-network subsystem that can be used to create many different types of wireless connected multimedia nodes: handheld computes, wireless cameras, wireless IP phones, etc. The subsystem is composed of a wireless network processor, codecs, and radio to provide all the necessary wireless networking and multimedia processing capabilities. In the architecture of *WAND*, a low-power embeddable module built at UCLA for creating multimedia wireless terminals, the general-purpose processor is moved out of the packet flow data path, and the data streams flow directly between the radio and the speech and image codecs. A full-fledged PC or PDA may be adjunct to WAND, but its presence is optional and, in many wireless terminals unnecessary.

Other research is mainly concentrated on specific topics, and is not covering the system architecture of a mobile computer as a whole. There is much research on multimedia processors, hardware accelerators, and heterogeneous multiprocessor architectures mainly targeted for DSP algorithms (e.g. [1][18][20]). In recent years much research has been done in providing QoS over a wireless link. Access protocols for these systems typically only address network performance metrics such as throughput, efficiency, and packet delay. However, thus far, little attention is given to energy conserving protocols,

and researchers mainly focuses their effort on energy reduction by circuit design. Very recently there is a growing interest in energy-efficient design, although mainly concentrating on medium access and link-layer energy reduction techniques. Chapters 5 and 6 give more details on this research.

1.5 Thesis overview

This thesis is divided into 6 chapters. This chapter has presented a survey of the principal challenges faced when designing a mobile multimedia handheld computer. It presented a motivation for and introduction to the low-power methodologies and systems that will be presented in subsequent chapters. The following chapters are largely based on papers presented at conferences and published in journals. The structure of the thesis is guided along these papers.

Chapter 2 will describe low-power design techniques at all levels ranging from process technology to applications, and will motivate the need for a vertical system-wide energy-efficient design approach. The chapter does not aim to be a complete presentation in the field, but is instead focussed on issues of relevance to the discussion in other chapters of the dissertation.

The chapter is based on papers [6], [7] and [9]:

- "Design techniques for low power systems", Havinga P.J.M., Smit G.J.M., *Journal of Systems Architecture*, Vol. 46, Issue 1, 2000.
- "Minimizing energy consumption for wireless computers in Moby Dick", Havinga, P.J.M., Smit, G.J.M., *Proceedings IEEE International Conference on Personal Wireless Communication (ICPWC'97)*, pp. 306-310, December 1997.
- "Minimizing energy consumption for handheld computers in Moby Dick", Havinga P.J.M., Smit G.J.M., *Proceedings of the 23rd Euromicro Conference 97*, pp. 196-201, September 1997.

Chapter 3 addresses fundamental issues in the architecture, design and implementation of low-power multimedia hand-held computers, with particular emphasis on energy conservation. This chapter introduces the system architecture of the portable computer that is topic of our research, called *Mobile Digital Companion*, which provides support for handling multimedia applications efficiently. The Mobile Digital Companion saves energy by using system decomposition at different levels of the architecture and exploits locality of reference with dedicated, optimised modules. The approach is based on dedicated functionality and the extensive use of energy reduction techniques at all levels of system design. The system has an architecture with a general-purpose processor accompanied by a set of heterogeneous autonomous programmable modules, each providing an energy efficient implementation of dedicated tasks.

Chapter 3 is based on papers [8] and [22]:

- "The Pocket Companion's architecture", Havinga P.J.M., Smit G.J.M., *1st Euromicro summer school on mobile computing '98*, pp. 25-34, Oulu, August 1998.
- "An overview of the Moby Dick project", Smit G.J.M., Havinga P.J.M., et al., *1st Euromicro summer school on mobile computing '98*, pp. 159-168, Oulu, August 1998.

Chapter 4 presents the reconfigurable internal communication network switch, called *Octopus*. The switch is implemented as a simplified ATM switch and provides Quality of Service guarantees and enough bandwidth for multimedia applications found in a handheld computer. We have built a testbed of the architecture, of which we will present performance and energy consumption characteristics.

Chapter 4 is based on papers [11] and [14]:

- "Octopus: embracing the energy efficiency of handheld multimedia computers" , Havinga P.J.M., Smit G.J.M., *Proceedings fifth annual ACM/IEEE international conference on mobile computing and networking (Mobicom'99),* pp. 77-87, August 1999.
- "Octopus – an energy-efficient architecture for wireless multimedia systems", Havinga P.J.M., Smit G.J.M., *Proceedings ProRISC workshop on Circuits, Systems and Signal Processing (ProRISC'99)*, pp. 185-192, November 1999.

In Chapter 5 we delve into the problems related to wireless communication. We present an energy-efficient highly adaptive architecture of a network interface and novel data link layer protocol for wireless networks that provides Quality of Service (QoS) support for diverse traffic types. In our approach we apply adaptability through all layers of the protocol stack, and provide feedback to the applications. In this way the applications can adapt the data streams, and the network protocols can adapt the communication parameters. Since high error rates are inevitable to the wireless environment, *energy-efficient error control* is an important issue for mobile computing systems. We therefore investigate the energy efficiency of error-control mechanisms that can be used to build adaptive error-control schemes.

Chapter 5 is based on papers [10], [12] and [13]:

- "Energy efficient wireless ATM design", Havinga P.J.M., Smit G.J.M., Bos M., to appear in *ACM/Baltzer Journal on Mobile Networks and Applications (MONET), Special issue on Wireless Mobile ATM technologies,* Vol. 5, No 2., 2000.
- "Energy efficiency of error correction on wireless systems", Havinga P.J.M., *Proceedings IEEE Wireless Communications and Networking Conference (WCNC'99*), September 1999.
- "Energy efficient wireless ATM design", Havinga P.J.M., Smit G.J.M., Bos M., *Proceedings second IEEE international workshop on wireless mobile ATM implementations (wmATM'99)*, pp. 11-22, June 1999.

To conclude, Chapter 6 gives a brief evaluation, provides directions for future research, and gives some general conclusions.

References

[1] Abnous A., Seno K., Ichikawa Y., Wan M., Rabaey J.: "Evaluation of a low-power reconfigurable DSP architecture", *proceedings 5th Reconfigurable Architectures workshop (RAW'98)*, March 30, 1998, Orlando, USA. (URL: http://xputers.informatik.uni-kl.de/RAW/RAW98/adv_prg_RAW98.html)

[2] Berkel K. van, Rem M.: "VLSI programming of asynchronous circuits for low power", *Nat.Lab. Technical Note Nr. UR 005/94*, Philips Research Laboratories, Eindhoven, the Netherlands, 1994.

[3] Black A., Inouye J.: "System support for mobility", proceedings 1996 SIGOPS European Workshop, pp.129-132, 1996.

[4] "Rock 'n' Scroll – Button-free Tilt and Gesture Input for Itsy", http://www.research.digital.com/wrl/projects/RocknScroll/RocknScrollOverview.htm.

[5] Forman G.H.: "The challenges of mobile computing", UW CSE Tech report # 93-11-03, ftp.cs.washington.edu.

[6] Havinga, P.J.M., Smit, G.J.M.: "Minimizing energy consumption for wireless computers in Moby Dick", *proceedings IEEE International Conference on Personal Wireless Communication ICPWC'97*, pp. 306-310, December 1997.

[7] Havinga P.J.M., Smit G.J.M.: "Minimizing energy consumption for handheld computers in Moby Dick", *Proceedings of the 23rd Euromicro Conference 97*, pp. 196-201, September 1997.

[8] Havinga P.J.M., Smit G.J.M.: "The Pocket Companion's architecture", *Euromicro summer school on mobile computing '98*, Oulu, pp. 25-34, August 1998

[9] Havinga P.J.M., Smit G.J.M.: "Design techniques for low power systems" *Journal of Systems Architecture*, Vol. 46, Iss. 1, 2000, a previous version appeared as CTIT Technical report, No. 97-32, Enschede, the Netherlands, ISSN 1381-3625

[10] Havinga P.J.M., Smit G.J.M., Bos M.: "Energy efficient wireless ATM design", *proceedings second IEEE international workshop on wireless mobile ATM implementations (wmATM'99)*, pp. 11-22, June 1999.

[11] Havinga P.J.M., Smit G.J.M.: "Octopus: embracing the energy efficiency of handheld multimedia computers" , *proceedings fifth annual ACM/IEEE international conference on mobile computing and networking (Mobicom'99),* pp.77-87, August 1999.

[12] Havinga P.J.M.: "Energy efficiency of error correction on wireless systems", *proceedings IEEE Wireless Communications and Networking Conference (WCNC'99)*, September 1999.

[13] Havinga P.J.M., Smit G.J.M., Bos M.: "Energy efficient wireless ATM design", to appear in *ACM/Baltzer Journal on Mobile Networks and Applications (MONET), Special issue on Wireless Mobile ATM technologies,* Vol. 5, No 2., 2000.

[14] Havinga P.J.M., Smit G.J.M.: "Octopus – an energy-efficient architecture for wireless multimedia systems", *Proceedings Program for Research on Integrated Systems and Circuits (ProRISC'99)*, pp. 185-192, November 1999.

[15] Kantarjiev C. et al.: “Experiences with X in a wireless environment”, *Mobile and location-independent computing symposium*, Cambridge MA, August 1993.

[16] Kozyrakis C.E., Patterson D.A.: “A new direction for computer architecture research”, *Computer*, Nov. 1998, pp. 24-32.

[17] Landman P.E.: “Low-power architectural design methodologies”, Ph.D. thesis, *University of California at Berkeley*, 1994.

[18] Leijten J.A.J.: “Real-time constrained reconfigurable communication between embedded processors”, *Ph.D. thesis, Eindhoven University of Technology*, November 1998.

[19] Lettieri P., Srivastava M.B.: “Advances in wireless terminals”, *IEEE Personal Communications*, pp. 6-19, February 1999.

[20] Nieuwland A.K., Lippens P.E.R.: “A heterogeneous HW-SW architecture for hand-held multi-media terminals”, *proceedings IEEE workshop on Signal Processing Systems*, SiPS’98, pp. 113-122, 1998.

[21] Sheng S., Chandrakasan A., Brodersen R.W.: “A Portable Multimedia Terminal”, *IEEE Communications Magazine*, pp. 64-75, vol. 30, no. 12, Dec., 1992.

[22] Smit G.J.M., Havinga P.J.M., et al.: “An overview of the Moby Dick project”, *1st Euromicro summer school on mobile computing*, pp. 159-168, Oulu, August 1998.

[23] Srivastava M.: “Design and optimization of networked wireless information systems”, *IEEE VLSI workshop*, April 1998.

[24] Truman T.E., Pering T., Doering R., Brodersen R.W.: The InfoPad multimedia terminal: a portable device for wireless information access”, *IEEE transactions on computers*, Vol. 47, No. 10, pp. 1073-1087, October 1998.

[25] Yeap G.K.: “Practical low power digital VLSI design”, *Kluwer Academic Publishers*, ISBN 0-7923-80.

[26] Toetenel H.: “The ubiquitous communication program”, *Euromicro summer school on mobile computing ’98*, Oulu, pp. 181-189, August 1998, http://ubicom.twi.tudelft.nl.

[27] Weiser M.: “Some computer science issues in ubiquitous computing”, *Communications of the ACM*, 36(7):75-84, July 1993.

2

Design techniques for energy efficient and low-power systems

Portable systems are being used increasingly. Because these systems are battery powered, reducing energy consumption is vital. In this chapter we give an overview of low-power design and provide a review of techniques to exploit them in the architecture of the system. We focus on: minimising capacitance, avoiding unnecessary and wasteful activity, and reducing voltage and frequency. We review energy reduction techniques with applications in the architecture and design of a hand-held computer including its wireless communication system.

2.1 Introduction

The portability requirement of hand-held computers and other portable devices places severe restrictions on size and power consumption. Even though battery technology is improving continuously and processors and displays are rapidly improving in terms of power consumption, battery life and battery weight are issues that will have a marked influence on how hand-held computers can be used. These devices often require real-time processing capabilities, and thus demand high throughput. Power consumption is becoming the limiting factor in the amount of functionality that can be placed in these devices. More extensive and continuous use of network services will only aggravate this problem since communication consumes relatively much energy. Research is needed to provide policies for careful management of the energy consumption while still providing the appearance of continuous connections to system services and applications. In this chapter[1] we will explore sources of energy consumption and provide a variety of energy reduction techniques at various levels in the design flow of a computer system. We will

[1] Major parts of this chapter will be published in the *Journal of Systems Architecture*, 2000 [25] and were presented at the *IEEE International Conference on Personal Wireless Communications* (ICPWC'97), 1997 [24].

try to point out the main driving forces in current research. This provides the foundation of the techniques we have applied in the design of the *Mobile Digital Companion* that is topic of the research presented in this thesis.

2.1.1 The advance of technology

The semiconductor technology has continuously improved and has lead to ever smaller dimensions of transistors, higher packaging density, faster circuits, and lower power dissipation. Over a three year period from 1998 to 2001 there will be a factor 100 increase in 3D graphics performance and nearly a factor 10 increase in hard disk capacity – far outstripping Moore's law [81]. The bandwidth of wireless networks has doubled every six months. Significant new features are being added. Video capturing, for example, is becoming a mainstream feature with MPEG-2 video encoding and decoding available on low-cost video adapters. These dramatic improvements are occurring even as the cost of computing for the average user is quickly dropping. This has been possible due to the use of parallel hardware, on-chip memory (RAM), new algorithms, and the increased level of integration of IC technology. Over the past five years, feature sizes have dropped from about 0.8µm to about 0.35µm. Semiconductor Industry Associates (SIA) have developed a road map for the next few years [62]. It is expected that a feature size of 0.1µm will be reached in 2007 within the context of our current CMOS technology. Such advances provide an effective area increase of about an order of magnitude. To avoid the effect of high-electric fields, which is present in very small devices, and to avoid the overheating of the devices, power supply must be scaled down. The power supply voltage is expected to be as low as 0.9 V in 2007.

The rapid advance in technology can be used for several purposes. It can be used to increase performance, to add functionality, but also to reduce energy consumption. One way to use this opportunity would be to continue advancing our chip architectures and technologies as just more of the same: building microprocessors that are simply more complicated versions of the kind built today. For more than thirty years, performance optimisation has been extensively studied at all abstraction levels. The current trend in industry is to focus on high-performance processors, as this is the area in which a semiconductor vendor can enhance status [10]. Therefore, the architecture of a general-purpose processor is most widely studied, and optimisations for processor performance is the main goal. Technology innovation has lead to a number of processor improvements like superscalar technology, multi-level pipelines, large on-chip caches, etc.

Another environment that will become more important in the near future is that of application specific or embedded processors. The goal of these processors is to optimise the overall cost-performance of the system, and not performance alone. The modern application-specific processors can use the novel technology to increase functionality such as compression and decompression, network access, and security functions.

Energy consumption

Power consumption has become a major concern because of the ever-increasing density of solid-state electronic devices, coupled with an increasing use of mobile computers and portable communication devices. The technology has thus far helped to build low-power systems. The speed-power efficiency has indeed gone up since 1990 by 10 times each 2.5 years for general-purpose processors and digital signal processors (DSPs). Table 1 shows the performance and power consumption of some recent processors [71]. However, this help will slow down, because physical limits seem to be reached soon.

Processor	MHz	Year	SPECint-95	Watts	Watts/SPECint
P54VRT (Mobile)	150	1996	4.6	3.8	0.83
P55VRT (Mobile MMX)	233	1997	7.1	3.9	0.55
PowerPC 603e	300	1997	7.4	3.5	0.47
PowerPC 740 (G3)	300	1998	12.2	3.4	0.28
Mobile Celeron	333	1999	13.1	8.6	0.65

Table 1: Speed and power characteristics of some recent processors.

Design for low-energy consumption is certainly not a new research field, and yet remains one of the most difficult as future mobile system designers attempt to pack more capabilities such as multimedia processing and high bandwidth radios into battery operated portable miniature packages. Playing times of only a few hours for personal audio, notebooks, and cordless phones are clearly not very consumer friendly. Also, the required batteries are voluminous and heavy, often leading to bulky and unappealing products. The primary problem is that in the case of battery technology, there is no equivalent of Moore's Law which forecasts a doubling of the complexity of microelectronic chips every 18 months, and Gilder's Law, which theorises a similar exponential growth in communication bandwidth. In contrast, battery technology has improved very slowly, and only a 20% improvement in capacity is expected over the next 10 years [63]. These trends are depicted in Figure 1 [71].

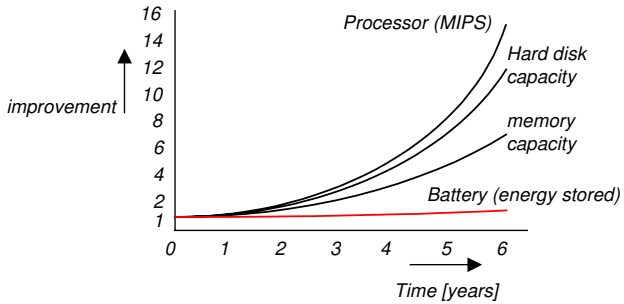


Figure 1: Improvement in technology.

With increasing computation and communication functions desired for wireless mobile systems, the energy density of existing battery technologies are far from what is needed. Table 2 shows the energetic potentials of current battery technology.

Battery	rechargeable	Wh/kg	Wk/litre
Alkaline MnO2	no	130	347
Li/MnO2	no	210	550
Zinc Air	no	280	1150
Lead acid	yes	30	80
Nickel-Cadmium NiCd	yes	40	130
Nickel-metal hybride NiMH	yes	60	200
Lithium-ion	yes	60	200
Methanol fuel cell	yes	6200	4900

Table 2: The energetic potentials of batteries.

The most recent advances in laptop batteries are in the form of better 'fuel gauging' of the battery, to give a more precise measure of the charge level and to estimate the time left before a recharge is needed [84]. Although this is a useful technique, it does not extend battery life.

A promising technique might be *fuel cells*. A fuel cell is an electrochemical device that converts the chemical energy of a fuel directly to usable energy – electricity and heat – without combustion. The energetic potential is very high, a fuel cell running on methanol could provide power for more than 20 times longer than traditional nickel cadmium batteries in a comparably sized package [15]. They are theoretically quiet and clean like normal batteries. Another benefit is that fuel cells do not require lengthy recharching; they can instead be replenished quickly, simply by adding more fuel. Fuel cells were once prohibitively expensive. But sophisticated engineering has recently

driven cost down considerably. However, designing a miniature fuel cell that can be mass-produced cheaply is a formidable task. Although initial prototypes have been built, no one has yet demonstrated a compact device that could be mass-produced at a cost lower than that of comparable rechargeable batteries.

Several researchers have studied the power consumption pattern of mobile computers. Laptops use several techniques to reduce this energy consumption, primarily by turning them off after a period of no use, or by lowering the clock frequency. However, because they studied different platforms, their results are not always in agreement, and sometimes even conflicting. Lorch reported that the energy use of a typical laptop computer is dominated by the backlight of the display, the disk and the processor [36]. Stemm et al. concluded that the network interface consumes at least the same amount of energy as the rest of the system (i.e. a Newton PDA) [73]. If the computer can receive messages from the network even when it is 'off', the energy consumption increases dramatically. Ikeda et al. observed that the contribution of the CPU and memory to power consumption has been on the rise the last few years [27].

Even though it is difficult to compare these results because the measurements are made for different architectures, operating systems, communication interfaces, and benchmarks, there is a common pattern: there is no primary source of energy consumption. The energy spent is distributed over several devices and for several operations. The conclusion is that implementing an energy efficient system involves looking at all the functions in the system, and not just a single function such as for example, network protocol processing.

2.1.2 Outline

With the increasing integration levels, energy consumption has become one of the critical design parameters. Consequently, much effort has to be put in achieving lower dissipation at all levels of the design process. It was found that most low-power research is concentrated on components research: better batteries with more power per unit weight and volume; low-power CPUs; very low-power radio transceivers; low-power displays. We found that there is very little systems research on low-power systems. While these low-level circuit and logic techniques have been well established for improving energy efficiency, they do not hold promise for much additional gain. While low-power components and subsystems are essential building blocks for portable systems, a system-wide architecture that incorporates the low-power vision into all layers of the system is beneficial because there are dependencies between subsystems, e.g. optimisation of one subsystem may have consequences for the energy consumption of other modules.

The key to energy efficiency in future mobile systems will be designing higher layers of the mobile system, their system architecture, their functionality, their operating system, and indeed the entire network, with energy efficiency in mind. Furthermore, because the applications have direct knowledge of how the user is using the system, this knowledge must be penetrated into the power management of the system.

In this chapter we will discuss a variety of energy reduction approaches that can be used for building an energy-efficient system. We have no intention to give an exhaustive overview of existing methodologies and tools for low-power systems, but try to point out the main driving forces in current research. We first explore sources of energy consumption and show the basic techniques used to reduce the power dissipation. Then we give an overview of energy saving mechanisms at the system and architectural level.

2.2 Fundamentals of low-power design

Throughout this chapter, we discuss 'power consumption' and methods for reducing it. Although they may not explicitly say so, most designers are actually concerned with reducing energy consumption. This is because batteries have a finite supply of energy (as opposed to power, although batteries put limits on peak power consumption as well). Energy is the time integral of power; if power consumption is a constant, energy consumption is simply power multiplied by the time during which it is consumed. Reducing power consumption only saves energy if the time required to accomplish the task does not increase too much. A processor that consumes more power than a competitor's may or may not consume more energy to run a certain program. For example, even if processor A's power consumption is twice that of processor B, A's energy consumption could actually be less if it can execute the same program more than twice as quickly as B.

Therefore, we introduce a metric: *energy efficiency*. We define the energy efficiency e as the energy dissipation that is essentially needed to perform a certain function, divided by the actually used total energy dissipation.

$$e = \frac{\textit{Essential energy dissipation for a certain function}}{\textit{Actually used total energy dissipation}} \qquad (1)$$

The function to be performed can be very broad: it can be a limited function like a multiply-add operation, but it can also be the complete functionality of a network protocol. Let us for example consider a medium access (MAC) protocol that controls access to a wireless channel. The essential energy dissipation is the energy dissipation needed to transfer a certain amount of bits over the wireless channel, and the total energy dissipation also includes the overhead involved in additional packet headers, error control, etc.

Note that the energy efficiency of a certain function is independent from the actual implementation, and thus independent from the issue whether an implementation is low-power. It is possible to have two implementations of a certain function that are built with different building blocks, of which one has a high energy efficiency, but dissipates more energy than the other implementation which has a lower energy efficiency, but is built with low-power components.

2.2.1 Design flow

The design flow of a system constitutes various levels of abstraction. When a system is designed with an emphasis on power optimisation as a performance goal, then the design must embody optimisation at all levels of the design. In general there are three main levels on which energy reduction can be incorporated. The *system* level, the *logic* level, and the *technological* level. For example, at the system level power management can be used to turn off inactive modules to save power, and parallel hardware may be used to reduce global interconnect and allow a reduction in supply voltage without degrading system throughput. At the logic level asynchronous design techniques can be used. At the technological level several optimisations can be applied to chip layout, packaging and voltage reduction.

An important aspect of the design flow is the relation and feedback between the levels. The system has to be designed targeted to the possible reduction of energy consumption at the technological level. Figure 2 shows the general design flow of a system with some examples of where or how energy reduction can be obtained.

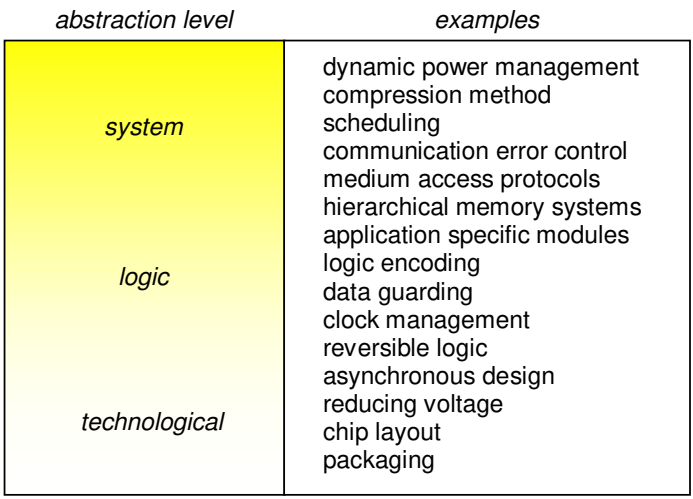


Figure 2: General design flow and related examples for energy reduction.

Given a design specification, a designer is faced with several choices at different levels of abstraction. The designer has to select a particular algorithm, design or use an architecture that can be used for it, and determine various parameters such as supply voltage and clock frequency. This multi-dimensional design space offers a large range of possible trade-offs. At the highest level the design decisions have the most influence. Therefore, the most effective design decisions derive from choosing and optimising architectures and algorithms at the highest levels. It has been demonstrated by several researchers [63] that system and architecture level design decisions can have dramatic impact on power consumption. However, when designing a system it is a problem to predict the consequences and effectiveness of high level design decisions because

implementation details can only be accurately modelled or estimated at the technological level and not at the higher levels of abstraction. Furthermore, the specific energy reduction techniques that are offered by the lower layers can be most effective only when the higher levels are aware of these techniques, know how to use them, and apply them.

2.2.2 CMOS component model

Most components are currently fabricated using CMOS technology. Main reasons for this bias is that CMOS technology is cost efficient and inherently lower power than other technologies. The sources of energy consumption on a CMOS chip can be classified as *static* and *dynamic* power dissipation. The main difference between them is that dynamic power is frequency dependent, while static is not. Bias (P_b) and leakage currents (P_l) cause static energy consumption. Short circuit currents (P_{sc}) and dynamic energy consumption (P_d) is caused by the actual effort of the circuit to switch.

$$P = P_d + P_{sc} + P_b + P_l \qquad (2)$$

The contributions of this static consumption are mostly determined at the circuit level. While statically-biased gates are usually found in a few specialised circuits such as PLAs, their use has been dramatically reduced in CMOS design [5]. Leakage currents also dissipate static energy, but are also insignificant in most designs (less than 1%). In general we can say that careful design of gates generally makes their power dissipation typically a small fraction of the dynamic power dissipation, and hence will be omitted in further analysis.

Dynamic power can be partitioned into power consumed internally by the cell and power consumed due to driving the load. Cell power is the power used internally by a cell or module primitive, for example a NAND gate or flip-flop. Load power is used in charging the external loads driven by the cell, including both wiring and fanout capacitances. So the dynamic power for an entire chip is the sum of the power consumed by all the cells on the chip and the power consumed in driving all the load capacitances.

During the transition on the input of a CMOS gate both *p* and *n* channel devices may conduct simultaneously, briefly establishing a short from the supply voltage to ground ($I_{crowbar}$). This effect causes a power dissipation of approx. 10 to 15%.

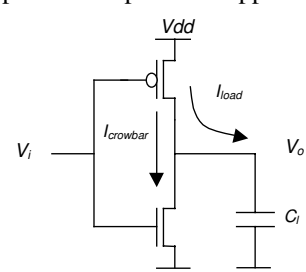


Figure 3: Dynamic power in a CMOS inverter.

The more dominant component of dynamic power is capacitive power. This component is the result of charging and decharging parasitic capacitances in the circuit. Every time a capacitive node switches from ground to V_{dd} an vice-versa energy is consumed.

The dominant component of energy consumption (85 to 90%) in CMOS is therefore *dynamic*. A first order approximation of the dynamic energy consumption of CMOS circuitry is given by the formula:

$$P_d = C_{eff} V^2 f \qquad (3)$$

where P_d is the power in Watts, C_{eff} is the effective switch capacitance in Farads, V is the supply voltage in Volts, and f is the frequency of operations in Hertz [33]. The power dissipation arises from the charging and discharging of the circuit node capacitance found on the output of every logic gate. Every low-to-high logic transition in a digital circuit incurs a voltage change ΔV, drawing energy from the power supply. C_{eff} combines two factors C, the capacitance being charged/discharged, and the *activity weighting* α, which is the probability that a transition occurs.

$$C_{eff} = \alpha C \qquad (4)$$

A designer at the technological and architectural level can try to minimise the variables in these equations to minimise the overall energy consumption. However, as will be shown in the next sections, power minimisation is often a subtle process of adjusting parameters in various trade-offs.

2.2.3 Power modelling and analysis

The search for the optimal solution must include, at each level of abstraction, a 'design improvement loop'. In such a loop a power analyser/estimator ranks the various design, synthesis, and optimisation options, and thus helps in selecting the one that is potentially more effective from the energy consumption standpoint. Obviously, collecting the feedback on the impact of the different choices on a level-by-level basis, instead of just at the very end of the flow (i.e. at the gate level), enables a shorter development time. On the other hand, this paradigm requires the availability of power estimators, as well as synthesis and optimisation tools, that provide accurate and reliable results at various levels of abstraction. Power analysis tools are available primarily at the gate and circuit levels, and not at the architecture and algorithm levels where they could really make an impact. Current research is trying to fill this gap [33][43][86].

2.2.4 How much is a picojoule?

In Table 3 we compare the energy of a transition of a CMOS gate (about 1 picojoule) with a variety of energy quantities.

Note that neural transitions are an order of magnitude more efficient (in average). An assignment consumes about 100 picojoules for a word of eight bits. Note that the execution of a single instruction of the most efficient commercial available 32-bit microprocessor around, the ARM, dissipates two order of magnitude more. The

instruction of a DEC Alpha consumes even a thousand times more energy. This clearly suggests that micro-processors, due to their general-purpose nature, are not particularly energy efficient.

Quantity	Energy	Remark
Uv photon	-18	
Neural transition	-13	Varies with size
CMOS transition	-12	Gate with 100fF load
α-particle	-12	From space or IC package
8-bit assignment	-10	
PCB transition	-10	10 pF load
ARM instruction	-8	
8-bit access 16Mb SRAM	-8	
DEC Alpha instruction	-7	
Correcting DCC word	-6	
NiCd penlight battery	3	
Can of beer	6	600 kJ
Lead-acid car battery	6	5kg x 40Wh/kg
Kg coal	7	
Daily human consumption	7	2500 kilocalories
Man-made nuclear explosion	14	Trinity (July 16, 1944)
1906 San Francisco earthquake	17	8.3 on the Richter scale
Nova	37	
Big bang	73	

Table 3: The 10-log of the energy for various quantities [10-log Joules] (from [7]).

2.3 Low-power technological-level design

The previous section has presented the theoretical foundation for the analysis of energy consumption. From this section onwards, we will discuss the energy reduction techniques and trade-offs that involve energy consumption of digital circuits. We use a bottom-up organisation starting at the lowest level of abstraction and proceed upward. As we move up the abstraction level, the techniques and trade-offs become less exact due to more freedom in design configuration and decision.

The Equations (3) and (4) suggest that there are essentially four ways to reduce power:

- reduce the capacitive load C,
- reduce the supply voltage V,
- reduce the switching frequency f,
- reduce the switching activity α.

Despite the differences in optimisation and trade-off possibilities at the various levels of abstraction, the common themes of the low-power techniques are quite similar.

The technological level comprises the technology level, dealing with packaging and process technologies, the layout level that deals with strategies for low-power placement and routing, and the circuit level that incorporates topics like asynchronous logic and dynamic logic.

2.3.1 Minimise capacitance

Energy consumption in CMOS circuitry is proportional to capacitance C. Therefore, a path that can be followed to reduce energy consumption is to minimise the capacitance. This can not only be reached at the technological level, but much profit can be gained by an architecture that exploits locality of reference and regularity.

Connection capacity and packaging

A significant fraction of the chip's energy consumption is often contributed to driving large off-chip capacitances, and not to core processing. Off-chip capacitances are in the order of five to tens of picofarads, while on-chip capacitances are in tens of femtofarads. For conventional packaging technologies, [3] suggests that pins contribute approximately 13-14 pF of capacitance each (10 pF for the pad and 3-4 pF for the printed circuit board). Since Equation (3) indicates that energy consumption is proportional to capacitance, I/O power can be a significant portion of the overall energy consumption of the chip. Therefore, in order to save energy, use few external outputs, and have them switch as infrequently as possible.

Packaging technology can have a impact on the energy consumption. For example, in multi-chip modules where all of the chips of a system are mounted on a single substrate and placed in a single package, the capacitance is reduced. Also, accessing external memory consumes much energy. So, a way to reduce capacitance is to reduce external accesses and optimise the system by using on-chip resources like caches and registers.

Example

To indicate the contribution of the interconnect to the total energy consumption we will compare the energy consumption that is required to compute a 32 x 32 bit multiply with the energy that is needed to fetch a 32-bits word from external memory.

Energy needed to perform a 32 x 32 multiply

The minimal amount of energy consumed for a single m x n multiply is given by [64]:

$$E_{mul} = m \,.\, n \,.\, \rho_{mul} \,(E_{fa} + E_{and}) \qquad (5)$$

When implemented in a 1 μm, 5V CMOS process, the energy for a full adder E_{fa} = 2.41 pJ and the energy to perform the AND E_{and} = 0.35 pJ. The ripple factor ρ_{mul} represents any extra energy due to ripples within the arithmetic element. Depending on the architecture it can take values between 1.5 and 2.5. So, with a ρ_{mul} = 2, the amount of energy for a single 32 x 32 multiplication equals approximately 5.7 nJ.

Energy needed to transfer data from memory

We are not interested here in the energy dissipation of the memory-core itself and concentrate on the energy consumption to transfer data bits over capacitive I/O pins. This amount of energy consumed can be expressed with:

$$E_{pad} = p \,.\, ½\, C_{IO} \,.\, V^2 \qquad (6)$$

in which *p* equals the transition probability. When we estimate the transition probability to be 0.5, use a capacitance C_{IO} of an I/O pad of 5 pF and an I/O voltage of 5 V, then one I/O pad needs an energy E_{pad} of 31.25 pJ. With a memory organisation consisting of 16 banks of 2 bits connected to the processor with an interface chip (e.g. an MMU) we have 19 I/O pads for the address (1 to the processor, 2 to the memory interface, and 16 to the memory), and 4 I/O pads for the data. The amount of energy required transferring a 32-bit data over the I/O pins between memory and processor using 24 bits address and 3 control lines thus requires (24.19 + 3.19 + 32.4). E_{pad} = 20 nJ.

The amount of energy consumed just to transfer one 32 bit word between memory and processor is thus almost four times the amount of energy needed for a 32 x 32 bit multiplication. When a better process – like 0.25 μm and 1.8 V – is used, the multiply will become much less energy consuming because it is smaller (16 times) (see the next paragraph) and because of the lower voltage (7.6 times less energy). The I/O pad has no advantage of the smaller feature size because its capacitance will remain about the same.

❑

Technology scaling

The process technology has been improved continuously, and as the SIA roadmap indicates, the trend is expected to continue for years [62]. Scaling of the physical dimension involves reducing all dimensions: thus transistor widths and lengths are reduced, interconnection length is reduced, etc. Consequently, the delay, capacitance and energy consumption will decrease substantially. For example, MIPS Technologies attributed a 25% reduction in power consumption for their new processor solely to a migration from a 0.8 μm to a 0.64 μm process [90].

Another way to reduce capacitance at the technology level is thus to reduce chip area. However, note that a sole reduction in chip area at architectural level could lead to an energy-inefficient design. For example, an energy efficient architecture that occupies a larger area can reduce the overall energy consumption, e.g. by exploiting locality in a parallel implementation.

Chip layout

There are a number of layout-level techniques that can be applied. Since the physical capacitance of the higher metal layers are smaller, there is some advantage to select upper level metals to route high-activity signals. Furthermore, traditional placement involves minimising area and delay, which in turn translates to minimising the physical capacitance (or length) of wires. Placement that incorporates energy consumption, concentrates on minimising the activity-capacitance product rather than capacitance alone. In general, high-activity wires should be kept short and local. Tools have been developed that use this basic strategy to achieve about 18% reduction in energy consumption [12].

Conclusion on capacitance reduction

The capacitance is an important factor for the energy consumption of a system. However, reducing the capacity is not *the* distinctive feature of low-power design, since in CMOS technology energy is consumed only when the capacitance is switched. It is more important to concentrate on the switching activity and the number of signals that need to be switched. Architectural design decisions have more impact than solely reducing the capacitance.

2.3.2 Reduce voltage and frequency

One of the most effective ways of energy reduction of a circuit at the technological level is to reduce the supply voltage, because the energy consumption drops quadratically with the supply voltage. For example, reducing a supply voltage from 5.0 to 3.3 Volts (a 44% reduction) reduces power consumption by about 56%. As a result, most processor vendors now have low voltage versions. The problem that then arises is that lower supply voltages will cause a reduction in performance. In some cases, low voltage versions are actually 5 Volt parts that happen to run at the lower voltage. In such cases the system clock must typically be reduced to ensure correct operation. Therefore, any such voltage reduction must be balanced against any performance drop. To compensate and maintain the same throughput, extra hardware can be added. This is successful up to the point where the extra control, clocking and routing circuitry adds too much overhead [58]. In other cases, vendors have introduced ‘true’ low voltage versions of their processors that run at the same speed as their 5 Volt counterparts. The majority of the techniques employing concurrency or redundancy incur an inherent penalty in area, as well as in capacitance and switching activity. If the voltage is allowed to vary, then it is typically worthwhile to sacrifice increased capacitance and switching activity for the quadratic power improvement offered by reduced voltage.

The variables voltage and frequency have a trade-off in delay and energy consumption. Reducing clock frequency f alone does not reduce energy, since to do the same work the system must run longer. As the voltage is reduced, the delay increases. A common approach to power reduction is to first increase the performance of the module – for example by adding parallel hardware –, and then reduce the voltage as much as possible so that the required performance is still reached (Figure 4). Therefore, major themes in

many power optimisation techniques are to optimise the speed and shorten the critical path, so that the voltage can be reduced. These techniques often translate in larger area requirements, hence there is a new trade-off between area and power.

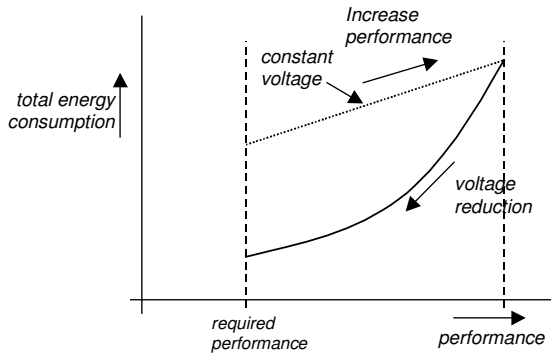


Figure 4: Impact of voltage scaling and performance to energy consumption.

Weiser et al. [76] have proposed a system in which the clock frequency and operating voltage is varied dynamically under control of the operating system while still allowing the processor to meet its task completion deadlines. In order to operate properly at a lower voltage, the clock rate must be simultaneously reduced.

The main limitation of all voltage scaling approaches is that they assume the designer has the freedom of choosing the voltage supply for the design. Unfortunately, for many real-life systems, the power supply is not a variable to be optimised. Furthermore, in current and future technologies, voltage scaling will become increasingly difficult because of reduced noise margins and deterioration of device characteristics [5].

2.3.3 Avoid unnecessary activity

We can summarise the previous subsections as follows: the capacitance can only marginally be changed and is only important if switched, the voltage is usually not under designer's control, and the clock frequency, or more generally, the system throughput is rather a constraint than a design variable. The most important factor contributing to the energy consumption is the switching activity. Actually, once the technology and supply voltage have been set, major energy savings come from the careful minimisation of the switching activity α of Equation (4).

While some switching activity is functional, i.e. it is required to propagate and manipulate information, there is a substantial amount of unnecessary activity in virtually any digital circuit. Unnecessary switching activity is due to 1) spurious transitions due to unequal propagation delays (glitches), 2) transitions occurring within units that are not participating in a computation or 3) whose computation is redundant.

Reordering of logic inputs to circuits can have significant energy consumption consequences. For example, Figure 5 shows two functional identical circuits, but with a

different energy consumption due to the different signalling activity. The normalised energy consumption equals 0.11 of circuit *a*, and 0.021 for circuit *b*.

Thus, much energy can be saved by minimising the amount of switching activity needed to carry out a given task within its performance constraints. The activity weighting α of Equation (4) can be minimised by avoiding unnecessary and wasteful activity. There are several techniques to achieve this. In this section we will only mention the techniques at the technological level and circuit level. The techniques that are possible at the logic, architectural and system level are discussed in later sections.

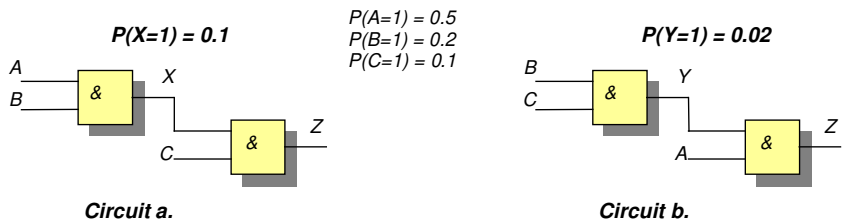


Figure 5: Reordering logic inputs.

Asynchronous design

One way to avoid unnecessary activity is by applying an asynchronous design methodology [6][47][23][55]. CMOS is a good technology for low power as gates mainly dissipate energy when they are switching. Normally this should correspond to the gate doing useful work, but unfortunately in a synchronous circuit this is not always the case. The circuit activity is often low (below 10%), for one of the following reasons [7].

- The clock frequency often exceeds the sample frequency by several orders of magnitude, or order to allow for time sharing of resources such as busses, I/O pads, memories, etc.
- Large ICs consist of a number of more-or-less independent modules. These modules generally have different optimal clock frequencies. Nevertheless, the number of different clock frequencies on the IC is kept small to avoid problems with interfacing, clock distribution, and testing.
- The clock frequency must be such that a worst-case workload can be handled in the allocated time. This generally implies an excessively high clock frequency for the average case.

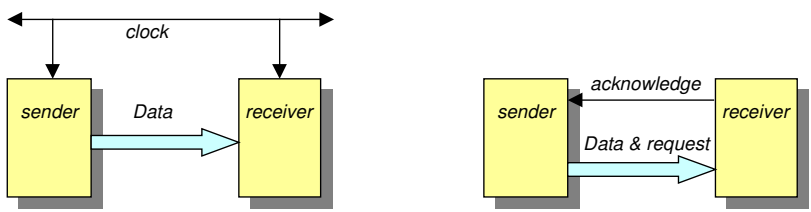
Many gates switch because they are connected to the clock, not because they have new inputs to process. A synchronous circuit therefore wastes power when particular blocks of logic are not utilised, for example, in a floating point unit when integer arithmetic is being performed. The biggest gate of all is the clock driver that must distribute a clock signal evenly to all parts of a circuit, and it must switch all the time to provide the timing reference even if only a small part of the chip has something useful to do.

Example [86]

The chip size of a CPU is 15 x 25 mm with clock frequency of 300 MHz operating at 3.3V. The length of the clock routing is estimated to be twice the circumference of the chip. Assume that the clock signal is routed on a metal layer with width of 1.2 µm and the parasitic capacitance of the metal layer is 1 fF/µm^2. Using Equation (3) the power dissipation of the clock signal is then 627 mW.

❑

This example is even conservative: in the DEC Alpha the distribution of the single-wire 200 MHz clock requires 7.5 Watts out of a total of 30 Watts. The clock driver dissipates another 4.5 Watts [7]!



a) synchronous communication protocol *b) self-timed protocol*

Figure 6: Synchronous and self-timed protocols.

In *asynchronous systems* there is no global synchronisation of the modules within the system, but modules do synchronise locally through their communication protocols [20]. The set of asynchronous communication protocols that use some form of handshake between sender and receiver are known as *self-timed*. Asynchronous circuits are inherently data driven and are only active when performing useful work. An important property of self-timed protocols is that they are speed independent: the protocols work regardless of the speed that the modules produce results. In addition, protocols such as the dual-rail code are delay-insensitive: arbitrary delays may occur on the wiring and the protocol will still work. Among several advantages like modularity and robustness, power consumption is low.

The main drawback of asynchronous systems is that extra circuitry and signals is required to generate the timing information.

2.3.4 Technological and circuit-level conclusions

Clearly, numerous techniques at the technological and circuit level are available to the low-power system designer. These techniques include optimisation in packaging, technology scaling and layout level, a careful selection of the techniques used at circuit level, and applying energy saving techniques at gate level.

The gains that can be reached at these levels are, however, limited. The technology scaling for example offers significant benefits in terms of energy consumption only up to

a certain point. Once parasitics begin to dominate, the power improvements slack off or disappear completely. So we cannot rely on technology scaling to reduce energy consumption indefinitely. We must turn to other techniques for lowering energy consumption. Some of the techniques can be applied in conjunction with higher-level energy-reduction techniques.

The main concepts that are used at gate-level are associated with avoiding unnecessary activity and trading energy for performance through low-voltage concurrent processing. The gains reported by low-power designers working at the gate level are typically on the order of a factor of two or less [33]. However, technology- and circuit-level techniques can have major impact because some circuits are repeated thousands of times on a chip.

So, while these techniques should be exploited whenever possible, this should not be done at the expense of the larger gains achievable at the architecture and algorithm levels, which will be discussed in the following sections.

2.4 Low-power logic-level design

We consider the logic level as the level between the technological related issues and the system level. Issues in the logic level relate to for example state-machines, clock gating, encoding, and the use of parallel architectures.

At the logic level, opportunities to economise on power exist in both the capacitance and frequency spaces. The most prevalent theme in logic-level optimisation techniques is the reduction of switching activities. We will now give some typical examples of reducing energy consumption at this level.

2.4.1 Cell library

The choice of the cell library to use for a chip design provides the first obvious opportunity to save energy. Standard cells have lower input capacitances than gate arrays because they use a variety of transistor sizes. For the same reason, the cells themselves consume less power when switching. Using libraries designed for low power can also reduce capacitance. These libraries contain cells that have low-power micro-architectures or operate at very low voltages. Some of the leading application-specific IC (ASIC) vendors are providing such libraries today, and many captive design groups are producing specialised libraries for low-power applications. But no matter which type of library is utilised, the logic designer can minimise the power used by each cell instance by paying careful attention to the transition times of input and output signals. Long rise and fall times should be avoided in order to minimise the crowbar current component of the cell power.

2.4.2 Clock gating

Several power minimisation techniques work especially well at the logic level. Most of them rely on switching frequency. The best example of which is the use of *clock gating*. Because CMOS power consumption is proportional to the clock frequency, dynamically turning off the clock to unused logic or peripherals is an obvious way to reduce power consumption [28][35]. In clock gating, a control signal enables a clock signal so that the clock toggles only when the *enable* signal is true, and is held steady when the *enable* signal is false. Gated clocks are used, in power management, to shut down portions of the chip, large and small, that are inactive. This saves on clock power, because the local clock line is not toggling all the time.

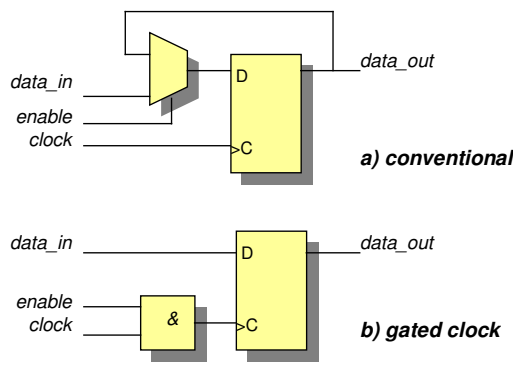


Figure 7: Clock gating.

Consider the case of a data bus input register as depicted in Figure 7. With the conventional scheme, the register is clocked all the time, whether new data is to be captured or not. If the register must hold the old state, its output is fed back into the data input through a multiplexer whose enable line controls whether the register clocks in new data or recycles the existing data. With a gated clock, the signal that would otherwise control the select line on the multiplexer now controls the gate. The result is that the energy consumed in driving the register's clock input is reduced in proportion to the decrease in average local clock frequency. The two circuits function identically, but utilisation of the gated clock reduces the power consumption.

Clock gating can be implemented locally by gating the clocks to individual registers, or globally, by building the gating structures into the overall architecture to turn off large functional modules. While both techniques are effective at reducing energy, global gating results in much larger energy reductions and is often used in implementing power-down and power-management modes. Some processors and hardware devices have sleep or idle modes. Typically they turn off the clock to all but certain sections to reduce power consumption. While asleep, the device does no work. Control can be done at the hardware level or the operating system or the application can manage it. The PowerPC603, for example, contains three power management modes – doze, nap, and sleep – that are controlled by the operating system and cut power use overall when the processor is idle for any extended period of time. With these modes, chip power can go

from 2.2 W in active mode to 358 mW in doze, 126 mW in nap, and as low as 1.8 mW in sleep.

A wake-up event wakes the device from the sleep mode. Devices may require different amounts of time to wake up from different sleep modes. For example, many 'deep sleep' modes shut down on-chip oscillators used for clock generation. A problem is that these oscillators may require microseconds or sometimes even milliseconds to stabilise after being enabled. So, it is only profitable to go into deep sleep mode when the device is expected to sleep for a relatively long time. The system has to predict whether it is profitable to shut down parts of the system.

2.4.3 State-machine modifications

A state-machine is an abstract computation model in which the designer specifies a state-transition graph. This can be implemented using Boolean logic and flip-flops. Among the modifications that can be made to achieve a higher energy efficiency are *decomposition* and *restructuring*. These approaches try to minimise the activity along the lines connecting the sub-machines, which tend to drive heavier loads. Shutdown techniques like clock-gating can be applied to the individual machines because only one is active at any point in time.

The *encoding* of the state machine (which is the encoding of states to bits in the state register) is another important factor that determines the quality (area, energy consumption, speed, etc.) of the system. Several key parameters have been observed to be important to the energy efficiency of state encoding. One such parameter is the expected number of bit transitions in the state register. Another parameter is the expected number of transitions of output signals.

Consider for example two functionally identical state machines M1 and M2 with different encoding shown in Figure 8 (figure and example from [89]). The labels at the state transitions edges represent the probability that transitions will occur at any given clock cycle. The expected number of state-bit transitions $E[M]$ is given by the sum of products of edge probabilities and their associated number of bit-flips as dictated by the encoding.

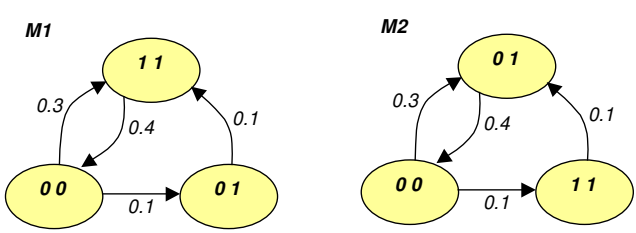


Figure 8: Functionally identical state machines with different encoding

The expected transitions per clock cycle for the two machines are:

$$E[M1] = 2\ (0.3 + 0.4) + 1\ (0.1 + 0.1) = 1.6$$

$$E[M2] = 1\ (0.3 + 0.4 + 0.1) + 2\ (0.1) = 1.0$$

In general, machines with lower E[M] are more energy efficient because there are fewer transitions of the state register and fewer transitions are propagated into the combinatorial logic of the machine.

Note that encoding affects the energy dissipation as well as the required area of the machine. If we encode the states to minimise the expected transition, the area is often large because the logic synthesis system has lost the freedom of assigning state codes that favour area optimisation. One practical solution is to encode only the subset of states that spans the high probability edges. The remaining state codes can be left to the logic synthesis system

Unlike area optimisation, the power optimisation of state machines requires the knowledge of the probabilities of input signals and hence state transitions. Without this knowledge, power optimisation of state encoding is impossible.

The principle of clock-gating discussed above can also be applied in state-machines: for example in the design of synchronous finite state machines (FSM) the clock can be disabled if the unit is not needed at a certain moment. For example Koegst et al. [31] use gated clocks in FSM designs to disable the state transition of so called self-loops.

2.4.4 Logic encoding

Energy consumption is proportional to the frequency at which signals change state and to the capacitance on the signal line. This is true for every signal in a system, whether it is a clock signal, a data pin, or an address line. This implies that power consumption can be reduced by carefully minimising the number of transitions. The designer of a digital circuit often has the freedom of choosing the encoding scheme. Different encoding implementations often lead to different power, area and delay trade-off. A correct choice of the representation of the signals can have a large impact on the switching activity. Usually, encoding techniques require the knowledge of signal statistics.

The frequency of consecutive patterns in the traffic streams is the basis for the effectiveness of encoding mechanisms. For example, program counters in processors generally use a binary code. On average, two bits are changed for each state transition. Using a Gray code, which will typically result in single bit changes, can give interesting energy savings. However, a Gray code incrementer requires more transistors to implement than a ripple carry incrementer [57]. Therefore, a combination can be used in which only the most frequently changing LSB bits use a Gray code.

A simple, yet effective, low-power encoding scheme is the bus-invert code [72]. If the Hamming distance between two successive patterns is larger than $N/2$, where N is the bus width, the current pattern is transmitted with inverted polarity. A redundant bus line is needed to signal to the receiving end of the bus which polarity is used for the transmission of the incoming pattern. The method guarantees a maximum of $N/2$

transitions, and performs well when the patterns are randomly distributed in time and no information about their correlation is available.

2.4.5 Data guarding

Switching activity is the major cause of energy dissipation in most CMOS digital systems. In order to transfer data and do computation, switching activities cannot be avoided. However, switching activities that do not contribute to the actual communication and computation should be eliminated. The basic principle is to identify logical conditions at some inputs to a logic circuit that is invariant to the output. Since those input values do not affect the output, the input transitions can be disabled. The approach is based on reducing the switching activities by placing some *guard logic*, consisting of transparent latches with an enable signal, at the inputs of each block of the circuit that needs to be power managed. This logic will guard not useful switching activities to propagate further inside the system. The latches are transparent when the data is to be used. Otherwise, if the outputs of a unit are not used, then they do not change.

The position of the registers within a design may greatly affect the area and performance of the implementation. The transformation that repositions the registers of a design without modifying its external behaviour is called *retiming*. If a register can be positioned before the output of the circuit, some spurious transitions (i.e. glitches) are filtered by the register and thus not propagated further.

2.4.6 Conclusion

With the advance of logic synthesis tools and structured VLSI design practice today, logic design is seldom performed manually. However, the logic-level design can have a high impact on the performance and energy-efficiency of the system. Even with the use of hardware description languages like VHDL, there are still many techniques for the designer to reduce energy consumption at the logic level. The most effective technique used at this level is the reduction of switching activities.

2.5 Low-power system-level design

In the previous section we have explored sources of energy consumption and showed the low-level design techniques used to reduce the power dissipation. We already concluded that the impact of these techniques is limited due to several factors. It is unlikely that the combined effort of technology-level, gate-level and circuit level reduce power by more than a factor of two in average [5]. Technology roadmaps and trend analyses [62] clearly show that this result is far from being sufficient. In this section we will concentrate on the energy reduction techniques at architecture and system level, and we will evaluate the relevance for low-power system design.

We define a *system* as an interconnection of possibly heterogeneous resources (electronic, electro-mechanical, optical, etc.) which are often separately designed. System-level design deals with connecting the resources in a functionally correct and efficient fashion. In this definition all practical digital devices like computers and PDAs are systems. A chip can be a system if its components are designed and optimised as separate resources.

The two main themes that can be used for energy reduction at these higher levels are:

- avoid unnecessary activity, and
- exploit locality of reference.

Typical examples at these levels include algorithmic transformations, partitioning, memory organisations, power management, protocol design and selecting dedicated versus programmable hardware.

2.5.1 Optimise communication channels

The observation has already been made that energy in real-life systems is to a large extend dissipated in *communication channels*, sometimes even more than in the computational elements. Experiments have demonstrated that in designs, about 10 to 40% of the total power may be dissipated in buses, multiplexers and drivers [1]. This amount can increase dramatically for systems with multiple chips due to large off-chip bus capacitance.

The power consumption of the communication channels is highly dependent on algorithm and architecture-level design decisions. Two properties of algorithms and architectures are important for reducing the energy consumption due to the communication channels: locality and regularity.

Locality relates to the degree to which a system or algorithm has natural isolated clusters of operation or storage with few interconnections between them. Partitioning the system or algorithm into spatially local clusters ensures that the majority of the data transfers take place within the clusters and relatively few between clusters. The result is that the local buses with a low electrical capacity are shorter and more frequently used than the longer highly capacitive global buses. Locality of reference can be used to partition memories. Current high-level synthesis tools are targeted to area minimisation or performance optimisation. However, for power reduction it is, for instance, better to minimise the number of accesses to long global buses and have the local buses be accessed more frequently. In a direct implementation targeted at area optimisation, hardware sharing between operations might occur, destroying the locality of computation. An architecture and implementation should preserve the locality and partition and implement it such that hardware sharing is limited. The increase in the number of functional units does not necessarily translate into a corresponding increase in the overall area and energy consumption since (1) localisation of interconnect allows a more compact layout and (2) fewer (access to) multiplexers and buffers are needed.

Localisation reduces the communication overhead in processors and allows the use of minimum sized transistors, which results in reductions of capacitance. Pipelining and

caching are examples of localisation. Another way to reduce data traffic over a 'long' distance is to integrate a processor in the memory, as for example proposed by Patterson in intelligent RAM [53][48]. This approach also reduces the processor-memory bottleneck.

At system level locality can be applied by dividing the functionality of the system into dedicated modules [1][44]. When the system is decomposed into application-specific modules, the data traffic can be reduced, because unnecessary data copies are removed. For example, in a system where a stream of video data is to be displayed on a screen, the data can be copied directly to the screen memory, without going through the main processor.

Regularity in an algorithm refers to the repeated occurrence of computational patterns. Common patterns enable the design of less complex architecture and therefore simpler interconnect structure (buses, multiplexers, buffers) and less control hardware. Several researchers (e.g. Mehra [49] and Rabaey [59]) have exploited these techniques, but mainly in the DSP domain where a large set of applications inherently have a high degree of regularity.

2.5.2 Low-power memory organisation

Closely related to the previous section that discussed the optimisation of communication channels, is the memory organisation. In processor based systems, a significant fraction of the total energy budget is consumed in memories and buses. Minimising the memory accesses, and clustering them, minimises the cost of bus transactions and can reduce energy consumption. Accesses can be reordered and special bus-encoding techniques can be used to minimise the transition activity on memory busses.

There are furthermore various techniques to reduce the energy consumption of the secondary storage. Secondary storage is in general non-volatile and is used to store large amounts of data. Caching techniques can be used for both main memory and secondary storage to increase performance and reduce energy consumption.

Main memory

Main memory is generally implemented using Dynamic Random Access Memories (DRAM or SDRAM). These chips can be in three modes: active, standby, and off. In active mode, the chip is reading or writing. In standby mode, the chip needs to be refreshed periodically to maintain the data stored in the memory cells. The energy consumption of DRAM can be significant, for example 8 MB of EDO DRAM memory from Micron consumes about 580 mW in active mode, and 1.65 mW in standby mode. Static memory (SRAM) does not need to be refreshed, and therefore consumes less energy in standby mode. SRAM is in general faster than DRAM, requires more chip area, and is more expensive. For example, the Samsung KM616FS1000ZI 128K*16 100 ns SRAM consumes 216 mW when active, and 0.1 mW when standby. Note that using more energy-consuming memory can be more energy-efficient when it is faster, and can thus remain longer in a lower energy consuming mode [64].

The 'off' state of main memory chips can only be used when it is determined that the entire system will be idle for a significant period of time. The content of all main memory is saved to secondary storage before the main memory system can be turned off.

There are various ways to reduce the energy consumption of a memory array. Tierno and Martin describe various memory organisations in relation to their energy consumption [74]. They conclude that a memory organisation as a multidimensional array of memory cells gives an improvement in energy per access, but requires more chip area.

Breaking up the memory into several sub-arrays can further reduce the energy per access, so that only one of the smaller sub-arrays is accessed in each memory reference. This technique is for example applied in the Direct Rambus DRAM system (RDRAM) [80]. The Rambus physical layer contains 30 high speed, controlled impedance, matched transmission lines. These high-speed signals are terminated at their characteristic impedance at the RDRAM end of the channel. Power is dissipated on the channel only when a device drives a logic '1' (low-voltage) on the pin. All high-speed signals use low-voltage swings of 800 mV. The RDRAM has several built-in operating states to reduce energy consumption. In active mode the RDRAM is ready to immediately service a memory transaction request. At the end of a transaction an RDRAM automatically transitions to standby mode.

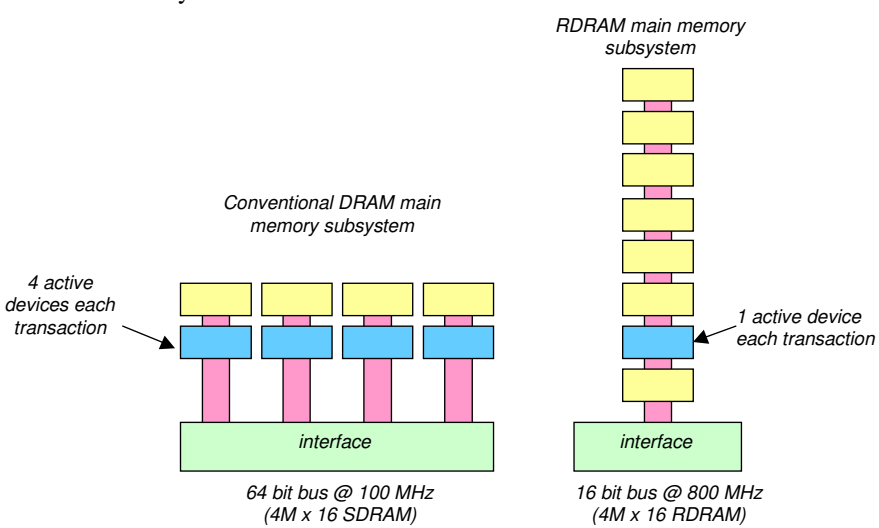


Figure 9: Conventional SDRAM and RDRAM block diagram

The organisation of a RDRAM device allows an RDRAM main memory subsystem to consume less power than a conventional system. Figure 9 shows a conventional DRAM memory subsystem topology compared to a RDRAM design. Note that for each transaction request from the memory subsystem an entire bank of conventional DRAM devices are activated and consume energy versus a single device activation with the Rambus design. Since a conventional DRAM device's throughput is much lower than an

RDRAM, several DRAM devices are operated in parallel as a bank of memory to provide the desired data bandwidth onto a 64-bit data bus. When a memory transaction request is sent out to the memory array the appropriate RDRAM device services the request and moves to active mode while the other RDRAMs remain in standby mode.

Employing nap mode can further reduce power consumption. In nap mode the energy consumption is reduced to 10% of the energy consumption in standby mode. Some extra latency is introduced to return from nap mode to standby.

A similar memory organisation technique can be used with conventional memory devices. When the memory is divided into several small blocks that can be individually powered down, then the memory allocation strategy and garbage collector of the operating system can take benefit of this by allocating the memory in clustered memory blocks, such that unused memory is not spread over all memory banks.

Caching

The previously described memory organisation techniques try to minimise the energy cost of a single access to memory, under the assumption that all addresses are equally probable. In most applications, address distributions are far from random: the past history of the address sequence can be used to increase memory throughput and decrease the average energy per access. The assumptions made about the address sequence are *spatial* and *temporal locality*. Spatial locality indicates that once an address has been accessed, there is a strong probability that a nearby address will be accessed in the near future. Temporal locality indicates that, once an address has been accessed, there is a strong probability that the same address will be accessed again in the near future. Spatial locality is used by pre-fetch mechanisms. The cost per word of fetching a multi-word line from memory decreases with the number of words on the line (the energy cost of decoding the address is shared among more words). Temporal and spatial locality can be used to store a copy of the contents of the memory locations most likely to be needed in the future, in a small, fast, energy-efficient memory. If the locality is strong enough, most of the memory references will be serviced by the small memory (e.g. a cache memory), with a corresponding improvement in energy performance.

By employing an on-chip *cache* significant power reductions can be gained together with a performance increase. For example, improvements in cache organisations for high performance processors also reduce traffic on high capacitance busses. As, most of the time, only the cache is read, the energy consumption is reduced. This phenomenon clearly helps to save energy although the primary goal is improving performance. However, as these caches are typically implemented with high-performance static RAM cells, these caches often consume a significant amount of energy. For example, two modern embedded RISC microprocessors, the StrongARM 110 and a PowerPC, the energy consumption of the cache is either the largest or second largest power-consuming block [29].

A cache designed for low-energy has to optimise the hit ratio at relatively small cache sizes. This is because in general caches with good hit ratios use complicated architectures, which make the energy cost of a cache access high. Special techniques,

like using a small *filtering cache* that is placed just before the normal (large, complex and high energy consuming) cache, trade off performance for energy consumption. Experimental results across a wide range of embedded applications show that the filter cache results in improved memory system energy efficiency (for example a direct mapped 256-byte filter cache can achieve a 58% energy reduction while reducing the performance by 21%) [22].

Note, however, that, although for many applications and data streams these techniques are profitable for performance and energy consumption, for streaming data, these caches might even become an obstacle to high performance and low power. This kind of traffic – typically multimedia traffic – is characterised by its one-time reference [2]. The locality principle, the key property behind the use of caches, is not valid for this kind of traffic. The media functions typically involve processing a continuous stream of input [32], thereby effectively emptying the cache of useful processor data. It has been observed that future processor designs spend a large fraction of their area budget on local caches, and not on energy conserving techniques. The *Computer* journal produced a special issue on "Billion-transistor architectures" that discussed problems and trends that will affect future processor designs [10]. Most of these designs focus on the desktop and server domain. The majority use 50 to 90 percent of their transistor budget on caches, which helps mitigate the high latency and low bandwidth of external memory. In other words, in the vision of future computer designers most of the billion-transistor budget is spent on redundant, local copies of data normally found elsewhere in the system [32].

The compiler used to make a program can also utilise the locality principles by reducing the number of instructions with memory operands. Much energy can be saved by a proper utilisation of registers [75]. It was also noted that writes consume more energy, because a processor with a write-through cache (like the Intel 486) always causes an off-chip memory operation.

Secondary storage

Secondary storage in modern computers generally consists of a magnetic disk supplemented by a small portion of main memory RAM used as disk cache. Having the motor of the disk off saves energy. However, when it needs to be turned on again, it will take considerable time and energy to return to full operation.

The larger the cache, the better the performance. Energy consumption is reduced because data is kept locally, and thus requires less data traffic. Furthermore, the energy consumption is reduced because less disk and network activity is required. Unfortunately, there is a trade-off in the size of the cache memory since the required amount of additional DRAM can use as much energy as a constantly spinning hard disk [86].

Because of size, energy consumption and weight limitations of handheld machines a possible technology for secondary storage is flash memory. Like a hard disk, flash memory is non-volatile and can hold data without consuming energy. Furthermore, when reading or writing flash memory, it consumes only 0.15 to 0.47 W, far less than a

hard disk. It has read speed of about 85 ns per byte, quite like DRAM, but write speed of about 4-10 us, about 10-100 times slower than hard disk. However, since flash memory has no seek time, its overall write performance is not that much worse than a magnetic disk; in fact, for sufficiently small random writes, it can actually be faster [42]. Since flash is practically as fast as DRAM at reads, a disk cache is no longer important for read operations.

The cost per megabyte of flash is about 17-40 times more expensive than hard disk, but about 2-5 times less expensive than DRAM. Thus, flash memory might also be effective as a second level cache below the standard DRAM disk cache.

2.5.3 Programmability

Programmability is an attractive feature for any system, since it allows one system to be used for many applications. Programmability is even more important for mobile systems because they operate in a dynamically changing environment and must be able to adapt to the new environment. For example, a mobile computer will have to deal with unpredicted network outage or should be able to switch to a different network, without changing the application. It should therefore have the *flexibility* to handle a variety of multimedia services and standards (like different video decompression schemes and security mechanisms) and the *adaptability* to accommodate the nomadic environment, required level of security, and available resources [63][70].

The requirement for programmability in systems is also triggered by economical reasons. The high costs involved in designing and implementing a chip does not justify the design of a system that implements only a single application. Furthermore, because the requirements of applications are increasing rapidly, new chips need to be designed more often.

The structure of systems and applications is usually based on a modular design. This is needed to manage the complexity of the design, and also allows the designer to compose different applications of different combinations of the modules. The functions expressed by the modules typically consist of algorithms comprised of basic arithmetic operations (e.g. add, subtract, multiply). Descriptions in these basic operations we will call *fine-grain*, whereas descriptions of functions such as algorithms in which the complexity of the functions is large in comparison to the basic mathematical primitives will be called *coarse-grain*. Microprocessor designers are generally committed to the fine-grain level since they must provide generally applicable elementary operations. Implementing functions at the coarse-grain level usually yields sufficient flexibility for the application domain.

Each level of granularity has its own preferred and optimal application domain. Some of these levels are illustrated in Figure 10, which shows three different approaches in the spectrum of applications and hardware implementations.

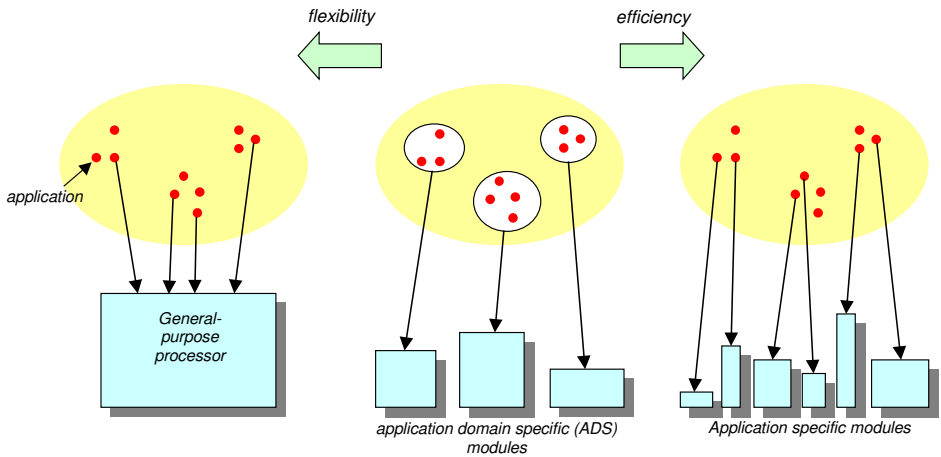


Figure 10: The spectrum of applications and hardware implementations [38].

General-purpose processors

There are two conflicting demands in the design of a high-performance architecture: efficiency and flexibility. The current trend is to focus on flexibility with *high performance general-purpose processors* as this is the area in which a semiconductor vendor can enhance its status [17][32]. Therefore, the architecture of a general-purpose processor is most widely studied, and optimisations for processor performance is the main goal. The advance in technology has lead to number of processor improvements like superscalar technology, VLIW architectures, reduce in cycle time, large on-chip caches, etc. With the general-purpose processors it is possible to map any type of application on the hardware simply by writing the right software for it. The only limitations are processing capacity and storage. It has been reported that 50 to 90 percent of the number of transistors in current high performance processors and future processor architectures is used for caches and main memory [32].

This flexibility and high-performance poses high demands on technology, and the resulting chips are usually large in terms of chip area and dissipate much energy. While general-purpose processors and conventional system architectures can be programmed to perform virtually any computational task, they have to pay for this flexibility with a high energy consumption and significant overhead of fetching, decoding and executing a stream of instructions on complex general-purpose data paths. General-purpose processors often have to perform tasks for which they are not ideally suited. Although they can perform such tasks, they will take longer, and will use more energy, than a custom hardware implementation (see Section 2.2.4). The energy overhead in making the architecture programmable most often dominates the energy dissipation of the intended computation. For example, when executing a typical DSP application program on the TMS320C5x family of general-purpose DSPs from Texas Instruments with a 20% mix of multiply-accumulate operations, instruction fetching and decoding is

responsible for 76% of the total supply current of the processing core [1]. Clearly a heavy penalty is paid for performing computations on a general-purpose data path under the control of an instruction stream.

Application specific modules

Another environment that will rapidly become more important in the near future is that of *application specific processors or modules*. The goal of these processors is to optimise the overall cost-performance of the system, and not performance alone. The modern application processor can use the technology to increase functionality to provide services such as multimedia devices, compression and decompression, network access, and security functions. Application specific solutions present the most effective way of reducing energy consumption and have been shown to lead to huge power savings [49][38].

Performing complex multimedia-data processing functions in dedicated hardware that is optimised for energy-efficient operation reduces the energy-per-operation by several orders of magnitude relative to software. Conventional general-purpose processors (e.g. Alpha, Pentium) focus entirely on instruction-per-second metrics, and typically require 100 mW/MIP; energy optimised general-purpose processors such as the StrongARM require 1-10 mW/MIP. Fully dedicated, optimised hardware, on the other hand requires less than 0.01 mW/MIP [84]. However, the disadvantage of dedicated hardware is the lack of flexibility and programmability, their functionality is restricted to the capabilities of the hardware. Current implementations of these systems have focussed on teleconferencing type applications and are limited to those. The wide range of multimedia applications being described in the literature cannot all be implemented with this specialised hardware [2].

Application domain specific modules

The difference in area and power dissipation between a general-purpose approach and application specific architectures can be significant. Furthermore, the technological challenges in the design of custom ASICs are usually significantly smaller than the design of general-purpose circuits. This means that high-performance custom chips can be designed and manufactured at relatively low cost. However, this comes at the price of less flexibility, and consequently a new chip design is needed for even the smallest change in functionality.

A hybrid solution with *application domain specific modules* should offer enough flexibility to be able to implement a predefined set of (usually) similar applications, while keeping the costs in terms of area, energy consumption and design time to an acceptable low level. The modules are optimised for one specific application domain. A system designer can use the general-purpose processor for portions of algorithms for which it is well suited, and craft an application domain specific module for other tasks. Unused parts of the system must be switched off when not needed. This is a good example of the difference between power and energy: although the application-specific

coprocessor may actually consume more power than the processor, it may be able to accomplish the same task in far less time, resulting in a net energy savings.

When the system is partitioned in multiple modules that each are able to perform some application-domain specific tasks, then *voltage-scaling* might become attractive. The key idea is that energy consumption can be reduced by first increasing the speed beyond the required timing constraints, then by reducing the voltage supply slowing them down until the timing constraints are not exactly met. As practical difficulties and cost were important obstacles in applying this technique at the technological level, at the system level the large size and small number of components may actually allow multiple (and adjustable) voltage supplies.

Reconfigurable computing

Reconfigurable computing systems combine programmable hardware with programmable processors to capitalise on the strengths of hardware and software [87]. While low-power solutions are already available for application specific problems, applying these solutions in a reconfigurable environment is a substantially harder problem, since programmable devices often incur significant performance and energy-consumption penalties [45][46]. To reduce the energy overhead in programmable architectures, the computational granularity should be matched to the architectural granularity. Performing multiplications on a FPGA is bound to carry huge amount of waste, so does executing large dot-vector products on a microprocessor.

2.5.4 Operating system

Up to this point, we have mainly discussed low-power techniques related purely to hardware components of the system. Software and algorithmic considerations can also have a severe impact on energy consumption. Digital hardware designers have promptly reacted to the challenge posed by low-power design. Designer skills, technology improvements and CAD tools have been successful in reducing the energy consumption. Unfortunately, software engineers and system architects are often less energy-conscious than digital designers, and they also lack suitable tools to estimate the energy consumption of their designs. As a result, energy-efficient hardware is often employed in a way that does not make optimal use of energy saving possibilities.

In this section we will show several approaches to reduce energy consumption at the operating system level and to the applications.

Dynamic power management

The essential characteristic of energy consumption for static CMOS circuits is that quiescent portions of a system dissipate a minimal amount of energy. Power management exploits periods of *idleness* caused by system under-utilisation. Especially in mobile systems, the utilisation is not constant. Designers naturally focus on worst-case conditions, peak performance requirements and peak utilisation. As a result, systems are often designed to operate under high utilisation, but they are actually fully

exploited during a relatively small fraction of their lifetime. Dynamic power management refers to the general class of techniques that manage the performance and throughput of a system based on its computational needs within the energy constraints.

We can identify two basic flavours of dynamic power management.

- *Binary power management*

  The most conservative and simple, although quite effective, is to deactivate some functional units when no computation is required. This can be done at different hierarchies and at different levels of design. The main problems involved in dynamic power management is the cost of restaring a powered down module or component. Restarting induces an increase in latency (e.g. time to restore a saved CPU state, spin-up of a disk), and possibly also an increase in energy consumption (e.g. due to higher start-up current in disks). The two main questions involved are then: 1) when to shut-down, and 2) when to wake-up.

  The activity of components in a computing system is usually *event driven*: for example the activity of display modules, communication interfaces, and user interface functions is triggered by external events and is often interleaved with long periods of quiescence. To take advantage of low-power states of devices, the operating system needs to direct (part of) the device to turn off (or down) when it is predicted that the net savings in power will be worth the time and energy overhead of turning off and restarting. Alternatively, the modules use a demand- or data-driven computation to automatically eliminate switching activity of unused modules. The trade-off is to justify the additional hardware and design complexity. The effectiveness is further determined by the extra energy required to implement this technique, and the time (and thus energy) that is required to determine when a module can be shut down (the so-called *inactivity threshold*). The inactivity threshold can be assigned statically or dynamically. In a *predictive* power management strategy the threshold is adapted according to the past history of active and idle intervals. The effectiveness at system level can be high because little additional hardware and design complexity is needed.

  Another question is *when to wake-up*, where the typical policy is to wake up in response to a certain event such as user interaction or network activity. The problem with such a demand policy is that waking up takes time, and the extra latency is not always tolerable. Again, a predictive approach, where the system initiates a wakeup in advance of the predicted end of an idle interval, often works better.

- *Dynamic power management of adaptive modules*

  More advanced power management schemes are based on a quality-of-service framework. Computer subsystems may be designed for multiple levels of reduced energy consumption at the expense of some other system performance measure (e.g. throughput). Key to the approach is a high degree of adaptability of the modules. What really matters in many cases is not sheer performance, but a balance of performance and availability. Users may tolerate performance degradation if functionality is provided for a longer period.

A hierarchical – QoS based – model of the whole system (covering the architecture, wireless communication, distributed processing, and applications) is of great importance for this technique. It is needed to adapt to the changing operating conditions dynamically in the most (energy) efficient way. Besides the functional modules and their ability to adapt (e.g. the effects on its energy consumption and QoS when the image compressor changes its frame rate, its resolution, or even its compression mechanism) this model also includes the interaction between these modules. Such a model should predict the overall consequences for the system when an application or functional module adapts its QoS. Using this model the inherent trade-offs between e.g. performance and energy consumption can be evaluated and a proper adaptation of the whole system can be made.

The whole system (hardware and software) should be designed taking power management into account. The division of the system into modules must be such that the modules provide a clustered functionality which are mutually exclusive and which can be idle for a large fraction of the operation time. For example, in the memory system, locality of reference can be exploited during memory assignment to induce an efficient and effective power down of large blocks of memory. This shows once again that a close co-operation between the operating system that performs the memory allocation, the energy manager that controls the power states of the devices, together with a suitable hardware architecture is crucial for the energy reduction of future mobile systems. Power management directly influences the design cycle, and is not only a matter of post-optimisation and tuning.

In order to control the modules, changes must be made to current architectures for hardware, drivers, firmware, operating system, and applications. One of the key aspects is to move power management policy decisions and co-ordination 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. This knowledge can be used and integrated in the Quality of Service model of the system.

The applicability of dynamic power management is not limited to the system level. The same concept has been exploited successfully at the logic level [52].

Scheduling

In a system scheduling is needed when multiple functional units need to access the same object. In operating systems scheduling is applied at several parts of a system for processor time, communication, disk access, etc. Currently scheduling is performed on criteria like priority, latency, time requirements etc. Power consumption is in general only a minor criterion for scheduling, despite the fact that much energy could be saved.

Subsystems of a computer, such as the CPU, the communication device, and storage system have small usage duty cycles. That is, they are often idle and wait for the user or network interaction. Furthermore, they have huge differences in energy consumption between their operating states (such as on, standby, sleep).

We will now show several possible mechanisms in which an energy efficient scheduling can be beneficial.

- *Processor time scheduling*

  Most systems spend only a fraction of the time performing useful computation, and the rest of the time is spent idling. The operating systems energy manager should track the periods of computation, so that when an idle period is entered, it can immediately power off major parts of the system that are no longer needed [5]. Since all power-down approaches incur some overhead, the task of an energy aware scheduler is to collect requests for computation and compact the active time-slots into bursts of computation.

  Experiments at UCLA with an X server and typical applications on a wireless terminal show that, in theory, a large reduction in CPU power consumption can be obtained if the terminal is shut down whenever it is idle [39]. They noticed that 96 to 98% of the time was spent in the blocked state, and that the average time in the blocked state is very short (much less than a second). Potential energy reduction is from 29 to 62 times.

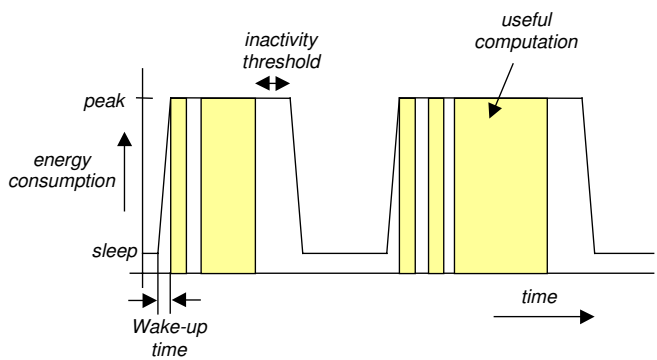


Figure 11: Power consumption in time of a typical processor system.

Weiser et al. [76] have proposed a system that reduces the cycle time of a processor for power saving, primarily by allowing the processor to use a lower voltage. For background and high latency tolerable tasks, the supply voltage can be reduced so that just enough throughput is delivered, which minimises energy consumption. By detecting the idle time of the processor, they can adjust the speed of the processor while still allowing the processor to meet its task completion deadlines. Suppose a task has a deadline of 100 ms, but needs only 50 ms of CPU time when running at full speed to complete. A normal system would run at full speed for 50 ms, and then be idle for 50 ms in which the CPU can be stopped. Compare this to a system that runs the task at half speed, so that it completes just before its deadline. If it can also reduce the voltage by half, then the task will consume a quarter of the energy of the normal system. This is because the same number of cycles are executed in both systems, but the modified system reduces energy use by reducing the operating voltage.

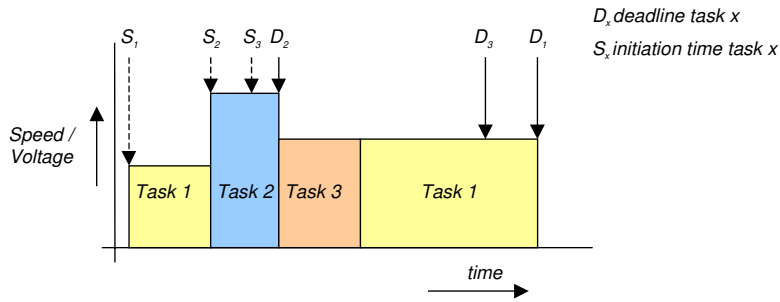


Figure 12: Voltage scheduling under deadline constraints.

Weiser et al. classified idle periods into 'hard' and 'soft' events. Obviously, running slower should not allow requests for a disk block to be postponed. However, it is reasonable to slow down the response to a keystroke, such that processing of one keystroke finishes just before the next. Another approach is to classify jobs or processes into classes like background, periodic and foreground. With this sort of classification the processor can run at a lower speed when executing low priority background tasks only.

- *File system*

  The file system is another issue in the interaction between hardware facilities for power management and the system software.

  Several people have investigated powering down disk drives on portable computers to conserve energy (e.g. [14][86]). Shutting down a disk is a relatively easy task, but unfortunately turning it on is much more expensive in time and energy. Golding et al. tackle the general problem of finding the best way to exploit idle periods during operations of a hard disk [21]. A power management policy that spins down the disk when it is idle can be successful only if it can predict with sufficient accuracy the start time and the duration of the idle interval. The main feature of their model is that it is convenient to spin down the disk as long as it remains in sleep for a period of time longer than a threshold T_{min}.

  It is one thing to turn off a disk when it has been idle for some time, but it is much better to design a file system in such a way that it takes advantage of the possibility of turning the disk off. For example the operating system's file system a scheduler can try to collect disk operations in a cache and postpone low priority disk I/O only until the hard drive is running already or has enough data.

- *Battery relaxation*

  In recent years batteries have become smaller and they have got more capacity. The capacity of the battery is strongly influenced by the available relaxation time between periods of operation. The relationship between how much of the battery capacity is recovered during an off period depends on the cell chemistry and geometry. By taking into consideration the dynamic charge recovery, it is possible

for most types of batteries to get more out of a given battery. In [78] the authors studied cylindrical alkaline cells subject to a periodically pulsed current discharge, and found that the cell capacity increases as the duty cycle decreases and the frequency increases. When the system has knowledge of these battery characteristics, the behaviour and energy demands of the system can be adapted such that it tries to discharge the battery only when completely recovered from the previous discharge. However, this may not be synchronised with the current demand of the application.

The energy efficiency of communication protocols can be enhanced through the use of communication protocols that exploit the charge recovery mechanism [13]. Primarily delaying some power consuming activity such as transmission of a packet will perform the discharge demand shaping. Intuitively, it seems clear that the fundamental trade-off here is between delay and energy efficiency.

System decomposition

System decomposition can be applied at various levels: at the computer system internally, or externally by a functional partitioning between a wireless terminal and the network infrastructure. Let us first consider the *internal system decomposition,* that is decomposition of functionality within the mobile.

In a mobile multimedia system many trade-offs can be made concerning the required functionality of a certain mechanism, its actual implementation, and values of the required parameters. In an architecture with reconfigurable modules and data streams, functions can be dynamically migrated between functional modules such that an efficient configuration is obtained. For example, when we consider the transmission of an image over a wireless network, there is a trade-off between image compression, error control, communication, and energy consumption. Functionality can be partitioned between a program running on the general-purpose CPU, dedicated hardware components (like a compressor or error correction device), and field programmable hardware devices (like FPGAs). Of course, the actual trade-off will depend on the particularities of the system, the nature of the data sent, and so on.

The networked operation of a mobile system opens up additional opportunities for decomposition to increase energy efficiency. A careful analysis of the data flow in the system and decomposition of the system functions between wireless terminal and network infrastructure can reduce energy consumption considerably. One opportunity is offloading computation dynamically from the mobile system, where battery energy is at a premium, to remote energy-rich servers in the wired backbone of the network. In essence, energy spent in communication is traded for computation. Partitioning of functions is an important architectural decision, which indicates where applications can run, where data can be stored, the complexity of the terminal, and the cost of the communication service [39]. The key implication for this architecture is that the runtime hardware and software environment on the mobile computer and in the network should be able to support such adaptability, and provide application developers with appropriate interfaces to control it. Software technologies such as proxies and mobile agents, and

hardware technologies such as adaptive and reconfigurable computing are likely to be the key enablers.

A good example of such a decomposition that partitions the computational effort for *video compression* is described by Rabiner [60][61]. Due to the large amount of data needed for video traffic, efficient compression techniques are important when the video frames are transmitted over wireless channels. Motion estimation has been shown to help significantly in the compression of video sequences. However, since most motion estimation algorithms require a large amount of computation, it is undesirable to use them in power constrained applications, such as battery operated wireless video terminals. Since the location of an object in the current frame can be predicted from its location in previous frames, it is possible to optimally partition the motion estimation computation between battery operated portable devices and high powered compute servers on the wired network.

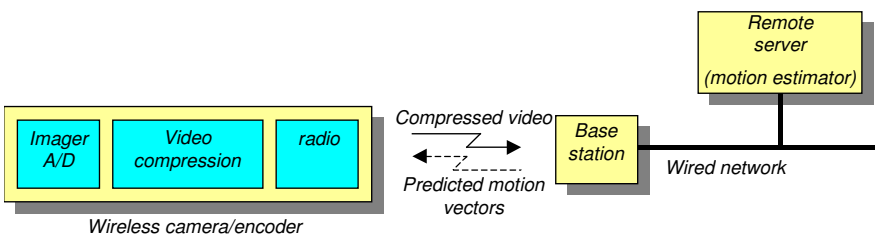


Figure 13: Partitioned video compression.

Figure 13 shows a block diagram of a wireless video terminal in a networked environment. A resource on the wired network with no power constraints can estimate the motion of the sequence based on the previous (decoded) frames and use this information to predict the motion vectors of the current frame. This can achieve a reduction in the number of operations performed at the encoder for motion estimation by over two orders of magnitude while introducing minimal degradation to the decoded video compared with full search encoder-based motion estimation.

Intelligent agents, sometimes called *proxies*, can be used to process control information, or to manipulate user information that is being exchanged between the mobile device and a network-based server. A proxy can be executed in a fixed location, or it may be mobile, and move with the user that it serves. The general benefits of a network-based proxy are that it can execute complex functions, perform aggressive computation and use large amounts of storage on behalf of clients. One example of a complex function that may be performed by a mobile device is the format translation of information sent for display. For example, a wireless web browser may provide a picture in 16 bits colour while the device can only display black and white. To have an end device perform this conversion requires a significant amount of storage and processing. Instead a proxy may perform this conversion, filter out any unusable graphics, and forward only the black and white pixels to the display. The Wireless Application Protocol (WAP) uses a similar approach to be able to support Internet content and services on differing wireless network technologies and device types [88].

Energy is saved because the amount of computation and communication for the mobile is reduced. Other advantages of proxies are that proxies may account for mobiles that are in a disconnected state, and that they provides a solution to the problem of client and network heterogeneity and allow interoperability with existing servers [18][22].

Network-based proxies may thus be used to perform various functions in lieu of the mobile. However, for applications and protocols developed specifically for wireless mobile devices, solutions inherent in their design may be more efficient. For example, the network protocols such as TCP/IP can be implemented more energy efficiently. Certain parts of the network protocol stack can be migrated to servers residing on the fixed network (i.e. the base station). For example, a base station could handle parts of the network protocol stack in lieu of the mobile. The remote server has a private dedicated communication protocol with the mobile so that the mobile units can use an internal, lightweight, protocol to communicate with the base station rather than TCP/IP or UDP. The net result is saving in code and energy. In Section 2.5.6 we will exploit the possibilities of energy reduction in communication in more detail.

In the early phases of the design of any part of the system, either hardware or software, the designer needs to experiment with alternative designs. However, energy efficiency is not only a one-time problem that needs to be solved during the design phase. When the system is operational, frequent adaptations to the system are required to obtain an energy efficient system that can fulfil the requirements imposed in terms of a general QoS model. Finding the energy management policy that minimises energy consumption without compromising performance beyond acceptable levels is already a complex problem. If the resources are also flexible, and can adapt their functionality, this problem becomes even bigger.

2.5.5 Applications, compilation techniques and algorithms

Applications

The best policy for deciding when to shut down or wake up a specific part of the system is in general application-dependent, since applications are in the best position to know usage and activity patterns for various resources. Applications play a critical role in the user's experience of a power-managed system. Power management circuits and operating systems that lack application-specific knowledge can only rely on the generic policies discussed above. In traditional power-managed systems, the hardware attempts to provide automatic power management in a way that is transparent to the applications and users. This has resulted in some legendary user problems such as screens going blank during video or slide-show presentations, annoying delays while disks spin up unexpectedly, and low battery life because of inappropriate device usage. Because the applications have direct knowledge of how the user is using the system to perform some function, this knowledge must penetrate into the power management decision-making system in order to prevent the kinds of user problems described above.

This suggests that operating systems ought to provide application programming interfaces (APIs) so that energy-aware applications may influence the scheduling of the

system's resources. Obviously, careless application's use of the processor and hard disk drastically affects battery lifetime. For example, performing non-essential background tasks in the idle loop prevents the processor from entering a low power state (see for example [41]). So, it is not sufficient to be low power, but the applications running for a system have to be made energy aware as well.

Prefetching and caching of data has been used to improve performance in many applications and file systems. In a mobile environment, these techniques are used by many systems to limit communication and energy consumption caused by mobility, and to improve performance and availability of services. In [79] two systems have been discussed: a file system application and a browsing application.

Code and algorithm transformation

Software design for low power has become an active area of research in the last few years. Software optimisation that properly selects and orders the instructions of a program to reduce the instruction bus activity are based on the simple observation that a given high-level operation can be compiled into different machine instruction sequences. As much of the power consumed by a processor is due to the fetching of instructions from memory, high code density, or even instruction compression, can reduce energy consumption. This is not only because fewer processor cycles are required for a given function, but also because denser code means better cache performance. So, the reduction in bus traffic may be better than linear with decreasing code size [69]. However, this only works well when the execution cycle is not (much) longer. Luckily, power and performance can be improved at the same time by optimising the software. If an algorithm can be optimised to run in fewer cycles, it is faster, and consumes less energy.

Today, the cost function in most compilers is either speed or code size, so the most straightforward way to proceed is to modify the objective function used by existing code optimisers to obtain low-power versions of a given software program. The energy cost of each instruction (determined a priori) must be considered during code optimisation. An energy aware compiler has to make a trade-off between size and speed in favour of energy reduction.

The energy consumed by a processor depends on the previous state of the system and the current inputs. Thus, it is dependent on instruction choice and instruction ordering. Reordering of instructions can reduce the switching activity and thus overall energy consumption. However, it was found not to have a great impact [75][64]. Research has shown that improvements that can be gained using ARM compiler optimisations are marginal compared to writing more energy efficient source code [64]. The largest energy savings are observed at the inter-procedural level that compilers have not been able to exploit. For DSP processors, low-power compilation techniques have produced more interesting results. In particular, it was shown in [36] that instruction scheduling has sizeable impact on global power consumption. This difference can be explained because in DSP processors the energy consumption is dominated by the functional units in the data-path, hence there is a strong correlation between the executed instruction and the execution unit involved in the computation.

Another technique that can be applied is to reduce the cost of memory accesses. Again, this is a similar objective of current compilers developed in the context of high-performance code generation. Further improvements in the power budget can be achieved by applying techniques that explicitly target the minimisation of the switching activity on the address bus and that best exploit the hierarchy in the memory system as described above.

Recent work has shown that with approaches as described above, a particular Fujitsu digital signal processor could run on 26 percent to 73 percent less power – with no hardware changes [37]. They did it by, among other things, carefully assigning data to specific memory banks and by using packed, instead of unpacked instructions. In the former technique, if operands will be needed together for computations, they are assigned to the same memory bank to take advantage of a double-operand move operation out of the one memory. Since the double-operand move takes only a single cycle, instead of two cycles for two single-operand moves, the access draws less power. For the same reason, using packed, instead of unpacked, instructions also consumes less power: instructions are chosen that reduce the number of execution cycles and so are fundamentally more efficient.

At the algorithm level functional pipelining, retiming, algebraic transformations and loop transformations can be used [49]. The system's essential power dissipation can be estimated by a weighted sum of the number of operations in the algorithm that has to be performed [10]. The weights used for the different operations should reflect the respective capacitance switched. The size and the complexity of an algorithm (e.g. operation counts, word length) determine the activity. Operand reduction includes common sub-expression elimination, dead code elimination etc. Strength reduction can be applied to replace energy consuming operations by a combination of simpler operations (for example by replacing multiplications into shift and add operations). Drawbacks of this approach are that it introduces extra overhead for registers and control, and that it may increase the critical path [43].

2.5.6 Energy reduction in communication

Up to this point we have mainly discussed the techniques that can be used to decrease the energy consumption of digital systems and focussed on computer systems. In this subsection we will discuss some techniques that can be used to reduce the energy consumption that is needed for the (wireless) communication external of the computer. In [24] we give more detailed information.

Sources of energy consumption

In its most abstract form, a networked computer system has two sources of energy drain during operation:

- *Communication*, due to energy spent by the wireless interface. Communication energy is, among others, dictated by the signal-to-noise ratio (SNR) requirements.

- *Computation*, due to (signal) processing and other tasks required during communication. Computation energy is a function of the hardware and software used for tasks such as compression and forward error correction (FEC).

Broadly speaking, minimising energy consumption is a task that will require minimising the contributions of communication and computation, making the appropriate trade-offs between the two. For example, reducing the amount of transmitted data may be beneficial. On the other hand, the computation cost (e.g. to compress the data being sent) might be high, and in the extreme it might be such that it would be better to just send the raw data.

For long distance wireless links, the transmit-communication energy component dominates. However, for short distance wireless links and in harsh environments where much signal processing and protocol computation may be used, the computation component can be significant or dominant.

The wireless network interface of a mobile computer consumes a significant fraction of the total power [73]. Measurements show that on typical applications like web-browsing or handling e-mail, the energy consumed while the interface is 'on' and idle is more than the cost of actually receiving packets. That is because most applications have little demanding traffic needs, and hence the transceiver is *idling* most of the time. The access to the wireless channel is controlled by a MAC protocol. Many MAC protocols for wireless networks are basically adaptations of MAC protocols used in wired networks, and ignore energy issues [39]. For example, random access MAC protocols such as carrier sense multiple access with collision avoidance (CSMA/CA) and 802.11 typically require the receiver to be powered on continually and monitor the channel for traffic. The typical *inactivity threshold*, which is the time before a transceiver will go in the off or standby state after a period of inactivity, causes the receiver to be needlessly in an energy consuming mode for a significant time. Significant time and energy is further spent by the mobile in switching from transmit to receive modes, and vice-versa. In broadcast networks *collisions* may occur (during high load situations). This causes the data to become useless and the energy needed to transport that data to be wasted.

The next step is to *reduce the amount of data*, which must be pushed through the channel. This goal can be reached in a number of ways. One is to reduce the *overhead of a protocol* which influences the energy requirements due to the amount of 'useless' control data and the required computation for protocol handling. The high *error rate* that is typical for wireless links is another source of energy consumption for several reasons. First, when the data is not correctly received the energy that was needed to transport and process that data is spoiled. Secondly, energy is used for error control mechanisms. Finally, because in wireless communication the error rate varies dynamically over time and space, a fixed-point error control mechanism that is designed to be able to correct errors that hardly occur, spoils energy and bandwidth. If the application is error-resilient, trying to withstand all possible errors spoils even more energy for needless error control. Reducing the amount of data is also an application-layer issue. For example, the application might change the compression rate or possibly reduce the data resolution. Instead of sending an entire large full-colour image, one can send black-and-white half-size images with lossy compression.

Network protocol stack

Data communication protocols govern the way in which electronic systems exchange information by specifying a set of rules that, when followed, provide a consistent, repeatable, and well-understood data transfer service. In designing communication protocols and the systems that implement them, one would like to ensure that the protocol is correct and efficient.

Portable devices have severe constraints on the size, the energy consumption, the communication bandwidth available, and are required to handle many classes of data transfer over a limited bandwidth wireless connection, including delay sensitive, real-time traffic such as speed and video. Multimedia applications are characterised by their various media streams. Each stream can have different quality of service requirements. Depending on the service class and QoS of a connection a different policy can be applied to the communication protocol by the application to minimise energy consumption. For example, by avoiding error-control overhead for connections that do not need it and by never transmitting stale data, efficiency is improved. This combination of limited bandwidth, high error rates, and delay-sensitive data requires tight integration of all subsystems in the device, including aggressive optimisation of the protocols that suit the intended application. The protocols must be robust in the presence of errors; they must be able to differentiate between classes of data, giving each class the exact service it requires; and they must have an implementation suitable for low-power portable electronic devices.

In order to save energy a normal mode of operation of the mobile will be a sleep or power down mode. To support full connectivity while being in a deep power down mode the network protocols need to be modified. Store-and-forward schemes for wireless networks, such as the IEEE 802.11 proposed sleep mode, not only allow a network interface to enter a sleep mode but can also perform local retransmissions not involving the higher network protocol layers. However, such schemes have the disadvantage of requiring a third party, e.g. a base station, to act as a buffering interface. This example, however, shows that the network protocols of a wireless system can be changed in such a way that it minimises its energy consumption.

Considerations of energy efficiency are fundamentally influenced by the trade-off between energy consumption and achievable Quality of Service (QoS). With the provision of universal roaming, a mobile user will be faced with an environment in which the quality of service can vary significantly within and across different wireless networks. In order to deal with the dynamic variations in networking and computing resources gracefully, both the mobile computing environment and the applications that operate in such an environment need to *adapt* their behaviour depending on the available resources including the batteries. Energy reduction should be considered in the whole system of the mobile and through all layers of the protocol stack, including the application layer. Adaptability of the protocols is a key issue. We will now provide various ways that can be used to reduce energy consumption at the layers of a typical network protocol stack.

- *Physical layer* – At the lowest level we need to apply an energy-efficient radio that can be in various operating modes (like variable RF power and different sleep modes) such that it allows a dynamic power management. Energy can also be saved if it is able to adapt its modulation techniques and basic error-correction schemes. The bandwidth offered by the radio also influences its energy consumption. The energy per bit transmitted or received tends to be lower at higher bit rates. For example, the WaveLAN radio operates at 2Mb/s and consumes 1.8 W, or 0.9 μJ/bit. A commercially available FM transceiver (Radiometrix BIM-433) operates at 40 kb/s and consumes 60 mW, or 1.5 μJ/bit. This makes the low bit-rate radio less efficient in energy consumption for the same amount of data. However, when a mobile has to listen for a longer period for a broadcast or wake-up from the base station, then the high bit-rate radio consumes about 30 times more energy than the low bit rate radio. Therefore, the low bit-rate radio must be used for the basic signalling only, and as little as possible for data transfer.

  To minimise the energy consumption, but also to mitigate interference and increase network capacity, the transmit power on the link should be minimised, if possible.

- *Medium access layer* – In an energy efficient MAC protocol the basic objective is to minimise all actions of the network interface, i.e. minimise 'on-time' of the transmitter as well as the receiver. Another way to reduce energy consumption is by minimising the number of transitions the wireless interface has to make. By scheduling data transfers in bulk, an inactive terminal is allowed to doze and power off the receiver as long as the network interface is reactivated at the scheduled time to transceive the data at full speed. An example of an energy-efficient MAC protocol is E^2MaC [26]. This is a TDMA protocol in which the QoS manager at the base-station schedules all traffic according to the QoS requirements and tries to minimise the energy consumption of the mobiles. The E^2MaC protocol is a subject of Chapter 5.

- *Logical Link Control layer* – Due to the dynamic nature of wireless networks, *adaptive error control* gives significant gains in bandwidth and energy efficiency [86][68]. This avoids applying error-control overhead to connections that do not need it, and it allows to selectively match the required QoS and the conditions of the radio link. Above these error control adaptations, a scheduler in the base-station can also adapt its traffic scheduling to the error conditions of wireless connections to a mobile. The scheduler can try to avoid periods of bad error conditions by not scheduling non-time critical traffic during these periods.

  Flow control mechanisms are needed to prevent buffer overflow, but also to discard packets that have exceeded the allowable transfer time. Multimedia applications are characterised by their various media streams. Each stream can have different quality of service requirements. Depending on the service class and QoS of a connection a different flow control can be applied so that it minimises the required bandwidth and energy consumption. For instance, in a video application it is useless to transmit images that are already outdated. It is more important to have the 'fresh' images. For such traffic the buffer is probably small, and when the connection is hindered

somewhere, the oldest data will be discarded and the fresh data will be shifted into the fifo. Flow control would needlessly spend energy for transmitting 'old' images and flow-control messages. An energy-efficient flow control adapts it control mechanism to the requirements of the connection.

- *Network layer* – Errors on the wireless link can be propagated in the protocol stack. In the presence of a high packet error rate and periods of intermittent connectivity of wireless links, some network protocols (such as TCP) may overreact to packet losses, mistaking them for congestion. TCP responds to all losses by invoking congestion control and avoidance algorithms. These measures result in an unnecessary reduction in the link's bandwidth utilisation and increases in energy consumption because it leads to a longer transfer time. The limitations of TCP can be overcome by a more adequate congestion control during packet errors. These schemes choose from a variety of mechanisms to improve end-to-end throughput, such as local retransmissions, split connections and forward error correction. In [4] several schemes have been examined and compared. These schemes are classified into three categories: end-to-end protocols, where the sender is aware of the wireless link; link-layer protocols, that provide local reliability and shields the sender from wireless losses; and split-connection protocols, that break the end-to-end connection into two parts at the base station. Their results show that a reliable link-layer protocol with some knowledge of TCP provides good performance, more than using a split-connection approach. Selective acknowledgement schemes are useful, especially when the losses occur in bursts.
- *Operating system level* – Another way to avert the high cost (either performance, energy consumption or money) of wireless network communication is to avoid use of the network when it is expensive by predicting future access and fetching necessary data when the network is cheap. In the higher level protocols of a communication system caching and scheduling can be used to control the transmission of messages. This works in particular well when the computer system has the ability to use various networking infrastructures (depending on the availability of the infrastructure at a certain locality), with varying and multiple network connectivity and with different characteristics and costs [16]. True prescience, of course, requires knowledge of the future. Two possible techniques, *LRU caching* and *hoarding*, are for example present in the Coda cache manager [30]. In order to effectively support mobile computers, system designers must view the network as a first-class resource, expending CPU and possibly disk resources to reduce the use of network resources during periods of poor network connectivity.

Modern high-performance network protocols require that all network access be through the operating system, which adds significant overhead to both the transmission path (typically a system call and data copy) and the receive path (typically an interrupt, a system call, and a data copy). This not only causes performance problems, but also incurs a significant of energy consumption. Intelligent network interfaces can relieve this problem to some extend. To address the performance problem, several *user-level communication architectures* have been developed that remove the operating system from the critical communication path [8].

System decomposition

In normal systems much of the network protocol stack is implemented on the main processor. Thus, the network interface and the main processor must always be 'on' for the network to be active. Because almost all data is transported through the processor, performance and energy consumption is a significant problem.

In a communication system locality of reference can be exploited by decomposition of the network protocol stack and cautious management of the data flow. This can reduce the energy consumption for several reasons:

- First, when the system is constructed out of independent components that implement various layers of the communication stack, unnecessary data copies between successive layers of the protocol stack may be eliminated. This eliminates wasteful data transfers over the global bus, and thus saves much dissipation in buses, multiplexers and drivers. Note, however, that modern, optimised systems, like the x-kernel, avoid data copies between protocol layers religiously.
- Secondly, dedicated hardware can do basic signal processing and can move the necessary data directly to its destination, thus keeping data copies off of the system bus. Moreover, this dedicated hardware might do its tasks much more energy efficiently than a general-purpose processor.
- Finally, a communications processor can be applied to handle most of the lower levels of the protocol stack, thereby allowing the main processor to sleep for extended periods of time without affecting system performance or functionality.

This decomposition can also be applied beyond the system level of the portable: certain functions of the system can be migrated from the portable system to a remote server that has plenty of energy resources. This remote server handles those functions that can not be handled efficiently on the portable machine. For example, a base station could handle parts of the network protocol stack in lieu of the mobile. The remote server has a private dedicated communication protocol with the mobile so that the mobile units can use an internal, lightweight, protocol to communicate with the base station rather than TCP/IP or UDP. The net result is saving in code and energy.

Low power short range networks

Portable 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. Applications require that networks are able to determine that the mobile has moved from one network to another network with a possible different QoS. The network that is most appropriate in a certain location at a certain time depends on the user requirements, network bandwidth, communication costs, energy consumption etc. The system and the applications might adapt to the cost of communication (e.g. measured in terms of ampère-hours or telephone bills).

Over short distances, typically of up to five metres, high-speed, low-energy communication is possible [68]. Private houses, office buildings and public buildings can be fitted with 'micro-cellular' networks with a small antenna in every room at

regular intervals, so that a mobile computer never has to communicate over a great distance – thus saving energy – and in such a way that the bandwidth available in the aether does not have to be shared with large numbers of other devices – thus providing high aggregate bandwidth. Over large distances (kilometres rather than metres), the mobile can make use of the standard infrastructures for digital telephony (such as GSM).

2.6 Conclusions

As there will become an increasing numbers of portable, battery powered systems, more and more attention will be focused on low-power design techniques. The art of low-power design used to be a narrow speciality in analog circuit design. As the issue of energy efficiency becomes even more pervasive, the battle to use the bare minimum of energy will be fought on multiple fronts: semiconductor technology, circuit design, design automation tools, system architecture, operating system, and application design. It is now appearing in the mainstream digital design community affecting all aspects of the design process. Eventually, the concern for low-power design will expand from devices to modules to entire systems, including application software. At technological and architectural level energy consumption can be decreased by reducing the supply voltage, reducing the capacitive load and by reducing the switching frequency. Much profit can be gained by avoiding unnecessary activity at both the architectural and system level. At system level, the system designer can take advantage of power management features where available, as well as decomposed system architectures and programming techniques for reducing power consumption.

Note that some low-power design techniques are also used to design high-speed circuits, and to increase performance. For example, optimised code runs faster, is smaller, and therefore also consumes less energy. Using a cache in a system not only improves performance, but – although requiring more space – uses less energy since data is kept locally. The approach of using application-specific coprocessors is not only more efficient in terms of energy consumption, but has also a performance increase because the specific processors can do their task more efficient than a general-purpose processor. Energy-efficient asynchronous systems also have the potential of a performance increase, because the speed is no longer dictated by a clock, but is as fast as the flow of data.

However, some trade-offs need to be made. Most energy-efficient systems use more area, not only to implement a new data flow or storage, but also to implement the control part. Furthermore, energy-efficient systems can be more complex. Another consequence is that although the application-specific coprocessor approach is more efficient than a general-purpose processor, it is less flexible. Furthermore, the latency from the user's perspective is increased, because a system in sleep has to be wakened up. For instance, spinning down the disk causes the subsequent disk access to have a high latency.

Programmability is important for mobile systems because they operate in a dynamically changing environment and must be able to adapt to the new environment. While low-

power solutions are already available for application specific problems, applying these solutions in a reconfigurable environment is a substantially harder problem, since programmable devices often incur significant performance and energy-consumption penalties. The research described in this thesis mainly focuses on higher level re-programmability. The *Chameleon* project [66], that just started and is a spin-off from the Moby Dick project and the research presented in this thesis, will deal with reconfiguration in more depth.

Applications play a critical role in the user's experience of a power-managed system. Therefore, the application and operating system must allow a user to guide the power management.

Any consumption of resources by one application might affect the others, and as resources run out, all applications are affected. Since system architecture, operating system, communication, energy consumption and application behaviour are closely linked, we believe that a QoS framework can be a sound basis for integrated management of all resources, including the batteries.

References

[1] Abnous A., Rabaey J.: “Ultra-low-power domain-specific multimedia processors”, *VLSI Signal processing IX*, ed. W. Burleson et al., IEEE Press, pp. 459-468, November 1996.

[2] Adam J.: “Interactive multimedia – applications, implications”, *IEEE Spectrum*, pp. 24-29, March 1993.

[3] Bakoglu H. “Circuits, Interconnections, and Packaging for VLSI”, *Addison-Wesley*, Menlo Park, CA, 1990.

[4] Balakrishnan H., et al.: “A comparison of mechanisms for improving TCP performance over wireless links”, Proceedings *ACM SIGCOMM'96*, Stanford, CA, USA, August 1996.

[5] Benini L., De Micheli G.: “Dynamic Power Management, design techniques and CAD tools”, *Kluwer Academic Publishers*, ISBN 0-7923-8086-X, 1998.

[6] Berkel K., et al.: “A fully asynchronous low power error corrector for the DCC player”, *Digest of Technical Papers, International Solid-State Circuit Conference*, pp. 88-89, 1994.

[7] Berkel K. van, Rem M.: “VLSI programming of asynchronous circuits for low power”, *Nat.Lab. Technical Note Nr. UR 005/94*, Philips Research Laboratories, Eindhoven, the Netherlands, 1994.

[8] Bhoedjang, R.A.F., Rühl T., Bal H.E.: “User-level network interface protocols”, *Computer*, November 1998, pp. 53-60.

[9] Burd T.D., Brodersen R.W.: “Energy efficient CMOS microprocessor design”, *Proceedings 28th. annual HICSS Conference*, Jan. 1995, vol. I, pp. 288-297.

[10] Burger D., Goodman J.: “Billion-transistor architectures”, *Computer*, Sept. 1997, pp. 46-47.

[11] Chandrakasan A.P., et al.: “Optimizing power using transformations”, *Transactions on CAD*, Jan. 1995.

[12] Chao K., Wong D.: “Low power considerations in floorplan design”, *1994 International Workshop on low power design*, Napa Valley, CA, pp.45-50, April 1994.

[13] Chiasserini C.F., Rao R.R.: "Pulsed battery discharge in communication devices", *proceedings ACM/IEEE MobiCom'99*, pp. 88-95, August 1999.

[14] Douglis F., Krishnan P., Marsh B.: “Thwarting the power-hungry disk”, *proceedings of USENIX Winter 1994 Technical Conference*, pp.292-306, USENIX Association, 17-21 January 1994.

[15] Dyer C.K.: "Replacing the battery in portable electronics", *Scientific American*, pp. 70-75, July 1999.

[16] Ebling, M.R., Mummert, L.B., Steere D.C.: “Overcoming the Network Bottleneck in Mobile Computing”, *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, Dec. 1994, Santa Cruz, CA.

[17] Flynn M.J.: “What's ahead in computer design?”, *Proceedings Euromicro 97*, pp. 4-9, September 1997.

[18] Fox A., Gribble S.D., Chawathe Y., Brewer E.A.: "Adapting to network and client variation using infrastructural proxies: lessons and perspectives, *IEEE Personal Communications*, pp. 10-19, August 1998.

[19] Frenkil J.: "A multi-level approach to low-power IC design", *IEEE Spectrum*, Volume 35, Number 2, February 1998.

[20] Gao B., Rees D.J.: "Communicating synchronous logic modules", *proceedings 21st Euromicro conference*, September 1995.

[21] Golding R., Bosch P., Staelin C., Sullivan T., Wilkes J.: "Idleness is not sloth", Winter'95 USENIX Conference proceedings, New Orleans, Jan. 1995.

[22] Han R., Bhagwat P., LaMaire R., Mummert T., Perret V., Rubas J.: "Dynamic adaptation in an image transcoding proxy for mobile web browsing", *IEEE Personal Communications*, pp. 8-17, December 1998.

[23] Hauck S.: "Asynchronous design methodologies: an overview", *Proceedings of the IEEE*, Vol. 83, No. 1, pp. 69-93, January 1995.

[24] Havinga, P.J.M., Smit, G.J.M.: "Minimizing energy consumption for wireless computers in Moby Dick", *proceedings IEEE International Conference on Personal Wireless Communication ICPWC'97*, Dec. 1997.

[25] Havinga P.J.M., Smit G.J.M.: "Design techniques for low power systems" *Journal of Systems Architecture*, Vol. 46, Iss. 1, 2000, a previous version appeared as CTIT Technical report, No. 97-32, Enschede, the Netherlands, ISSN 1381-3625

[26] Havinga P.J.M., Smit G.J.M., Bos M.: "Energy-efficient wireless ATM design", *proceedings wmATM'99*, June 2-4, 1999.

[27] Ikeda T.: "ThinkPad Low-Power Evolution", *IEEE Symposium on Low Power Electronics*, October 1994.

[28] Intel486SX: URL: http://134.134.214.1/design/intarch/prodbref/272713.htm.

[29] Kin J., Gupta M., Mangione-Smith W.H.: "The filter cache: an energy efficient memory structure", *Micro* 30, 1997.

[30] Kistler J.J.: "Disconnected operation in a distributed file system", PhD thesis, *Carnegie Mellon University*, School of Computer Science, 1993.

[31] Koegst, M, et al.: "Low power design of FSMs by state assignment and disabling self-loops", *Proceedings Euromicro 97*, pp 323-330, September 1997.

[32] Kozyrakis C.E., Patterson D.A.: "A new direction for computer architecture research", *Computer*, Nov. 1998, pp. 24-32

[33] Landman P.E.: "Low-power architectural design methodologies", Ph.D. thesis, *University of California at Berkeley*, 1994.

[34] Lapsley, P: "Low power programmable DSP chips: features and system design strategies", *Proceedings of the International Conference on Signal Processing*, Applications and Technology, 1994.

[35] Larri G.: "ARM810: Dancing to the Beat of a Different Drum", *Hot Chips 8: A Symposium on High-Performance Chips*, Stanford, August 1996.

[36] Lee M.T.-C et al. V. Tiwari et al. : "Power analysis and low-power scheduling techniques for embedded DSP software", *International Symposium on System Synthesis*, pp. 110-115, Sept. 1995.

[37] Lee M.T.-C, Tiwari V., Malik S., Fujita M. “Power Analysis and Minimization Techniques for Embedded DSP Software”, *IEEE Transactions on VLSI Systems*, March 1997, Vol. 5, no. 1, pp. 123-133.

[38] Leijten J.A.J.: “Real-time constrained reconfigurable communication between embedded processors”, *Ph.D. thesis, Eindhoven University of Technology*, November 1998.

[39] Lettieri P., Srivastava M.B.: “Advances in wireless terminals”, *IEEE Personal Communications*, pp. 6-19, February 1999

[40] Lorch, J.R.,: “A complete picture of the energy consumption of a portable computer”, *Masters thesis, Computer Science, University of California at Berkeley*, 1995

[41] Lorch, J.R., Smith, A.J.: “Reducing power consumption by improving processor time management in a single user operating system”, *Proceedings of 2nd ACM international conference on mobile computing and networking*, Rye, November 1996.

[42] Lorch, J.R., Smith, A.J.: "Software strategies for portable computer energy management", Report No. UCB/CSD-97-949, Computer Science Division (EECS), *University of California at Berkeley*, May 1997.

[43] Macii, E., Pedram M., Somenzi F.: “High-level power modeling, estimation, and optimization”, *IEEE transactions on computer-aided design of integrated circuits and systems*, Vol. 17, No. 11, pp. 1061-1079, November 1998.

[44] Mangione-Smith, B. 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 (ISLPED) 1996*, Monterey CA, USA, pp. 23-28, August 1996.

[45] Mangione-Smith W.H., et al.: “Seeking solutions in configurable computing”, *IEEE Computer*, pp. 38-43, December 1997.

[46] Mangione-Smith W.H., Hutchings B.L.: “Configurable computing: the road ahead”, *1997 reconfigurable architectures workshop*, 1997.

[47] Martin A.J., Burns S.M., Lee T.K., Borkovic D., Hazewindus P.J.: “The first asynchronous microprocessor: the test results”, *Computer Architecture News*, 17(4):95-110, June 1989.

[48] McGaughy, B: “Low Power Design Techniques and IRAM”, March 20, 1996, URL: http://rely.eecs.berkeley.edu:8080/researchers/brucemcg/iram_hw2.html.

[49] Mehra R., Lidsky D.B., Abnous A., Landman P.E., Rabaey J.M.: “Algorithm and architectural level methodologies for low power”, section 11 in “*Low power design methodologies*”, editors J. Rabaey, M. Pedram, Kluwer Academic Publishers, 1996.

[50] Mehra R., Rabaey J.: “Exploiting regularity for low power design”, *Proceedings of the International Conference on Computer-Aided Design*, 1996

[51] Merkle, R.C.: “Reversible Electronic Logic Using Switches”, *Nanotechnology*, Volume 4, pp 21 - 40, 1993 (see also: http://nano.xerox.com/nanotech/electroTextOnly.html)

[52] Monteiro J, Devadas S., Ashar P., Mauskar A.: “Scheduling techniques to enable power management”, Proceedings of the 33rd Design Automation Conference, pp. 349-352, Las Vegas, Nevada, June 1996.

[53] “A Case for Intelligent DRAM: IRAM”, *Hot Chips 8 A Symposium on High-Performance Chips*, information can be browsed on: http://iram.cs.berkeley.edu/publications.html.

[54] Nieuwland A.K., Lippens P.E.R.: "A heterogeneous HW-SW architecture for hand-held multi-media terminals", *proceedings IEEE workshop on Signal Processing Systems*, SiPS'98, pp. 113-122.

[55] Payne R.E.: "Self-Timed FPGA Systems", *Proceedings of the 5th International Workshop on Field Programmable Logic and Applications*, LNCS 975, September 1995.

[56] Pedram M.: "Power minimization in IC design: principles and applications", *ACM Transactions on Design Automation*, Vol. 1, no. 1, pp. 3-56, Jan 1996.

[57] Piguet, C, et al.: "Low-power embedded microprocessor design", *Proceeding Euromicro-22*, pp. 600-605, September 1996.

[58] Rabaey J. et al.: "Low Power Design of Memory Intensive Functions Case Study: Vector Quantization", *IEEE VLSI Signal Processing Conference*, 1994.

[59] Rabaey J., Guerra L., Mehra R.: "Design guidance in the Power Dimension", *Proceedings of the ICASSP*, 1995.

[60] Rabiner W., Chandrakasan A.: "Network-Driven Motion Estimation for Wireless Video Terminals", *IEEE Transactions on Circuits and Systems for Video Technologies*, Vol. 7, No. 4, August 1997, pp. 644-653.

[61] Rabiner W., Chandrakasan A.: "Network-Driven Motion Estimation for Portable Video Terminals", Proc. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)*, April 1997, Vol. 4, pp. 2865-2868.

[62] Semiconductor Industry Association: "The national technology roadmap for semiconductors: Technology needs", *Sematech Inc*., http://www.sematech.org, Austin, USA, 1997.

[63] Sheng S., Chandrakasan A., Brodersen R.W.: "A Portable Multimedia Terminal", *IEEE Communications Magazine*, pp. 64-75, vol. 30, no. 12, Dec., 1992.

[64] Simunic T., Benini L., De Micheli G: "Cycle-accurate simulation of energy consumption in embedded systems", *Proceedings Design Automation Conference DAC 99*, 1999.

[65] Smit J., Bosma M.: "Graphics algorithms on Field Programmable Function Arrays", *proceedings of the 11th EuroGraphics workshop on graphics hardware*, Eds. B.O. Schneider and A. Schilling, pp.103-108, 1996.

[66] Gerard J.M. Smit, Martinus Bos, Paul J.M. Havinga, Sape J. Mullender, Jaap Smit: "Chameleon - reconfigurability in hand-held multimedia computers", *proceedings First International Symposium on Handheld and Ubiquitous Computing*, HUC'99, September 1999.

[67] Smit J., Stekelenburg M., Klaassen C.E., Mullender S., Smit G., Havinga P.J.M.: "Low cost & fast turnaround: reconfigurable graph-based execution units", *proceedings 7th BELSIGN workshop*, Enschede, the Netherlands, May 7-8, 1998

[68] Smit G.J.M., Havinga P.J.M., van Opzeeland M., Poortinga R.: "Implementation of a wireless ATM transceiver using reconfigurable logic", *proceedings wmATM'99*, June 2-4, 1999.

[69] Snyder J.H., et al.: "Low power software for low-power people", *Symposium on low power electronics*, October 1994.

[70] Srivastava M.: "Design and optimization of networked wireless information systems", *IEEE VLSI workshop*, April 1998.

[71] Srivastava M.: "Designing energy effuicient mobile systems", Tutorial during *MobiCom'99*, http://www.janet.ucla.edu/~mbs/tutorials/mobicom99, August 1999.

[72] Stan M.R., Burleson W.P.: “Bus-invert coding for low-power I/O”, *IEEE Trans. VLSI Syst.*, Vol. 3, no. 1, pp. 49-58, 1995.

[73] Stemm, M, et al.: “Reducing power consumption of network interfaces in hand-held devices”, *Proceedings mobile multimedia computing MoMuc-3*, Princeton, Sept 1996.

[74] Tierno J.A., Martin A.J.: “Low-energy asynchronous memory design”, http://www.cs.caltech.edu/~alains/publications/pub.html.

[75] Tiwari V. et al.: “Compilation Techniques for Low Energy: An Overview”, *IEEE Symposium on Low Power Electronics*, October 1994.

[76] Weiser, M, et al.: “Scheduling for reduced CPU energy”, *proceedings of the first USENIX Symposium on operating systems design and implementation*, pp. 13-23, November 1994.

[77] “Minimizing power consumption in FPGA designs”, *XCELL* 19, page 34, 1995.

[78] Podlaha, E.J., Cheh, H.Y.: “Modeling of cylindrical alkaline cells. VI: variable discharge conditions”, *Journal of Electrochemical Society*, vol 141, pp. 28-35, Jan. 1994.

[79] Porta La T.F., Sabnani K.K., Gitlin R.D.: “Challenges for nomadic computing: mobility management and wireless communications”, *Mobile networks and applications*, Vol. 1, No. 1, pp. 3-16, August 1996.

[80] Rambus Inc.: “Direct Rambus Memory for mobile PCs”, http://www.rambus.com.

[81] Rashid R.F.: “Personal Computing – the new future”, *keynote speech MobiCom’99*, August 1999.

[82] Sheng S., Chandrakasan A., Brodersen R.W.: “A Portable Multimedia Terminal”, *IEEE Communications Magazine*, pp. 64-75, vol. 30, no. 12, Dec., 1992.

[83] Simunic T., Benini L., De Michelle G.: “Energy-efficient design of battery-powered embedded systems”, *Proceedings ISLPED’99*, 1999.

[84] Truman T.E., Pering T., Doering R., Brodersen R.W.: The InfoPad multimedia terminal: a portable device for wireless information access”, *IEEE transactions on computers*, Vol. 47, No. 10, pp. 1073-1087, October 1998.

[85] Udani S., Smith J.: “The power broker: intelligent power management for mobile computers”, Tech. report MS-CIS-96-12, *Department of Computer Information Science*, University of Pennsylvania, May 1996.

[86] Udani S., Smith J.: “Power management in mobile computing”, Tech. report MS-CIS-98-26, *Department of Computer Information Science*, University of Pennsylvania, August 1996.

[87] Villasenor J., Mangione-Smith W.H.: “Configurable Computing”, *Scientific American*, June 1997.

[88] Wireless Application Protocol Forum Ltd.: “Official Wireless Application Protocol”, *Wiley Computer Publishing*, 1999, http://www.wapforum.org.

[89] Yeap G.K.: “Practical low power digital VLSI design”, *Kluwer Academic Publishers*, ISBN 0-7923-80.

[90] Yeung N. et al.: “The design of a 55 SPECint92 RISC processor under 2W”, *Proceedings of the international solid-state circuits conference ’94*, San Francisco, CA, pp. 206-207, February 1994.

[91] Zorzi, M., Rao, R.R.: “Error control and energy consumption in communications for nomadic computing”, *IEEE transactions on computers*, Vol. 46, pp. 279-289, March 1997.

3

The design of a system architecture for mobile multimedia computers

This chapter[1] *discusses the system architecture of a portable computer, called Mobile Digital Companion, which provides support for handling multimedia applications energy efficiently. Because battery life is limited and battery weight is an important factor for the size and the weight of the Mobile Digital Companion, energy management plays a crucial role in the architecture. As the Companion must remain usable in a variety of environments, it has to be flexible and adaptable to various operating conditions.*

The Mobile Digital Companion has an unconventional architecture that saves energy by using system decomposition at different levels of the architecture and exploits locality of reference with dedicated, optimised modules. The approach is based on dedicated functionality and the extensive use of energy reduction techniques at all levels of system design. The system has an architecture with a general-purpose processor accompanied by a set of heterogeneous autonomous programmable modules, each providing an energy efficient implementation of dedicated tasks. A reconfigurable internal communication network switch exploits locality of reference and eliminates wasteful data copies.

3.1 Introduction

One of the most compelling issues in mobile computing is to keep the energy consumption of the mobile low. This chapter discusses the system architecture of a portable computer, called *Mobile Digital Companion*, which provides support for handling multimedia applications energy efficiently. The *Mobile Digital Companion* was designed as part of the MOBY DICK project [62][50]. This project addresses fundamental

[1] Major parts of this chapter have been presented in two presentations at the first *Euromicro summer school on mobile computing '98*, August 1998 [30][62].

issues in the architecture, design and implementation of low-power hand-held computers, with particular emphasis on energy conservation.

3.1.1 Mobile systems today

The research community and the industry have expended considerable effort toward mobile computing and the design of portable computers and communication devices. Inexpensive gadgets that are small enough to fit in a pocket (like PDAs, palmtop computers and digital cameras) are joining the ranks of notebook computers, cellular phones and video games. Present day portable computers run most common interactive applications like word processors and spreadsheets without any noticeable computation delay. These devices now support a constantly expanding range of functions, and multiple devices are converging into a single unit [37]. Personal computers are becoming an integral part of daily life, as portable appliances such as wristwatches and cellular phones have become over the last few years. The emergence of wireless communication and the enormous improvements in technology that allows us to integrate many functions in one chip has opened up many possibilities for mobile computing. Communication, data processing and entertainment will be supported by the same platform, enhanced by the world-wide connectivity provided by the Internet.

We first take a brief look at the various mobile systems on the market today. Note that many of these systems have no built-in wireless networking capability, but rather rely on an external wireless modem for wireless connectivity. The wireless modem is in general based on a cellular phone, or on wireless LAN (WLAN) products. Wireless LANs are generally intended for short-range (several hundred meters) indoor use, as opposed to the outdoor several-kilometres range of cellular systems. Wireless LANs have a higher data rate than cellular phones, on the order of megabits per second, and the size is smaller, the power consumption is comparable.

Current mobile systems can be classified into the following categories based on their functions and form factors.

- *Laptops* – Laptops are not really mobile systems since they are too large and too heavy. In essence they are just battery operated small desktop machines. Wireless communication is generally based on WLAN products that can be plugged in as a PC-card.
- *Pen tablets* – Pen tablets can be viewed as laptops without keyboards. Interaction with the pen tablet is through pen input. In most cases, the pen replaces the mouse as pointer device. Some tablets have an internal radio modem, whereas others require an external radio modem. Generally spoken, these terminals are no different from the average desktop.
- *Virtual books* – Recently several products have been introduced that replace paper as the medium for reading and browsing a wide variety of material [13][59][65]. These systems have good quality displays, and a rather conventional architecture. User input is limited to a few buttons, and a pen.

- *Handheld Personal Computers (HPC)* – Systems of this category are basically miniature laptops. They are characterised by a reduced form factor keyboard and a half-VGA resolution display. They usually run reduced versions of Windows applications, including word processing, presentation, and scheduling software. The communication is usually a wire-line modem and an infrared port.
- *Personal Digital Assistants (PDA)* – the PDA is generally a monolithic device without a keyboard (although some have small sized keyboards) and fits in the user's hand. As such, pen input is the norm, and handwriting recognition is common. Communication abilities involve a docking port or serial port for connecting to and synchronising with a desktop computer, and possibly a modem.
- *Smart phones* – Although cellular phones may have several peripheral functions like a calculator, date book, or phone book, they are foremost a communication tool. Combination devices like the Nokia 9000 are essentially PC-like devices attached to a cellular phone.
- *Wireless terminal* – These systems are basically nothing more than the wireless extended input and output of a desktop machine which acts as the server. These systems are designed to take advantage of high-speed wireless networking to reduce the amount of computation required on the portable.

It will be clear that current mobile systems are primarily either data processing terminals or communication terminals. The trend in data processing terminals has been to shrink a general-purpose desktop PC into a package that can be conveniently carried. Even PDAs have not ventured far from the general-purpose model, neither architectural nor in terms of usage model.

3.1.2 The future: Mobile Digital Companion

Topic of this research is the architecture of a future handheld device, called *Mobile Digital Companion* (in this thesis also referred to as *Companion*). A *Mobile Digital Companion* will be a personal machine, and users are likely to become quite dependent on it.

The *Mobile Digital Companion* 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 [50]. It will resemble a PDA, that is, it looks like a normal PDA, but the functionality and typical use of the system are very different. Typical applications of a *Mobile Digital Companion* are diary, e-mail, web browsing, note-taking, walkman, video player and electronic payments. The *Mobile Digital 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 (wireless) network with variable connectivity. Our primary objective in designing the architecture has been to support a wide variety of applications for mobile devices that make efficiently use of the available resources. Such companions must meet several major requirements: high performance, energy efficient, a notion of Quality of Service (QoS), small size, and low design complexity.

The *Mobile Digital Companion* is more than just a small machine to be used by one person at a time like the traditional organisers and desktop assistants. We distinguish two types of systems: 'desktop companions' and '*Mobile Digital Companion*s'. A desktop companion is a handheld machine that is designed to give roaming users access to their business data and applications while on the road. Desktop companions are designed and optimised for compatibility and communication with the user's desktop machine(s), e.g. via modem, infrared or a docking station. A typical example of a desktop companion is a PDA or (sub)notebook running Windows CE [26].

The *Mobile Digital Companion* extends the notion of a desktop companion in several ways.

- It will run applications typically found in desktop companions, but it will also run other applications using *external public services*. A *Mobile Digital 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'.
- *Multimedia computing* will also be an essential part of the *Mobile Digital Companion*. If a mobile computer has to be used for every day work, then multimedia devices, such as audio and video have to be included in the system. Nowadays, there are several portable multimedia devices available (digital cameras, MP3man, etc.), but all these systems are no more than dedicated devices. What lacks is a good integration between all these devices.
- All current desktop companions have communication facilities to communicate with the desktop master. However, as the dependence on network-accessible information storage and computation increases, the desire to ubiquitously access the network requires a much more *sophisticated wireless networking* capability. The network access should support heterogeneity in many dimensions (transport media, protocols, data-types, etc.).

The most important factors, which will determine the success of the *Mobile Digital Companion*, are the utility and convenience of the system. An important feature will be the interface and interaction with the user: voice and image input and output (speech and pattern recognition) will be key functions. The use of real-time multimedia data types like video, speech, animation and music greatly improve the usability, quality, productivity, and enjoyment of these systems. Multimedia applications require the transport of multiple synchronised media streams. Some of these streams (typically video streams) have high bandwidth and stringent real-time requirements. These applications also include a significant amount of user interaction. Most of the applications we consider require not only a certain Quality of Service for the communication (like high bandwidth and low latency), but also a significant amount of computing power. The compute requirements stem from operations such as compression/decompression, data encryption, image and speech processing, and computer graphics.

The Mobile Digital Companion is thus quite a versatile device. Nevertheless these functions have to be provided by relatively small amount of hardware because a main

requirement for the Companion is small size and weight. As most current battery research does not predict a substantial change in the available energy in a battery, energy efficiency plays a crucial role in the architecture of the *Mobile Digital Companion*. An integrated solution that reduces chip count is highly desirable.

3.1.3 Approach

The approach to achieve a system as described above is to have autonomous, reconfigurable modules such as network, video and audio devices, interconnected by a switch rather than by a bus, and to offload as much as work as possible from the CPU to programmable modules that are placed in the data streams. Thus, communication between components is not broadcast over a bus but delivered exactly where it is needed, work is carried out where the data passes through, bypassing the memory. Modules are autonomously entering an energy-conservation mode and adapt themselves to the current state of resources, the environment and the requirements of the user. The amount of buffering is minimised, and if it is required at all, it is placed right on the data path, where it is needed. To support this, the operating system must become a small, distributed system with co-operating processes occupying programmable components – like CPU, DSP, and programmable logic – among which the CPU is merely the most flexibly programmable one.

The interconnect of the architecture is based on a switch, called *Octopus*, which interconnects a general-purpose processor, (multimedia) devices, and a wireless network interface. The *Octopus* switch is subject of Chapter 4. Although not uniquely aimed at the desk-area, our work is related to projects like described in [4][20] and [32] in which the traditional workstation bus is replaced by a high speed network in order to eliminate the communication bottleneck that exists in current systems.

3.1.4 Outline

We first indicate in Section 3.2 the main challenges in mobile system design which will provide the motives why there is a need to revise the system architecture of a portable computer. Section 3.3 then describes the philosophy behind the architecture of the *Mobile Digital Companion*, and introduces the various basic mechanisms used: the connection-centric approach, the timing control, the Quality of Service framework, and finally presents the basic system architecture. Then we will give an overview of the state of the art in mobile multimedia computing in Section 3.4. Finally, we present the summary and conclusions in Section 3.5.

3.2 Design issues of mobile systems

There is a new and growing class of users whose primary computing needs are to access the information infrastructure, computing resources, and real-time interactive systems as well as direct communications with other people. These applications, which are

communication oriented rather than computation oriented, is the motivation for a re-examination of the requirements of the system architecture and the hardware that is needed. These applications require a personal mobile digital companion that primarily has support for high bandwidth real-time communication and multimedia capabilities [70]. High-performance general-purpose computing is not a prominent requirement.

Recent improvements in circuit technology and software development have enabled the use of real-time data types like video, speech, animation, and music. As mobile computers evolve, support for multimedia-rich applications will become the standard. It is expected that by 2000 90 percent of the desktop cycles will be spent on multimedia applications [16].

The computer industry has made enormous progress in the development of mobile computing and the design of portable computers. This is partly due to recent advances in technology. These systems are generally based on architectures of high performance personal computers that have some provisions for wireless computing and a rudimentary form of power management. In this section we will show that such an approach is not sufficient if we want to be able to have a hand-held machine with multimedia capabilities that can be used conveniently in a wireless environment.

3.2.1 Mobility

The emergence of novel multimedia applications and services that leverage the growth in mobile computing depends on the availability of a flexible broadband wireless infrastructure. Key technical issues of this infrastructure include Quality-of-Service control and application software integration. Mobile systems will have a set of challenges arising from the diverse data types with different quality-of-service (QoS) requirements they will handle, their limited battery resources, their need to operate in environments that may be unpredictable, insecure, and changing, and their mobility resulting in changing set of available services.

The following are the key technological challenges that we believe will need to be addressed before mobile systems like the Mobile Digital Companion will become real.

- *Energy efficiency* – As the current portable computers have shown to be capable of assisting mobile users in their daily work, it is becoming increasingly evident that merely increasing the processing power and raising raw network bandwidth does not translate to better devices. Weight and battery life have become more important than pure processing speed. Energy consumption is becoming the limiting factor in the amount of functionality that can be placed in portable computers like PDAs and laptops.
- *Infrastructure* – The design of mobile systems cannot be done in isolation. The mobile system of the future is likely to be designed to operate autonomously, but it is also very likely that it relies on an external infrastructure to access information of any kind. The mobile will likely encounter many, very diverse environments and various network infrastructures. Furthermore, mobiles may vary along many axes, including screen size, colour depth, processing power, and available functions. Servers (or proxy agents that are placed between mobiles and servers) can perform

computation and storage on behalf of clients. Partitioning of functions between the wireless system and servers residing on the network is an important architectural decision that dictates where applications can run, where data can be stored, the complexity of the terminal, and the cost of communication services.

- *Adaptability* – Wireless mobile systems face many different types of variability in their environment in both the short and the long term. Mobile systems will need the ability to adapt to these changing conditions, and will require adaptive radios, protocols, codecs and so on. Adaptive error control and adaptive compression are examples of such techniques.
- *Reconfigurability* – To combat a higher degree of variations in operational environment than is possible with adaptable systems, reconfigurable architectures can be used that allow new software and hardware functions to be downloaded. Thus rather than changing parameters of algorithms to current conditions, an entirely new set of protocols and algorithms can be used. An alternative approach to adapt to a change in environment would be to have a mobile system with all possible scenarios built-in. Such multimode systems become costly, and relatively inflexible.
- *Security* – When computers become more involved in people's personal and business activities security i.e. confidentiality, privacy, 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 or losing keys or confidential data.
- *User interfaces* – Traditional keyboards and display based interfaces are not adequate for the mobile systems of the future because of the required small size and weight of these system. Instead, intrinsically simpler interfaces based on speech, touch, pen and so forth are more likely to be used and more adequate to the small form factors of these systems. Because these systems will be consumer appliances that are used by non-experts, the complex environment should remain hidden from the user, or presented at a level that can easily be understood by the user.

In the remainder we shall focus on the issues that are related to energy consumption, i.e. energy-efficiency and adaptability.

3.2.2 Multimedia

The systems that are needed for multimedia applications in a mobile environment must meet different requirements than current workstations in a desktop environment can offer. The basic characteristics that multimedia systems and applications needs to support are [17]:

- *Continuous-media data types* – Media functions typically involve processing a continuous stream of data, which implies that temporal locality in data memory accesses no longer holds. Remarkably, data caches may well be an obstacle to high

performance and energy efficiency for continuous-media data types because the processor will incur continuous cache-misses.

- *Provide Quality of Service (QoS)* – Instead of providing maximal performance, systems must provide a QoS that is sufficient for qualitative perception in applications like video.
- *Fine-grained parallelism* – Typical multimedia functions like image, voice and signal processing require a fine-grained parallelism in that the same operations across sequences of data are performed. The basic operations are relatively small.
- *Coarse-grained parallelism* – In many applications a pipeline of functions process a single stream of data to produce the end result.
- *High instruction reference locality* – The operations on the data demonstrate typically high temporal and spatial locality for instructions.
- *High memory bandwidth* – Many multimedia applications require huge memory bandwidth for large data sets that have limited locality.
- *High network bandwidth* – Streaming data – like video and images from external sources – requires high network and I/O bandwidth.

Distributed multimedia applications running in a mobile environment have a number of special characteristics. Many future wireless mobile systems will operate in various, relatively unregulated environments such as home and workplace LANs with time varying interference levels. Existing cellular telecommunication networks can also be used to provide wireless access to wired computer networks. Applications cannot rely on the wireless network to provide high throughput or fast response times. Two of the most fundamental characteristics are:

- A *heterogeneous processing environment* (including relatively low-power mobile hosts) and,
- *rapid and massive fluctuations* in the quality of service provided by the underlying communication infrastructure.

QoS control is a key feature for efficient utilisation of resources in wireless networks supporting mobile multimedia. Traditional static resource-allocation models lack flexibility, and thus cope poorly with multimedia interactivity and session mobility.

The challenge is to maintain a high perceived end-to-end quality without limiting applications to the point where they are no longer useful. Multimedia networking requires at least a certain minimum bandwidth allocation for satisfactory application performance [58]. The minimum bandwidth requirement has a wide dynamic range depending on the users' quality expectations, application usage models, and applications' tolerance to degradation. In addition, some applications can gracefully adapt to sporadic network degradation while still providing acceptable performance. For example, while video-on-demand applications may in general tolerate bit rate regulations within a small dynamic range, applications such as teleconferencing may have a larger dynamic range for bit rate control. Other multimedia applications may allow a larger range of bit rate control by resolution scaling.

3.2.3 Limitation of energy resources

Although current portable computers have shown to be capable of assisting the mobile user in their daily work, it is becoming increasingly evident that merely increasing the processing power and raising raw network bandwidth does not translate to better devices. Weight and battery life has become more important than pure processing speed. These two factors are related by battery size: to operate a computer for a longer time without recharging, we need a larger, heavier, battery. The limitation is therefore the total amount of electric energy stored in that battery that is available for operation. Battery technology has improved at a glacial pace compared to the pace at which the amount of processing power in mobile systems is increasing while their size is decreasing. Energy consumption is becoming the limiting factor in the amount of functionality that can be placed in portable computers like PDAs and laptops.

To extend battery life, we have to design the system to be more efficient in the way it uses this energy. However, even today, research is still focussed on performance and circuit design. Due to fundamental physical limitations, progress towards further energy reduction will have to be found beyond the chip-level. Key to energy efficiency in future mobile systems will be the higher levels: energy-efficient architectures and protocols, energy aware applications, etc.

The vast majority of energy-critical electronic products are far more complex than a single chip. In most electronic products, the digital components consume a fraction of the energy consumed. Analog, electro-mechanical and optical components are often responsible for a large contributions to the power budget of a portable computer [42]. One of the most successful techniques employed by designers at the system level is *dynamic power management*, in which parts of the system have different energy modes, and can even be completely powered down.

3.2.4 System architectural problems

Within the traditional design of a mobile, a number of problem areas in hardware and software architectures can be identified concerning the energy consumption [8]. We will just mention a few.

- A key issue is the lack of interaction between hardware facilities for energy management (power saving modes, device interrupts that ‘wake up’ the CPU, etc.) and the operating system and application software. In particular, the device drivers, the operating system and the applications attempt to autonomously control the hardware. This mis-coordination causes inexplicable erratic behaviour. Examples of this problem are unexpected screen blanks during a presentation, or when the disk spins up and spins down unannounced (causing annoying delays).
- Second, opportunities for saving energy are not exploited because devices are controlled at a too low level, ignoring high-level information on what the user actually needs during system operation.
- Third, applications assume that the computer is always on. This assumption often causes excessive energy consumption. For example, polling cycles, when an

application is waiting for a response, are very inefficient from an energy point of view.

- Finally, the current operating system software and networking software emphasises flexibility and performance, and is constructed from components developed by independent groups. Within system design a key role lies in the development of interfaces. A good working practise is to define interfaces in a hierarchical way, since the complexity of the system is reduced to manageable proportions. Such an approach is also directed to by standardisation approaches like the ISO/OSI network layer structure. However, the result of this flexibility and this development approach is that in many cases numerous unnecessary data copies occur between different modules. Non-copying protocol stacks do exist [67], but are not widely used. Operations such as data copying, servicing of interrupts, context switches, software compression, are in current systems often responsible for poor performance and high energy consumption. Not well designed network protocols that do not efficiently make use of one of the most energy demanding devices of the mobile, the wireless interface, waste also a lot of energy.

Our vision is that there is a vital relationship between hardware architecture, operating system architecture, applications' architecture and human-interface architecture, where each benefits from the others: the applications can adapt to the power situation if they have an appropriate operating system API for doing so; the operating system can minimise the energy consumption by keeping as many as components turned off as possible; the hardware architecture can be designed to route data paths in such a way that, for specific functions, only a minimum of components need to be active.

3.2.5 System level integration

The design flow of a system consists of various levels of abstraction. By carefully designing all components that make up the mobile system (i.e. the hardware components, the architecture, the operating system, the protocols, and the applications) in a coherent and integrated fashion, it is possible to minimise the overhead resulting from the use of these operations and reduce the energy consumption. Any single application, device driver, or hardware module does not have sufficient knowledge of the status of the entire system to effectively make autonomous decisions concerning energy management.

An important aspect of the design flow is the relation and feedback between the levels. Given a design specification, a designer is faced with several different choices on different levels of abstraction. The designer has to select a particular algorithm, design or use an architecture that can be used for it, and determines various parameters such as supply voltage and clock frequency. This *multi-dimensional design space* offers a large range of possible trade-offs. The most effective design decisions derive from choosing and optimising architectures and algorithms at the highest levels. It has been demonstrated by several researchers [14] that system and architecture level design decisions can have dramatic impact on power consumption. However, when designing a system it is a problem to predict the consequences and effectiveness of design decisions

because implementation details can only be accurately modelled or estimated at the technological level and not at the higher levels of abstraction.

The ability to integrate diverse functions of a system on the same chip provides the challenge and opportunity to do system architecture design and optimisations across diverse system layers and functions. Especially a mobile computing device that combines multimedia computing and communication functions exemplifies the need for system level integration. Functions ranging from audio and video processing, radio modem, wireless interface, security mechanisms, and user interface oriented applications have to be integrated in a small portable device with a limited amount of energy. Information generated by a device or an application has to traverse and be processed at all these layers, providing the system architect with a rich design space of trade-offs.

3.2.6 Programmability and adaptability

As mobile computers must remain usable in a variety of environments, they have to support different encoding and encryption schemes and protocols to conform to different network standards, and to adapt to various operating conditions. A mobile computer will therefore require a large amount of circuits that can be customised for specific applications to stay versatile and competitive. *Programmability* and *adaptability* is thus an important requirement for mobile systems, since the mobiles must be flexible enough to accommodate a variety of multimedia services and communication capabilities and adapt to various operating conditions in an (energy) efficient way.

The requirement for programmability in systems on a chip is also triggered by economical reasons [72]. A well designed ASIC will solve the specific problem for which it was designed, but probably not a slightly modified problem introduced after the design was finished. Furthermore, even if a modified ASIC can be developed for the new problem, the original hardware circuits may be too highly customised to be reused in successive generations. Moreover, the high costs involved in designing and implementing a chip does not justify the design of a system that implements only a single application. Furthermore, because the requirements of applications are increasing rapidly and new standards are emerging quite fast, new chips need to be designed very often.

3.2.7 Discussion

Basically, there are two types of computer devices for use on the road: the palm-top computer and the notebook computer. Palm tops are mainly used for note-taking, electronic appointment books, and address books. Notebook computers are battery powered personal computers, and the current architectures for mobile computers are strongly related to the architecture of high-performance workstations. Both the notebook and the personal computer generally use the same standard PC operating system such as Windows 98 or Unix, same applications, use the same communication protocols and use the same hardware architecture. The only difference is that portable computers are smaller, have a battery, a wireless interface, and sometimes use low-power components. The problems that are inherent to mobile computing are either neglected (e.g. the

communication protocols are still based on TCP/IP, even though these behave poor in a wireless environment [6]), or tried to solve with brute force neglecting the increase in energy consumption (e.g. extensive error control, or software decompression). Adaptability and programmability should be major requirements in the design of the architecture of a mobile computer.

In the near future other small electronic gadgets like Web-phones, MP3 man, games and digital cameras will be integrated with the portable computers in a *personal mobile computing* environment. This not only leads to greater demand for computing power, but at the same time the size, weight and energy consumption must be small.

We are entering an era in which each microchip will have billions of transistors. One way to use this opportunity would be to continue advancing our chip architectures and technologies as just more of the same: building microprocessors that are simply complicated versions of the kind built today [5]. However, simply shrinking the data processing terminal and radio modem, attaching them via a bus, and packaging them together does not alleviate the architectural bottlenecks. The real design challenge is to engineer an integrated mobile system where data processing and communication share equal importance and are designed with each other in mind. Connecting current PC or PDA designs with an off-the-shelf communication subsystem, is not the solution. One of the main drawbacks of merely packaging the two is that the energy-inefficient general-purpose CPU, with its heavyweight operating system and shared bus, becomes not only the center of control, but also the center of data flow in the system [41].

Clearly, there is a need to revise the system architecture of a portable computer if we want to have a machine that can be used conveniently in a wireless environment. A system level integration of the mobile's architecture, operating system, and applications is required. The system should provide a solution with a proper balance between flexibility and efficiency by the use of a hybrid mix of general-purpose and the application-specific approaches.

3.3 The system architecture of a Mobile Digital Companion

In this section we describe the architecture of the *Mobile Digital Companion*. The properties to be achieved by the architecture are:

1. the *flexibility* to handle a variety of (multimedia) services and standards and
2. the *adaptability* to accommodate to its current environment for the changing conditions in communication connectivity, required level of security, and available resources.
3. Configuration parameters can be adapted according to the *QoS requirements*. The components of architecture should be able to adapt their behaviour to the current

environment and requirements to handle the required tasks efficiently. In doing this, the system should be fully aware of its *energy consumption.*

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 generalised 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. Using system decomposition at different levels of the architecture and exploiting locality of reference with dedicated, optimised modules much energy can be saved.

3.3.1 Approach

The scope of this section is the architecture of systems hardware, firmware and software in general, and the following issues in particular:

- Eliminate as much as possible the CPU as an active component in all data streams. In particular we aim to eliminate the active participation of the CPU in media transfers between components such as network, display and audio system (e.g. when the companion functions as a phone, walk-man, TV, or electronic newspaper). Unlike a local CPU architecture, in which I/O peripherals enhance the functionality of the core processor, our goal was to design intelligent peripherals that are capable of processing I/O events and can manage data streams without relying on a centralised processor.
- Eliminate as much as possible memory as the intermediate station for all data transfers between devices. The energy required to transfer and store the data is wasted if the data only occupies memory in transit between two devices (e.g. network and screen or network and audio).
- Use dynamic programmable and adaptable devices that convert incoming or outgoing data streams, in particular network, security, display and audio devices. Because they are programmable, they can handle different data encoding standards and communication protocols autonomously. This has two effects. First, devices can be designed to communicate directly with each other, instead of requiring CPU intervention for adapting data streams.
    - A display device will convert between, for example, MJPEG-compressed data and pixel data. Multimedia applications can benefit from compression as a means of saving (energy wasting) network bandwidth, but require a low power platform for the necessary calculation.
    - A network device will convert between byte streams used internally and, for example, TCP/IP packet streams. Network protocol stacks can be installed on the network interface device, or even on the base station, where they can handle much of the communication functions while the CPU is turned off.
    - Security protocols can be run in an environment beyond the direct control of

the operating system or applications. Regular software is prone to many forms of attack (viruses, Trojan horses, bugs).

The second effect is that, for a large number of these data-conversion functions (or filter functions), digital signal processors (DSPs), field-programmable hardware, or dedicated hardware are both faster and more energy efficient than general-purpose CPUs.

To limit the communication overhead and the required buffering, the granularity of the tasks on the devices is rather coarse, and the application is partitioned in large blocks. The programmability of each device (or *module*) is more fine-grained and is controlled by the individual autonomous module. The module application can be partitioned over various computational resources, based on the granularity of their application. The proposed architecture of the *Mobile Digital Companion* is shown in Figure 1. The figure shows a typical system with Processor module, Network module, Display module, Camera module, and Audio module, all interconnected by a switching fabric (the *Octopus* switch).

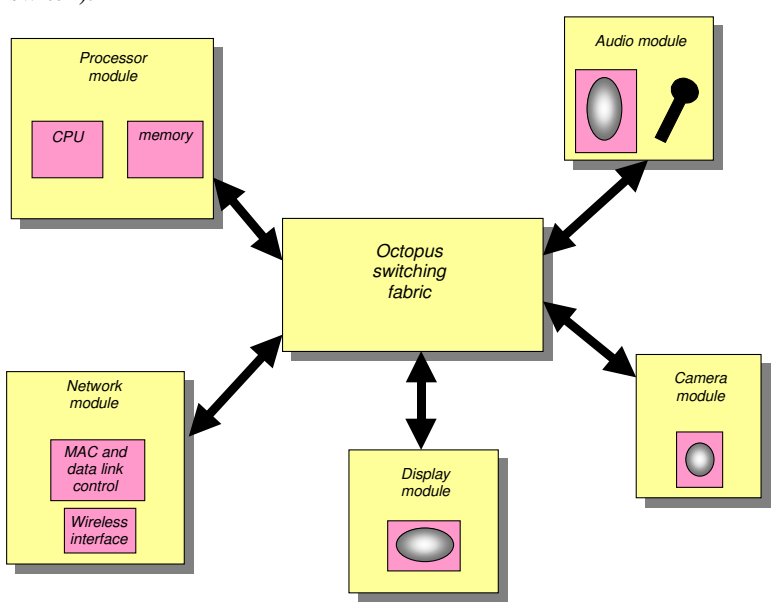


Figure 1: A typical Mobile Digital Companion architecture.

The system has a number of premises:

- An architecture with a general-purpose processor accompanied by a set of heterogeneous programmable modules, provides an energy efficient implementation of dedicated tasks.
- Communication between modules is based on *connections*. Connections are associated with a certain QoS. This identifier provides the mechanism to support lightweight protocols that provide data-specific transport services.
- A reconfigurable internal communication network exploits locality of reference and eliminates wasteful data copies. The data paths through the switch only consume energy when data is being transferred, leaving most of the switch turned off nearly all the time.
- 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 using intelligent network interfaces that can deal efficiently with a mobile environment, by using a energy aware network protocol, and by using an energy efficient MAC protocol that minimises the energy consumption of network interfaces [31].
- A Quality of Service framework for integrated management of the resources of the *Mobile Digital Companion* in which each module has its own – dedicated – local power management. The operating system will control the power states of devices in the system and share this information with applications and users.

The Mobile Digital Companion is quite a versatile device. Nevertheless these functions can be provided by relatively little hardware. All modules are programmable, but with the exception of the processor module, not as easily or flexibly programmable as conventional CPUs. The components of the modules in the prototype encompass (micro)processors, DSPs and programmable logic (Field Programmable Gate Arrays (FPGA), or Field Programmable Function Arrays (FPFA) [64]). Ultimately, all these components should be integrated into one large VLSI chip (a system-on-a-chip).

3.3.2 Philosophy

Our approach is based on dedicated functionality and the extensive use of energy reduction techniques at all levels of system design. We will use these techniques throughout the design of the *Mobile Digital Companion*, including technological level, architecture level, and system level. To preserve the locality inherent in the application or algorithm a hierarchical-granularity architecture is used that matches the computational granularity to the required operations.

The two main themes that can be used for energy reduction at system level are to *avoid waste*, and to *exploit locality of reference*.

Avoiding waste seems obvious, but in the design of a system it is difficult to avoid waste at various levels in the system. The reason for this is not only the carelessness of the designer, but is also due to the complexity of systems and the relations between the

various levels in the system. What is needed is a proper model in which the consequences of a design decision for other parts in the system can be predicted.

The component that contributes significantly to the total energy consumption of a system is the interconnect. Experiments have demonstrated that in chip-designs, about 10 to 40% of the total power may be dissipated in buses, multiplexers and drivers [47]. This amount can increase dramatically for systems with multiple chips due to large off-chip bus capacitance.

The amount of energy required for the transport of data can be reduced by using special memory interfaces (e.g. the Rambus memory technology [56]), or by using on-chip (cache) memory. However, as mentioned before, data caches in processors for multimedia applications are of little use, and may well become an obstacle to high performance and low power, because these media functions typically involve processing a continuous stream of input [37], thereby effectively emptying the cache of useful processor data. The temporal locality property in data memory access does not hold for such data traffic.

As already described in Chapter 2 there are two properties of algorithms important for reducing interconnect power consumption: locality and regularity.

- *Locality* relates to the degree to which a system or algorithm has natural isolated clusters of operation or storage with a few interconnections between them. Partitioning the system or algorithm into spatially local clusters ensures that the majority of the data transfers take place within the clusters and relatively few between clusters. Localisation reduces the communication overhead in processors and allows the use of reduced sized transistors, which results in a reduction of capacitance. The result is that the local buses are shorter and more frequently used than the longer highly capacitive global buses.
- *Regularity* in an algorithm refers to the repeated occurrence of computational patterns. Common patterns enable the design of less complex architecture and therefore simpler interconnect structure (buses, multiplexers, buffers) and less control hardware.

Most of the techniques for reducing energy consumption can be applied on general-purpose computing. However, for multimedia applications in particular, there is a substantial reduction in energy consumption possible as the computational complexity is high and they have a regular and spatially local computation. Also, 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 *Mobile Digital Companion*.

Locality of reference is exploited at several levels. The main philosophy used is that operations on data should be done at the place where it is most energy efficient and where it minimises the required communication. This can be achieved by matching computational and architectural granularity. In the system we have a *hierarchical granularity* in which we differentiate three main grain-sizes of operations:

- *fine grained* operations in the modules that perform functions like multiply and addition,
- *medium grained* operations are the functions of the modules. These functions are dedicated to the basic functionality of the module, e.g. the display module decompresses the video-stream.
- *course grained* operations are those tasks that are not specific for a module and that can be performed by the CPU module, or even on a remote compute server.

We use a *micro-distribution* to migrate tasks between functional modules within the *Mobile Digital Companion*, and a *macro-distribution* to migrate tasks between modules in the external world and on the portable system. In the latter approach certain functions are migrated from the portable system to a remote server that has plenty of energy resources. The remote server handles those functions that cannot be handled efficiently on the portable machine. A typical example of a macro-distribution can be utilised for the network protocol handling. Several researchers have showed that some network protocols perform badly over wireless channels [6]. A solution can be to split the connection in a separate wireless connection between the mobile and a (base)station on the fixed network, and a different connection over the wired network. The base-station can than perform part of the network protocol stack in lieu of the mobile, and use a dedicated and energy efficient protocol over the wireless channel. In such a system it is also simpler and efficient to adapt the protocols for the specific environment it is used in. For example, the network module of the *Mobile Digital Companion* is capable of adapting its error control, its flow control, and its scheduling policy to the current environment and requirements of the system and applications [31].

These principles have lead to our architecture that is capable of handling media-streams efficiently.

3.3.3 Memory-centric versus connection-centric

The system architecture of the *Mobile Digital Companion* is connection (or media) centric, which means that the media type of the traffic drives the data flow in the system using connections. For example, the video traffic from the network interface is transferred directly to the display, without interference from the CPU. This is in contrast to the memory-centric (or CPU-centric) architecture that is centered around a general-purpose processor that controls the media streams in a computer using a memory-addressing.

Memory-centric

Modern high-performance network protocols require that all network access is handled by the operating system, which adds significant overhead to both the transmission path (typically a system call and data copy) and the receive path (typically an interrupt, a system call, and a data copy). The communication costs can be broken up in per packet and per-byte costs. The per-packet cost can be optimised, and for large packets, this overhead is amortised over a lot of data. However, the cost of per-byte operations such

as data copying and checksumming is not reduced by increasing the packet size. Let us first take a look at the typical processing path that an information bit incurs in a multimedia networked computer. Typically, the information bits are generated at a module via a sensor such as a camera. Processing like coding and compression may be done by a codec at this stage. The control flow (and possibly also the information bits) passes through middleware / operating system layers. The bits will then be processed by an application for the transmission over the network. The bits are then sent by the application, again via middleware / operating system, to a network protocol stack which is composed of transport, network, link and medium access (MAC), and physical layer protocols. Typical functions in the protocol stack include routing, congestion control, error control, resource reservation, scheduling, etc. The bits are eventually sent over the network to other nodes where they traverse a reverse path. Notice how, instead of arithmetic functions like additions and multiplications, the primary importance in the system is processing for the protocols.

To address this performance problem, several *user-level communication architectures* have been developed that remove the operating system from the critical communication path [10] and to minimise the number of times the data is actually touched by the host CPU on its path through the system. The ideal scenario is a single-copy architecture in which the data is copied exactly once. Measurements of a single-copy protocol stack at Carnegie Mellon University show that for large reads and writes the single-copy path through the stack is five to seven times more efficient than the traditional implementation for large writes [67].

There are several ways to build flexible, high performance communication protocols. Advanced protocol design techniques include *application-level framing*, in which the protocol buffering is fully integrated with application-specific processing, and *integrated-layer processing*, in which many protocol layers are collapsed into highly efficient, monolithic code paths. Integrating buffer management between the application and network interface is important in eliminating data copies and reducing allocations and de-allocations. This integration, however, gives rise to additional complexity due to virtual addressing mechanisms and protection [22] and may need a substantial amount of memory [67].

The *traditional architecture* of a mobile, shown in Figure 2, is centered around a general-purpose processor with local memory and a bus that connects peripherals to the CPU. The long arrow in the figure indicates the essential data stream through the system when data arrives from the network, is transferred through the receive buffers on the network interface, copied to the ‘main’ memory, and then processed by the application. After the data is processed by the application, the data will traverse via ‘main’ memory, over the bus, to the output device (i.e. in the figure the display module). In general additional bus transfers between CPU and memory are introduced while traversing several protocol layers (e.g. for data conversion of the packets like Ethernet to IP, and subsequently IP to TCP). A large fraction of system time and power budget is thus devoted to bus transactions.

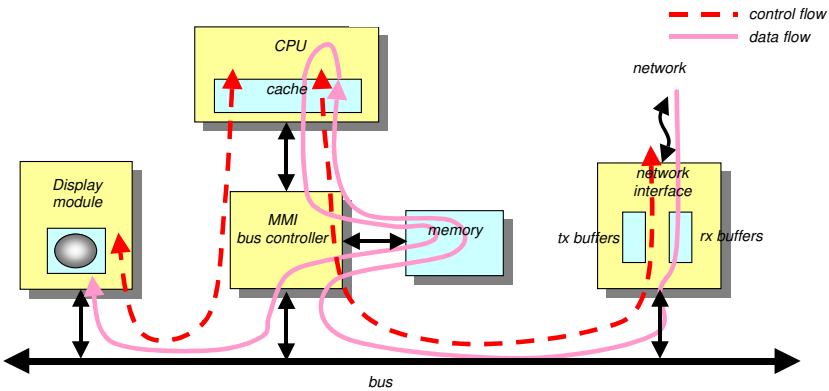


Figure 2: Data flow through a traditional architecture.

As can be seen in the figure, the CPU is required not only to handle the control path, but also to transfer the data between the devices. A better approach is to transfer vast amounts of data through the system by using Direct Memory Access (DMA) functionality of the interfaces and modules. DMA is used for the data transfer between the main memory and the buffers on the network interface. Figure 3 shows the separate data and control flows of such an optimised architecture. The CPU is now only required to perform the control flow between the devices, e.g. – like in our previous example – to the network interface and to the display module. Although this already reduces the demands laid on the processor drastically, the processor still needs to be active during the data transaction to perform the control flow.

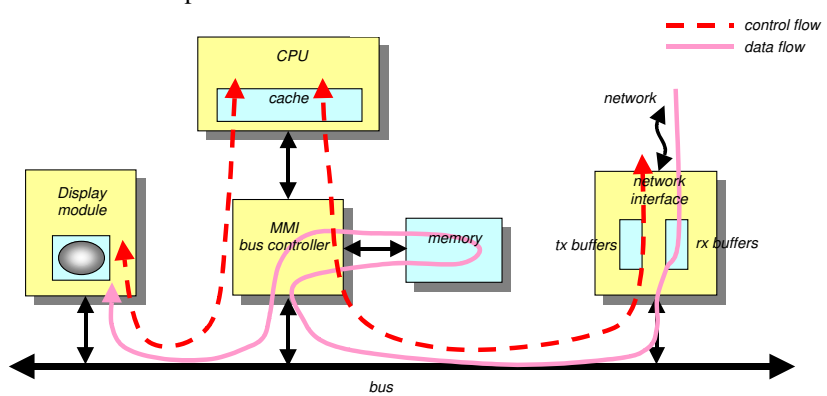


Figure 3: Separated data and control flow traditional architecture.

DMA can introduce some drawbacks. Checksumming needs to be done while copying the data, i.e. checksumming needs to be done in hardware. On workstations, the use of DMA to transfer data between network buffers and main memory is made more complicated by the presence of a cache and virtual memory. DMA can create

inconsistencies between the cache and main memory. Hosts can avoid this problem by flushing the data to memory before transferring it using DMA on transmit and invalidating the data in the cache before DMAing on receive. User pages have to be wired in memory to insure that they are not paged out while the DMA is in progress. Because of these extra overheads, it might sometimes be more efficient to use the CPU to copy packets between user and system space. Furthermore, it can be desirable to alter the data stream online, such as decompressing an MPEG audio or video stream. In such cases the CPU generally needs to perform the conversion.

In some special cases it might be possible to forward the data directly from source to destination. This can for example be applied to display graphics data on the screen if the display memory is accessible by the source. In this case the data only needs to travel once across the interconnect between source and sink.

Note that even in this optimised case, there are still one or two data copies required over the shared bus. Busses are significant sources of power dissipation due to high switching activities and large capacitance. Modern systems are typically characterised by wide and high-speed busses, which means that the capacitance and frequency factor of the power dissipation dominates.

The conventional memory-centred shared-bus architecture requires frequent traversal of multimedia streams over the highly capacitive central bus and through the layers of the operating system software for the simplest operations such as multiplexing/demultiplexing and interstream synchronisation. Indeed, measurements with a prototype wireless multimedia terminal at the University of California at Los Angeles (UCLA) with an embedded PC-based architecture show that large amounts of time and power go into memory and I/O transactions across the shared bus [41].

The same trend can be observed in microprocessor designs. *Computer* produced a special issue on “Billion-transistor architectures” that discussed problems and trends that will affect future processor designs, and several proposed microprocessor architectures and implementations [12]. Most of these designs focus on the desktop and server domain. The majority use 50 to 90 percent of their transistor budget on caches, which help mitigate the high latency and low bandwidth of external memory. In other words, the conventional vision of future computers spends most of the billion-transistor budget on redundant, local copies of data normally found elsewhere in the system [37].

Current systems based on a shared bus architecture are able to deliver the required performance for various multimedia applications not only by using the rapid advance in technology, but also by careful design and use of the interface modules. The process to achieve this requires a huge amount of effort of both the hardware designer of the I/O interfaces and the system designer. The hardware designer has to very carefully wire the devices to fit the needs of the application, tailor the circuits so that the wires are as short as possible and all the signals get from their originating point to the right place at exactly the right time. Then, the software designer must carefully determine in detail what the devices are capable of before designing a multimedia system [11]. There are many subtle device issues that can influence the overall I/O performance of a system. When, after a lot of fine-tuning, finally the system is running satisfactory, performance problems can

arise easily when the (hardware or software) configuration of the system is (slightly) changed, the operating system is updated, or the user is using a new application. The reason for these problems are often caused by the interconnect and the interconnection protocols. Since a shared bus cannot give QoS guarantees, a single device or application can reduce the throughput that is available for all devices.

Connection-centric

By designing an architecture that moves processing power closer to the data stream, it is possible to bypass the CPU altogether. This approach is especially well suited for continuous media data (e.g. audio, video, etc.), where the processing is actually of a very specialised nature (e.g. signal processing, compression, encryption, etc.) and needs to be carried out in real-time. The CPU is thus moved out of the data flow datapath, although it still participates in the control flow. The role of the CPU is reduced to a controller that initialises the system and handles complex protocol processing that are most easily implemented in software on a general-purpose processor.

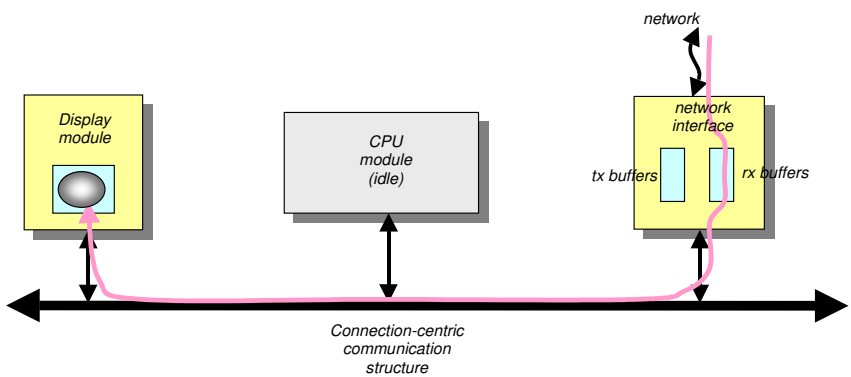


Figure 4: Data flow in a connection-centric architecture.

In contrast to memory-centric systems, a connection-centric system is decomposed out of application-specific coprocessors that communicate using connections. The operating system plays a crucial role in this architecture, as it is responsible to set-up the connections between the modules. The CPU and the operating system do not participate in the control flow during a transaction. The interconnection structure is not based on a bus that uses addresses, but is based on a connection-oriented communication structure (such as the *Octopus* switching fabric as depicted in Figure 1). In such a system the data traffic is reduced, mainly because unnecessary data copies are removed. For example, in a system where a stream of video data is to be displayed on a screen, the data can be copied directly to the display module without going through the main processor. The display module possibly converts the data stream and forwards it to its screen memory. The result is that instead of the eight data transactions (of which two over a large bus) that were needed in the traditional architecture, in the connection-centric architecture only two local transactions are required (network interface to switch, switch to display

module). The CPU module is mainly used to initiate the connections, and can be idling during the transaction.

In a connection-centric architecture, each connection can be associated with a certain QoS using a connection identifier. This identifier provides the mechanism to support lightweight protocols that provide data-specific transport services that are associated with a certain QoS. The careful design and fine-tuning process that is needed by system designers for the memory centric architecture, is not needed if QoS can be guaranteed throughout the whole system.

3.3.4 Application domain specific modules

Figure 1 gives a schematic overview of the *Mobile Digital Companion* architecture. In it, we distinguish a switch surrounded by several modules that are each optimised for a certain application domain. Moreover, these modules are reconfigurable, so that they are able to be adapted when this is required.

In general there is always a module with a general-purpose main processor that performs control-type operations. Other modules can be devices like display controllers, network interfaces and stable storage. The architecture comprises many devices normally found in multimedia workstations, but since our target is a portable computer, these devices generally do not have the performance and size of their workstation counterparts. Our ultimate target is to have a *system-on-a-chi*p, where all the functionality of a system is integrated on a single chip.

Note that our devices are not merely dedicated I/O devices in the traditional sense. We prefer to call the devices *modules*, or *I/O subsystems*, to emphasise the fact that they provide more functionality than a simple device. The modules differentiate to I/O devices in multiple ways. First, each module is an autonomous sub-system that can operate without intervention from the main CPU. Second, it has a control processor that performs diverse operations, including connection management and energy management. Finally, most modules are able to adapt their behaviour autonomously to the ‘wishes’ of the client or application, and try to operate in the most efficient way.

Advantages – It is often more sensible to implement device (or media) specific adaptation layers within the module, rather than requiring the network interface or the main CPU to implement a plethora of different adaptation layers [7]. The main reasons for this are:

- *Efficient processing* – The modules are capable of efficiently performing device or application specific tasks. It can for example decompress a video stream, just before it is displayed on the screen (this is a typical example of the locality of reference principle). Dedicated modules can be optimised to execute specific tasks efficiently, with minimal energy overhead. Instead of executing all computations in a general-purpose datapath, as is commonly done in conventional programmable architectures, the energy- and computation-intensive tasks are executed on optimised modules. For example, even when the application-specific coprocessor consumes more power than the processor, it may accomplish the same task in far less time, resulting in net energy savings. The processor can, for example, be

offloaded with tasks like JPEG and MP3 decoding, encryption, and some network protocol handling. A system designer can apply an application-specific coprocessor (e.g. custom hardware) for those tasks the module can handle efficiently, and use the processor for those portions of the algorithm for which the hardware is not well suited (e.g. initialisation).

- *Eliminate useless data copies* – When the data flows directly between the modules that need to process them, unnecessary data copies can be eliminated. Eliminating unnecessary data copies can reduce the traffic on the bus. For example, in a system where a stream of video data is to be displayed on a screen, the data can be copied directly from the network into the screen memory, without going through the main processor.
- *Relieve the general-purpose CPU* – In a connection-centric system data can flow between modules without any involvement of the main CPU and without using any processor cycles. The main CPU is also relieved of having to service interrupts and to perform context switches every time new data arrives, or needs to be sent from a local device. Instead of having one central system that needs to control and process fine-grained operations, a distributed control system is less complex. It is easier to provide real-time support for devices if a dedicated processor controls them.
- *Easy adaptations* – The modules can easily adapt their behaviour. If a module adapts it behaviour, it is able to react on changes in the environment, either imposed by the user (when it starts a new or different application) or by resource changes (for example when the network module notices a change in the wireless channel conditions).
- *Adequate energy management* – Each module contains specific knowledge about the usage patterns and the specific requirements for a device. Therefore, each module has its own responsibility and has some autonomy in deciding how to manage its state of operation to minimise its energy consumption without compromising its quality of service. This is in contrast to current systems in which the main CPU can control the power-state of the connected devices. We have given the modules their own responsibility in deciding how to manage their resources. The individual modules are controlled by a global energy policy that makes high-level decisions on the state of the entire system.
- *Flexible and adaptable* – Because the modules are programmable, they can offer the flexibility to provide support for various standards that a Companion might need to use (e.g. different encoding and encryption schemes), and the adaptability to adapt its mechanisms, algorithms and techniques to the various operating conditions. Of all the programmable modules, the general-purpose processor is merely the most flexible one. The processor will be used for all tasks that the application specific modules are not capable of, or when the implementation would not be efficient. The general-purpose processor will perform all computations that are too complex or would require too much area if they were implemented with the hardware modules. The general-purpose processor can thus also be seen as an application domain

specific module: its domain covers all areas that are not covered by the other modules.

Of course there are also some disadvantages. Most of the trade-offs involved have already been discussed in Section 3.2.6. The most apparent disadvantage seems to be that having application domain specific modules requires more hardware. Instead of processing all tasks on one general-purpose processor, these tasks are distributed over several modules. However, it is expected that the advance in technology give enough possibilities to take advantage of the increased effective chip-area and provide more functionality while keeping the energy consumption low.

Examples of modules

The modules can be very diverse and have different characteristics and requirements. The modules must be capable of handling connections with other modules. They typically contain some intelligence for connection setup and energy management. Typical examples of modules are:

- *CPU-module* – This is the module that can perform general-purpose applications, and provides a broad range of services. One important task of this module is that is responsible for connection management. If a connection has to be established that requires some quality of service guarantees, then the CPU module negotiates with the modules that take part in the connection. The CPU-module is the central place where all QoS related connections are managed.
- *Network module* – This module provides the access to and from the external (wireless) network. In our research we have developed an energy efficient network module that can handle multimedia traffic. This module is a major topic of Chapter 5. The module is able to route traffic according to the VCI of the ATM cells directly to the destination module. The base station plays a crucial role, since it handles most of the network protocol stack layers in the communication over the wired network in lieu of the mobile. In this way the wireless channel peculiarities are decoupled from the network protocol layers, and can provide a more efficient communication [6]. The base station can also act as a proxy server that adjusts the data to the format the mobile can use in an efficient way. For example, a video stream from the fixed network is converted in the base station to a format that the display can easily interpret [74]. The communication between the network module and the base station uses an energy efficient MAC protocol (E^2MaC) that is able to provide QoS guarantees over the wireless channel [31]. The architecture of the network module uses a dynamic error control adapted to the QoS and traffic type of a connection, and has dedicated connection queues and flow control for each connection.

  Note that while we propose to eliminate the dependency on software-based protocol stacks from the mobile, there is no reason to dogmatically preclude the involvement of the general-purpose processor. For example, for research purposes, it is desirable to have the ability to develop and test new algorithms and protocols. In a sense the stack remains a software stack: the devices are programmable. Traditional disadvantages of hardware stacks (inflexibility, cost in concrete) do not hold here.

- *Display module* – On a portable computer the display will generally be small and have a low resolution. The best approach is to adapt the data that is transmitted to the display module such that it can easily interpret the data and display it directly on the screen. If the display module contains an decompression engine, then energy and bandwidth can be saved because no uncompressed data would have to traverse the interconnect, and possibly also traverse the wireless network.
- *Reconfigurable computing module* – This module is basically a device that is capable to handle a wide variety of services. The module contains reconfigurable logic that can be (re)configured dynamically to the requirements. A typical example where such a module can provide a large flexibility is when it is used as an encryption/decryption engine. Data destined or coming from the fixed network could be made to 'pass through' the encryption device on the way to the destination module.

The architecture is modular and can be extended with modules that have a different functionality.

3.3.5 The interconnection network

The interconnection network is a key component for providing flexibility in reconfigurable systems [76]. All modules in the system communicate over a reconfigurable communication network that is organised as a switch. Conceptually, the architecture is analogous to a self-routing packet switch. The exact implementation of the interconnect is not a vital issue in the architecture of the *Mobile Digital Companion*. Just as rings, crossbars and busses have all been used in ATM switches [7], so they may be used in the Companion (although they differ in their complexity and energy consumption). It is the connection-oriented approach using fixed sized cells and the asynchronous multiplexing that are key factors. 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.

The switch interconnects the modules and provides a reliable path for communication between modules. Addressing is based on connections rather than memory addresses. This not only eliminates the need to transfer a large number of address bits per access, it also gives the system the possibility to control the QoS of a task down to the communication infrastructure. This is an important requirement since in a QoS architecture all system components, hardware as well as software, have to be covered end-to-end along the way from the source to the destination.

QoS is a general theme in our research. It is used as a means to deal with the dynamic behaviour of the communication channels, and to be able to provide the various streams in a multimedia mobile computer a satisfactory quality at the lowest cost. The whole system is based on connections between modules. Each connection is associated with a certain QoS. Applications must indicate the QoS is expects from the system, and its ability and willingness to change these QoS requirements. All modules communicate using these connections only. The network module uses the same mechanism to communicate over the wireless channel with a base station that is connected to a wired

environment. If the wired environment also provides mechanisms that are able to deal with QoS (like an ATM network), then we are in principal able to establish an *end-to-end QoS*. In this way we are able to establish a connection from applications on the wired network, through the wireless network, right to the destination where the data will be processed. In the path from source to destination we apply the QoS demands to the various resources involved including the error control over the wireless channel, the communication link layer, and the medium access protocol. The goal is to satisfy the QoS requirements of the connections at the lowest energy consumption. The consequences for the operating system are sketched in Section 3.3.8.

In our infrastructure all connections are identified with a connection identifier which is used to identify the type of data, and the module destination address. This identifier provides the mechanism to support lightweight protocols that provide data-specific transport services that are associated with a certain QoS.

An architecture in which a generalised packet switched interconnect is used to connect processors, memories, and devices has widely come to be known as a 'desk-area network' (DAN). Leslie, McAuley and Tennenhouse first introduced the concept of DAN [40]. We have adopted this concept, and will show that such an architecture is also suitable for low-power portable computers. Our architecture has therefore some similarities to for example the *Desk Area Network* from Cambridge [32], *VuNet* from MIT [34] and the *APIC* architecture from the University of Washington [18]. However, their main motivation was performance and interoperability between (ATM [54]) networks and devices. Our main motivation is reducing energy consumption and not only performance.

Ultimately, the architecture should be implemented in just a single chip. Therefore, we would not call the architecture a *Desk*-area network, but merely a *Chip*-area network. In the *Rattlesnake* ATM switch we already showed that it is quite feasible to build a cost effective ATM switching system that meets multimedia requirements [27][61]. Some of the ideas that were introduced in the architecture of the Rattlesnake switch have been used in the design of the *Octopus* switch that is subject of Chapter 4.

Every device is equipped with an interface to the ATM interconnect. This attachment is no more complicated than the equivalent bus interface, and even simpler than complicated high performance busses (Chapter 4). The header of a standard ATM cell is subdivided into a virtual path and a virtual channel. Within the *Mobile Digital Companion*, this subdivision is not significant. If we would generalise our architecture so that it becomes a full-fledged ATM network that can interface with different ATM devices, then we arrive at a more futuristic DAN-based architecture. While such a system would likely deliver a higher performance than the architecture of the *Mobile Digital Companion*, it does not come without some drawbacks. Among these is the high energy consumption that would needed to implement the full-blown ATM network stack. Furthermore, the costs would be higher because important considerations when designing a network architecture are scalability and tolerance to malfunctioning links and nodes. In a chip-area network the trust boundaries and the operating conditions are much different from those of the desk-area network and local or wide-area networks. Larger networks need protection from hostile or faulty clients and a great amount of

processing power must be put into devices to manage control and security functions. An interconnect architecture designed for interconnecting the components of a mobile system has to fulfil less stringent requirements. In our architecture, the end nodes are dedicated to one work environment, they can be controlled more easily, they can be trusted to be fair and do not exceed their resource allocations.

3.3.6 Energy analysis

In this section we will evaluate the energy impact of a connection-centric architecture based on a switch, compared to a memory-centric architecture based on a shared bus. Figure 5 shows the two architectures. In this analysis we will only deal with the energy effects of communication. Note that the energy consumption required for communication is only a small fraction of the total energy consumption of a system using a typical multimedia application (i.e. 1/10th, see Chapter 6[2]).

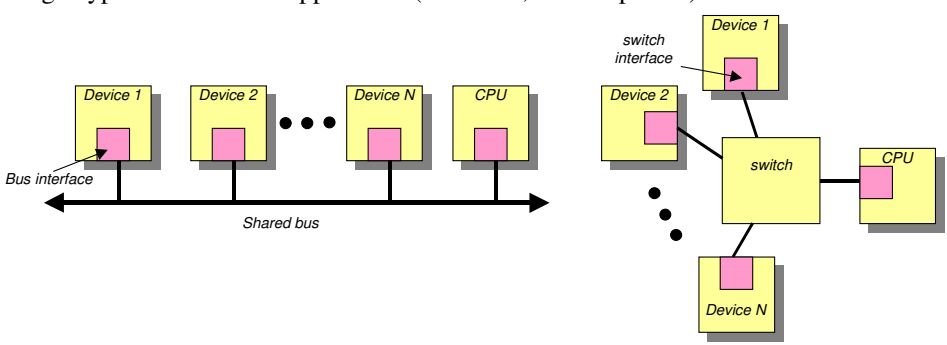


Figure 5: Shared bus architecture and switched architecture.

From Chapter 2 we know that a first order approximation of the dynamic energy consumption of CMOS circuitry is given by the formula:

$$P_d = C_{eff} V^2 f \qquad (1)$$

where P_d is the energy consumption in Watts, C_{eff} is the effective switching capacitance in Farads, V is the supply voltage in Volts, and f is the frequency of operations in Hertz. C_{eff} combines two factors, the capacitance C being charged/discharged, and the *activity weighting* α, which is the probability that a transition occurs.

$$C_{eff} = \alpha C \qquad (2)$$

Most parameters in these equations are affected by the choice of the architecture. In our analysis we make a few assumptions.

---

[2] In Chapter 6 we will evaluate the energy consumption of a mobile system for typical applications.

1. The elements we will consider to contribute to the energy consumption in both architectures are the energy consumption of connection interfaces (P_{bi} for the bus interface and P_{si} for the switch interface) and the energy consumption caused by the wiring (P_{bw} for each interface to the bus, and P_{sw} for each interface to the switch). The energy consumption caused by the switching fabric is P_{switch} per device interface.
2. The shared bus architecture is memory centric, which implies that to transfer a packet an address has to be provided. We will assume that the bus interface controller provides a burst mode, such that only one address is required for the whole packet. The switched architecture is connection centric, which implies that a connection identifier has to be provided per packet. In our analysis we will neglect the differences caused by these addressing schemes.
3. We assume that the N devices (modules) have a half-duplex connection. This implies that the aggregate throughput is thus maximal $N/2$ times the throughput of a single connection. For example, in the Octopus architecture there can be four simultaneous connections when the source and destination of these connections are disjoint. However, because connections in a system are not always disjoint, we will assume an average aggregate throughput of $N/4$.
4. We assume in our analysis that the complexity of bus interface logic is equal to that of a switch interface. Note that it is, however, more likely that the bus interface will be more complex. This is because a bus interface needs to be flexible (because it must be capable to handle a wide variety of devices) and have a high performance (because the bus is shared it for instance needs to implement burst data transfer modes to achieve the required data rates). Because a bus interface has to operate at a higher frequency, the energy consumption of a bus interface will be higher. Using Assumption 3 we will assume that the energy consumption of a bus interface is $N/4$ times the energy consumption of a switch interface, thus $P_{bi} = N/4 \,.\, P_{si}$.
5. From Equation (1) we know that there is a linear dependence of capacitance on the energy consumption. The capacitive load C_{out} of a CMOS logic gate G consists mainly of a) gate capacitance C_{fo} of transistors in gates driven by G, b) capacitance C_w of the wires that connect the gates and c) parasitic capacitance C_p of the transistors in gate G [8]. In symbols:

$$C_{out} = C_{fo} + C_w + C_p \qquad (3)$$

The fan-out capacitance depends on the number of logic gates driven by G and the dimensions of their transistors. In a bus architecture, this number is equivalent to the number of connected devices (including the CPU) N. The size of the transistors that need to drive a high speed bus is also larger than a transistor that only needs to drive one gate. It is extremely hard to estimate C_w accurately because it depends on the topology and routing of the wires and their size. Coupling between wires is becoming the most important factor for the wiring capacitance. The wiring capacitance dominates C_{out} for busses. The parasitic capacitance C_p is probably the

component causing the least concern, as it is relatively small compared to the other two contributions.

Taking these considerations into account, we will assume in our analysis that the capacity per device on the shared bus (C_{bw}) is twice of that capacity of the wires on a switch architecture (C_{sw}), thus $C_{bw} = 2\ C_{sw}$.

6. As described in Section 3.3.3, the shared bus architecture requires at least one, and in most situations two data transfers over the shared bus for a stream between two devices. We will assume that the data transfers over the interconnect are based on Direct Memory Access (DMA) performed by DMA controllers of the devices. The number of DMA copies is D. The complexity of the DMA controllers that are required for the shared bus architecture and for the switched architecture is assumed to be the same.
7. We ignore in this analysis the energy consumption that is due to the CPU controlling the data flow. Note, however, that this can be a significant part of the total energy consumption since in a shared bus architecture where the CPU has to manage the data flow in the machine, the CPU cannot enter a sleep mode for a reasonable time. In the switched architecture, the CPU is out of the data path and can enter sleep mode whenever the connection has been setup. We further ignore the energy consumption caused by the intermediate buffering in the CPU-node.
8. Voltage scaling is an effective way to reduce energy consumption. A switched architecture can operate at a lower voltage than a shared bus architecture because it can operate at a lower speed. In a shared bus architecture all data streams must be performed sequentially using one shared resource (the bus). To achieve the required throughput, the bus has to be fast. The switched architecture allows several parallel data streams. Because of this, the bandwidth of the individual connections in a switched architecture does not have to be as high as the bandwidth in a shared bus architecture. Because of the lower bandwidth required for the connections on a switched architecture, the voltage can be reduced. The delay of a CMOS inverter can be described by the following formula [8]:

$$T_d = \frac{C_{out} V_{dd}}{I} = \frac{C_{out} V_{dd}}{\eta (W/L)(V_{dd} - V_t)^2} \qquad (4)$$

where η is a technology-dependent constant, W and L are respectively the transistor width and length, and V_t is the threshold voltage. Many simplifying assumptions are made in the derivation of Equation (4). Nevertheless, the equation contains the variables on which gate delay actually depends, and the nature of their effect is correctly represented.

We will assume that the lower required bandwidth of the switched architecture (Assumption 3), together with the lower capacity of the wires of the switch, corresponds to a potential reduction in voltage of 50% ($V_{bus} = 2\ V_{switch}$), corresponding to a four times lower energy consumption.

The energy consumption P_{ba} required to transfer a certain amount of data (a packet) over a shared bus architecture with N devices is given by:

$$P_{ba} = N \,.\, D \,.\, (P_{bi} + P_{bw}) \qquad (5)$$

Similarly, the energy consumption P_{sa} required to transfer a packet over a switched architecture (from source device to switch, and from switch to sink device) is given by:

$$P_{sa} = 2 \,.\, (P_{si} + P_{sw} + P_{switch}) \qquad (6)$$

in which P_{switch} is the energy consumption of the switching fabric per interface connection. This leads to the following ratio R between the energy consumption of a shared bus architecture and the energy consumption of a switched architecture.

$$R = \frac{P_{ba}}{P_{sa}} = \frac{N \,.\, D\,(P_{bi} + P_{bw})}{2\,(P_{si} + P_{sw} + P_{switch})} \qquad (7)$$

P_{bw} and P_{sw} depend on the capacity and voltage of the wires. When we incorporate the effects of Assumption 5 ($C_{bw} = 2\ C_{sw}$), and 8 ($V_{bus} = 2\ V_{switch}$), we can deduce that $P_{bw} = 8\ P_{sw}$. Using this and Assumption 3 ($P_{bi} = N/4 \,.\, P_{si}$) we can rewrite Equation (7) as:

$$R = \frac{P_{ba}}{P_{sa}} = \frac{N \,.\, D\,(N/4 \,.\, P_{si} + 8\ P_{sw})}{2\,(P_{si} + P_{sw} + P_{switch})} \qquad (8)$$

To give a feeling of the effect of how this equation works out in a real system, we will use the values obtained by our testbed implementation of a switched architecture (the *Octopus* switch that is described in Chapter 4), and compare this with a shared-bus based system with an equivalent performance.

In the testbed of the Octopus architecture we are using a small low-power microcontroller that operates as the interface controller. The power consumption of this microcontroller is about 26 mW (at 3.3V, 33 MHz)[3]. So a reasonable value of P_{si} is thus 26 mW. The energy consumption P_{switch} of the switching fabric is about 15 mW per interface connection of our testbed implementation[4].

To calculate P_{sw} we assume an 8 bits wide interface operating at 3.3 V with a capacitance of 5 pF per wire. When we assume the operating frequency to be 33 MHz and a toggling probability of 0.25 at each clock cycle, then the power dissipation P_{sw} will be: (8 x 5 pF) x (0.25 x 33 MHz) x 3.3^2 V^2 $\approx$ 4 mW.

When we have the number of devices N=8, and the number of DMA transfers on a shared bus D=2, then the ratio R = 1344/90 $\approx$ 15. We can conclude that, even while the

[3] A dedicated controller would consume less energy. Note that a PCI bus controller has a much higher power consumption, e.g. the PCI9060 PCI bus master from PLX technology requires 680 mW to operate [54].

[4] The testbed was designed to be flexible. It is energy efficient, and is not low power (see Chapter 4). A dedicated implementation would consume much less energy.

assumptions we have made are very conservative and that we have used the power consumption of a *testbed* switched architecture, the switched architecture is much more energy efficient than the shared bus architecture.

In the previous discussion we did not consider dynamic power control, but assumed a connection between two devices with a continuous data stream. In situations where this is no activity on the interconnect, a well designed switched architecture is capable of operating at a low-power sleep mode, whereas in the shared bus architecture the bus interface has to be active all the time.

3.3.7 Timing control

The *Mobile Digital Companion* has a connection-centric approach for several reasons like performance, QoS provisions, energy efficiency, and complexity. There are yet other reasons to choose for a connection-centric architecture. These reasons all stem from the timing-control mechanism in the architecture.

Basic timing control structures

The choice and design of a proper timing control structure for a system is a vital and yet a very practical issue. The synchronous timing scheme is often the first choice in the system design because of the low hardware complexity and logic design simplicity. In a *synchronous system* clock signals serve two purposes: as a sequence reference, and as a time reference. As a sequence reference, a clock transition defines the instance at which the system may change state (so that random state changes and interference can be eliminated). As a time reference, the interval between clock level transitions defines a time region during which data can either move between successive processing stages or are processed in stages isolated from others. A clock signal can thus be viewed as a guard that controls when and what is to be done or not to be done. To ensure the correct system operations, a clock distribution scheme must be used to generate logically equivalent clock signals across an entire system. However, clock skews in the system are unavoidable and caused by many random factors such as various signal propagation delays on wires and in logic, capacitive loading variations at different points, and variations in device and process parameters. The clock signal typically drives a large load because it has to reach many sequential elements distributed throughout the chip. Therefore, clock signals have been a notorious source of energy dissipation because of high frequency and load. It has been observed that clock distribution can take up to 45% of the total energy dissipation of a high performance microprocessor [75].

In an adaptive and (re)configurable system, the synchronous timing method suffers from even more problems. In such a system the delay characteristics (in both communication delays and computation delays) may be very different with different configurations, and cannot, or very limited, be estimated in advance. Therefore, it will be very difficult to determine an appropriate clock speed for the system.

Because it is becoming more and more difficult to distribute a proper global clock network over a large area of silicon, and it is increasingly expensive to design an efficient schedule for a synchronous system with millions of transistors, *asynchronous*

design methods might give a solution. Such a self-timed system is built by decomposing the system into a set of combinatorial logic blocks and inserting an asynchronous hand-shaking control between each pair of connected blocks. Because the complexity of the hand-shaking circuits increases drastically with the number of inputs and outputs, the building blocks perform relatively simple functions. Because there is no global clock in such a system, the system performance and energy consumption is data dependent at run-time. Energy is dissipated only when the circuit is active. As a consequence, asynchronous circuits can have remarkable energy performance [9][46]. However, the circuit complexity to implement the handshaking control logic and the required area to implement such a system are relatively high if the size of the associated logic is small. Also, there are extra delays caused by the hand-shaking protocol and the logic needed to implement it. Asynchronous logic has failed to gain acceptance on the circuit level, mainly based on area and performance criteria, but also due to the design difficulty.

Timing control in a connection-centric architecture

A system can in general – and in a connection -centric system in particular – be composed into two essential parts: a set of functional modules and a communication network connecting these modules. The most efficient system with the highest performance can be achieved if both the modules and the communication network are running at their highest possible and/or most efficient performance, and these performances are well matched with each other. If all functional modules and the communication network of the system are timed separately, then there is a better chance to achieve this goal. The feasibility of meeting such a requirement depends not only on the timing scheme, but also on the architecture of the system.

Synchronous and asynchronous design approaches represent two extremes, and many variants in between exist. In a connection-centric system an interesting combination is to use clocks local to individual logic modules for synchronous operation in each module, and an asynchronous protocol between functional modules for asynchronous communication in the interconnection network. Recently several studies (e.g. [2][25]) indicate that it would be worthwhile to consider such an approach to eliminate the necessity of distributing a global clock between block of larger granularity. In this way, the interface circuitry would represent a very small overhead component, and the most energy consuming aspects of synchronous circuitry (i.e. global clock distribution) would be avoided. The timing difficulties in a synchronous system are localised to the logic inside a module, and do not affect the correct data transfer. An interface between the modules and the inter-communication network synchronises the events in a hand-shaking protocol at the input of a synchronous module with a local clock in the module.

3.3.8 Quality of Service framework

Applications must adapt to ever changing environments and they need the help of the operating system to provide the information for it. Traditional operating systems do not tell applications when the network is down, how much communication costs (in terms of cost per bit or energy consumption), or how much CPU resources are available.

Adaptation to available network bandwidth already exists in the context of multimedia communication. It can be very useful for mobile-computing applications as well to be aware of network outages and network communications cost. A *Mobile Digital Companion* may be in a location where communication over the network is expensive and of low bandwidth. When this is the case, a file system (to mention just one example) may do well to adapt its behaviour and stop prefetching to increase performance.

If one investigates by what methods applications can adapt their Quality of Service (QoS), one notices that, in order to bridge substantial changes in resource allocation (CPU, energy consumption and network bandwidth are the resources most affected), merely changing parameters is not sufficient. In a dynamic mobile environment more drastic changes are required, e.g. by changing algorithms. In the MOBY DICK architecture, Quality of Service is a framework to model integration and integrated management of all the system services and applications in the *Mobile Digital Companion*. The consumption of resources by one application might affect other applications, and as resources run out, all applications are affected. If the availability of a resource changes, whether it is a file, CPU cycles, or energy consumption, applications that use them are notified, and they can adapt their behaviour. For example, an application that maintains a distributed diary would request, for its highest QoS, to make use of a consistent view of its files, but, if this cannot be made available due to a network partition, it would accept an inconsistent version as the next best thing. Since communication bandwidth, energy consumption and application behaviour are closely linked, we believe that a QoS framework is a sound basis for integrated management of the resources of the *Mobile Digital Companion*.

The QoS framework influences a large number of parameters of various components in the system. Most of these parameters have also a significant impact on the energy consumption, in general a higher quality requires also more energy. Energy consumption is thus an important parameter in the QoS framework. In order to integrate power awareness in the QoS framework, changes must be made to hardware, drivers, firmware, operating system, and applications. The system needs to be flexible, and have several implementations of a function of which one can be chosen depending on the QoS and available resources. The operating system will control the power states of devices in the system and share this information with applications and users.

One of the key aspects of our QoS approach is to move power management policy decisions to the user and co-ordination 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. This, however, does not imply that modules have no responsibility. 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. To take advantage of low-power modes of the system's modules, the operating system needs to direct these modules to change its power mode when it is predicted that the net savings in energy will be worth the time and overhead of switching over and restarting.

3.4 Related work

In this section we will provide an overview of related work in the various topics that are covered by the architecture of the Mobile Digital Companion: i.e. multimedia architectures, heterogeneous architectures, network attached devices, and energy management.

3.4.1 Multimedia architectures

The problem of hardware architecture design for high-performance processors is a topic that is covered widely in the literature. Various architectures have been proposed to address the problems involved with multimedia computing. These approaches are based on high-performance technology and are mostly simple extensions to current architectures. These systems fail to exploit the opportunities for energy reduction offered by multimedia.

Systems like the *InfoPad* [60][70] and *ParcTab* [36] are designed to take advantage of high-speed wireless networking to reduce the amount of computation required on the portable. These systems are a kind of portable terminal and take advantage of the processing power of remote compute servers. This approach simplifies the design and reduces power consumption for the processing components, but significantly increases the network usage and thus potentially increases energy consumption because the network interface is energy expensive. These systems also rely on the availability of a high bandwidth network connectivity and cannot be used when not connected.

UCLA has constructed a network testbed [43] that uses a hardware architecture to localise data for both communication and video. In this way the data streams are reduced and efficiently transferred directly to their destination. The granularity of this system is much larger than the previous systems. Performance evaluation using the testbed has revealed the relative importance of the overhead incurred by the application and network protocols as well as the signal processing in the video and radio hardware [15]. For a high performance node, the overheads due to bus transfers, memory copies, and network processing are high. Bus transfer is the main source of limitations to system throughput for applications requiring movement of large blocks of data across the system bus.

Recent years show an increase in the use of application specific architectures in the general-purpose world. In this approach frequently used operations that are expensive in computation time are implemented in dedicated hardware inside the microprocessor. The hardware units are often called *hardware accelerators*. A typical example can be found in Intel's *MMX* ™ architecture [39]. To further increase performance several instructions can be performed in parallel, an example can be found in VLIW (Very Large Instruction Word) architectures. The term media processor is often used for a class of multimedia processors, predominantly aimed at the multi-media-PC market. For example, the *TriMedia* processor uses a VLIW architecture with hardware accelerators and a data highway to be able to handle applications such as decompression of real-time audio and video [57]. Although hardware accelerators enable the designer to implement higher-level operations, this level is still limited by the requirement of generally applicable

instructions to support a high degree of programmability. The amount of parallelism that can be obtained at such a level is rather limited, typically a factor of 3 to 5 [38]. The amount energy consumption required is generally no concern for the designers, and is high.

3.4.2 Heterogeneous parallel architectures

By adding special coprocessors next to the general-purpose processor, the grain of operations is increased to the level of complete functions that are executed on dedicated hardware. However, the coprocessors cannot operate independently from the general-purpose processor that performs the synchronisation of tasks. This leads to a significant overhead in execution time and limits the increase in concurrency that can be obtained. Furthermore, a communication bottleneck can easily occur because in multimedia applications that require a large amount of data, the bandwidth that is offered is highly insufficient because all processors must communicate over the same bus. Making use of function-level parallelism can increase the processing performance and efficiency.

Abnous and Rabaey propose an architecture for signal processing applications that is flexible and uses low power [1]. The architecture consists of a control processor surrounded by a heterogeneous array of autonomous, special-purpose satellite processors. The computational demand on the control processor is minimal, its main task is to configure the system and manage the overall control flow of a given signal-processing algorithm. The satellite processors perform the dominant, energy-intensive computational tasks of algorithms. The granularity of these tasks is relatively small. Some examples include address generators, multiply-accumulate processors for computing vector dot products, etc. The architecture does not allow multiplexing of different tasks on the same processor. This restricts the degree of efficiency, since for every task contained in an application a separate processor is required.

Nieuwland and Lippens [52] propose a heterogeneous multiprocessor architecture that supports a global memory model. Such a model allows for easy re-map of current typical programs on heterogeneous processing elements. A bus connects the heterogeneous processing elements. Local memory on the processing elements is positioned within a single global mapping of the application and is accessible by all other processing elements. Due to a well-defined communication interface, allocating tasks to another processing element does not require changes in the remaining application software. Experiments in software show that although the communication protocol runs rather efficient, a significant part of the speed up is lost in communication due to the small grain size of communication with the coprocessor task.

Leijten has proposed a heterogeneous multiprocessor template to be able to obtain a processing performance [38]. This is obtained by replacing the coprocessors by processors that have their own thread of control, that is, autonomous processors can execute tasks completely independently from the microprocessor. In the resulting multiprocessor solution the general-purpose microprocessor executes low-performance tasks requiring a high degree of programmability, while the other processors execute high-performance tasks requiring only limited programmability. These high-

performance processors are *application-domain-specific* (ADS) processors optimised in terms of speed, area and power, and tuned towards a well-defined set of tasks. The granularity of the operations of processors is relatively small, the main target of the system is to implement a multimedia processor.

The University of Twente has developed an architecture that is suitable for reconfigurable low-power DSP-like algorithms. *Field-Programmable* ***Function*** *Array* (FP*F*A) devices are reminiscent of FPGAs, but have a matrix of ALUs and lookup tables instead of CLBs (Configurable Logic Blocks). The construction of an ALU from multiple 1-bit-wide lookup tables is energy inefficient [64]. For a wide range of multimedia functions that use digital filtering algorithms on parallel data: video (de)compression, data encryption and digital signatures these devices do not posses the required processing power. For these functions 16/32 bit calculations (multiply, add) are required. Newer architectures are based on 'chunky' function units such as complete ALUs and multipliers. For example, a collection of multipliers might be available along with a crossbar interconnect to efficiently support a wide range of infinite-impulse response (IIR) filters. These architectures present an abstraction that is much higher than logic gates and flip-flops, but highly irregular computations will likely be a poor match.

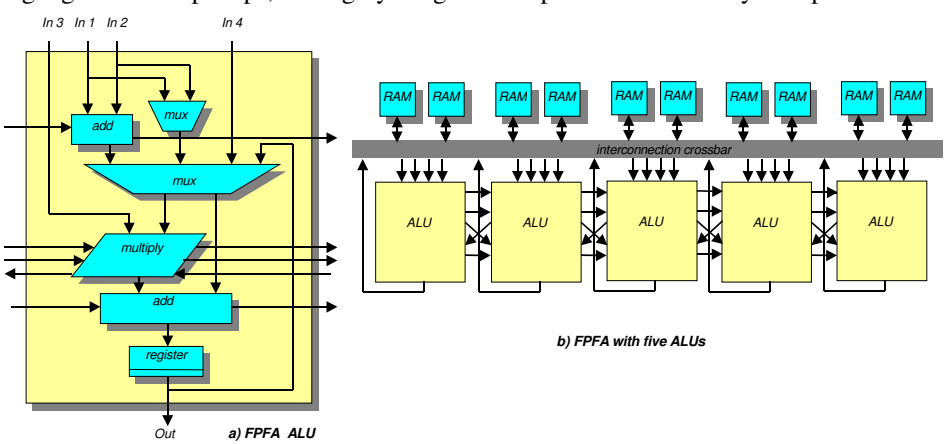


Figure 6: FPFA architecture.

The instruction set of an FPFA-ALU can be thought of as the set of ordinary ALU instructions, with the exception that there are no load and store operations which operate on memories. Instead, they operate on the programmable interconnect; that is, the ALU loads its operands from neighbouring ALU outputs, or from (input) values stored in lookup tables or local registers. The graph-based execution of the FPFA is used to execute the inner loop of an application. The regular, general-purpose structure of the device makes a rapid context switch from one inner loop to another possible, hence on-the-fly reconfiguration. This is how a broad class of compute intensive algorithms can be implemented on an FPFA [63].

At the M.I.T. Laboratory for Computer Science a new architecture is being developed that eliminates the traditional instruction set interface and instead uses a replicated

architecture directly to the compiler [73]. This allows the compiler to determine and implement the best resource allocation for each application. They call systems based on that approach *Raw architectures* because they implement only a minimal set of mechanisms in hardware. The architecture is based on a set of interconnected tiles, each of which contains instruction and data memories, an arithmetic logic unit, registers, configurable logic, and a programmable switch that supports both dynamic and compiler-orchestrated static-routing. Raw architectures are best suited for stream-based signal-processing computations.

The TMS320C80 device is a single-chip, parallel processor intended for applications such as real-time audio/video processing, high-end data communications, and image processing [69]. This chip proves the possibility of placing multiple interconnected processors on a single chip. This complex device contains four parallel processing DSPs (PP) with 64-bit instructions and 32-bit fixed-point data; a RISC master processor (MP) with a floating point unit; 50 Kbytes of on-chip RAM; a Video Controller; and a transfer controller that services data requests and cache misses by the MP and PPs. A crossbar switch provides the access of the MP and PPs to on-chip memory.

3.4.3 Network attached devices

A network-attached peripheral (NAP) is a computer peripheral that communicates via a network rather than a traditional I/O bus, such as SCSI [20]. Several research projects using network attached peripherals in multimedia workstations are ongoing in various universities. The canonical example of the uses for NAPs in multimedia is the desire to transmit data directly from a camera to a frame buffer without passing through the system's backplane, where it unproductively consumes bandwidth. Captures of video to disk and playback from disk are similar. We will now mention only some typical examples.

Desk Area Network – One way to provide real-time guarantees when transferring data inside a workstation is to also use an ATM network switch to interconnect the components of a workstation system. This work has been done at Cambridge [32] and some of this work has now been commercialised by Nemesys. The Desk-Area Network (DAN) carries this idea to the extreme in that the ATM switches are also used to interconnect the memory.

VuNet – The VuNet architecture was designed as part of the ViewStation multimedia project at MIT [4]. The VuNet is a gigabit per second network using ATM, which interconnects general-purpose workstations and multimedia devices. The VuNet is intended to be used as both a desk-area and local-area network. In their approach the multimedia information is channelled all to the workstation processor rather than bypassing it with specialised hardware. They expect that given the current rate of progression of workstation performance, performance levels that allow multimedia tasks to execute in parallel with other tasks, will be reached soon. Due to the software intensive approach to multimedia, the VuNet and custom video hardware were designed to provide efficient support for software driven handling of multimedia streams. The VuNet switch fabric is constructed out of a high performance 4 port crossbar chip and

FIFOs that can buffer 64 cells in the transmit direction and 256 cells on the receive direction.

Switcherland – This scalable communication architecture is based on crossbar switches that provide QoS guarantees for workstation clusters in the form of reserved bandwidth and bounded transmission delays [21]. Similar to ATM technology Switcherland provides QoS guarantees with the help of service classes. Their main target is to provide a high performance and good availability of processors and I/O devices by allowing any arbitrary topology. The switches can be used as an I/O interconnection fabric of a workstation as well as a network interconnection fabric of a workstation cluster.

3.4.4 Energy management

One of the most successful techniques employed by designers of current computers at the system level is dynamic power management [8]. There are, however, few operating systems designed specifically for portable computing equipment. Microsoft's Windows CE [26] is one, USRobotics PalmOS [71] is another. The power management in these systems consists almost exclusively of powering down the CPU and other devices when the system becomes idle and turning off the screen after a few minutes of user inactivity.

Currently several system developers and vendors are pursuing a long-term, wide scope strategy to greatly simplify the task of large and complex power-managed systems. The strategy is based on a standardisation initiative known as *Advanced Configuration and Power Interface* (ACPI). The *OnNow* initiative targets the migration of power management algorithms and policies into the computer's operating system [49]. OnNow and ACPI provide a framework for designers to implement power management strategies. The choice of the policy is left to the designer. OnNow is an initiative proposed by a single software company, and is tightly bound to the abstract model of a personal computer. Although OnNow requires ACPI as the interface between the operating system and the hardware, ACPI is more general in scope and does not depend on any operating system or hardware model. However, both ACPI and OnNow assume a CPU and operating system centric system, where the activities of the system are managed by a single entity. ACPI and OnNow are developed to support the implementation of power managed computer systems, and are too detailed to effectively support design exploration [8].

A modelling approach that is aimed at providing support for system-level architectural exploration of power-managed systems is described in [8]. In their model, a system is defined by a set of components and a communication pattern between components. Communication is modelled by abstract events. The abstraction of the model is much higher than ACPI, and no details are specified about the functional behaviour of a service provider (like disk driver unit or video driver).

3.5 Summary and conclusions

In this chapter we considered the problem of designing an architecture for a handheld mobile multimedia computer. The architecture of the *Mobile Digital Companion* is connection-centric in which the modules communicate using an asynchronous hand-shaking interface. These modules can be combinatorial or controlled by clocks local to each module. The CPU is moved out of the data stream, although it still participates in the control flow. Such a design approach offers a solution in the design of multimedia, low-power wireless terminals. The architecture presents several advantages over the traditional memory-centric models.

Energy management is the general theme in the design of the system architecture since battery life is limited and battery weight is an important factor. We have shown that there is a vital relationship between hardware architecture, operating system architecture and applications architecture, where each benefits from the others. In our architecture we have applied several supplementary energy reduction techniques on all levels of the system. Achieving high energy efficiency requires first of all the elimination of the waste that typically dominates the energy consumption in general-purpose processors. The second main principle used is to have a high locality of reference. The philosophy is that all operations that are required on the data should be done at the place where it the most efficient, thereby also minimising the transport of data through the system.

As the Mobile Digital Companion must remain usable in a wide variety of environments, it must be flexible enough to accommodate a variety of multimedia services and communication capabilities and adapt to various operating conditions in an (energy) efficient way. The approach made to achieve such a system is to use autonomous, adaptable components, interconnected by a switch rather than by a bus, and to offload as much as work as possible from the CPU to programmable modules that is placed in the data streams. Thus, communication between modules is delivered exactly to where it is needed, work is carried out where the data passes through, bypassing the ‘main’ memory, modules are autonomously entering an energy-conservation mode and adapt themselves to the current state of the resources and the requirements of the user. If buffering is required at all, it is placed right on the data path, where it is needed. The application domain specific modules offer enough flexibility to be able to implement a predefined set of (usually) similar applications, while keeping the costs in terms of area and energy consumption to an acceptable low level.

Having an energy-efficient architecture that is capable of handling adaptability and flexibility in a mobile multimedia environment requires more than just a suitable hardware platform. First of all we need to have an operating system architecture that can deal with the hardware platform and the adaptability and flexibility of its devices. Optimisations across diverse layers and functions, not only at the operating systems level, is crucial. Managing and exploiting this diversity is the key system design problem. A model that encompasses different levels of granularity of the system is essential in the design of an energy management system and in assisting the system designer in making the right decisions in the many trade-offs that can be made in the

system design. Finally, to fully exploit the possibilities offered by the reconfigurable hardware, we need to have proper operating system support for reconfigurable computing, so that these components can be reprogrammed adequate when the system or the application can benefit from it.

Although our design assumes a low-power, wireless multimedia computer, most of our ideas are applicable (perhaps with some modification) to many other types of computer (sub)systems, including high performance workstations and network interfaces.

References

[1] Abnous A., Seno K., Ichikawa Y., Wan M., Rabaey J.: "Evaluation of a low-power reconfigurable DSP architecture", *proceedings 5th Reconfigurable Architectures workshop (RAW'98)*, March 30, 1998, Orlando, USA. (URL: http://xputers.informatik.uni-kl.de/RAW/RAW98/adv_prg_RAW98.html)

[2] Abnous A., Rabaey J.: "Ultra-low-power domain-specific multimedia processors", *VLSI Signal processing IX*, ed. W. Burleson et al., IEEE Press, pp. 459-468, November 1996.

[3] Adam J.: "Interactive multimedia – applications, implications", *IEEE Spectrum*, pp. 24-29, March 1993.

[4] Adam J.F., Houh H.H., Tennenhouse D.L.: "Experience with the VuNet: a network architecture for a distributed multimedia system", *Proceedings of the IEEE 18th Conference on Local Computer Networks*, pp. 70-76, Minneapolis MN, September 1993.

[5] Agarwal A.: "Raw computation", *Scientific American*, pp. 44-47, August 1999.

[6] Balakrishnan H., et al.: "A comparison of mechanisms for improving TCP performance over wireless links", Proceedings *ACM SIGCOMM'96*, Stanford, CA, USA, August 1996.

[7] Barham P., Hayter M., McAuley D., Pratt I.: "Devices on the Desk Area Network", March 1994.

[8] Benini L., De Micheli G.: "Dynamic Power Management, design techniques and CAD tools", *Kluwer Academic Publishers*, ISBN 0-7923-8086-X, 1998.

[9] Berkel K., et al.: "A fully asynchronous low power error corrector for the DCC player", *Digest of Technical Papers, International Solid-State Circuit Conference*, pp. 88-89, 1994.

[10] Bhoedjang, R.A.F., Rühl T., Bal H.E.: "User-level network interface protocols", *Computer*, November 1998, pp. 53-60.

[11] Bosch P.: "Mixed-media file systems", *Ph.D. Thesis University of Twente*, June 1999.

[12] Burger D., Goodman J.: "Billion-transistor architectures", *Computer*, Sept. 1997, pp. 46-47.

[13] Chaiken D., Hayter M., Kistler J., Redell D.: "The Virtual Book", *SRC Research report 157*, Digital Systems Research Center, November 1998.

[14] Chandrakasan A., Brodersen R.W.: "A Portable Multimedia Terminal", *IEEE Communications Magazine*, pp. 64-75, vol. 30, no. 12, Dec. 1992.

[15] Chien C., et al.: "An integrated testbed for wireless multimedia computing", *Journal of VLSI Processing Systems* 13, pp. 105-124, 1996.

[16] Dally W.: "Tomorrow's Computing Engines", keynote speech, *Fourth International symposium High-performance Computer Architecture*, Feb. 1998.

[17] Diependorff K., Dubey P.: "How multimedia workloads will change processor design", *Computer*, Sept. 1997, pp.43-45.

[18] Ditta Z.D., Cox R.C., Parulkar G.M.: "Catching up with the networks: host I/O at gigabit rates", *Technical report WUCS-94-11*, Washington University in St. Louis, April 1994.

[19] Dorward S., Pike R., Presotto D., Ritchie D., Trickey H., Winterbottom P.: "Inferno", *Proceedings COMPCON Spring'97*, 42nd IEEE International Computer Conference, 1997, URL: http://www.lucent.com/inferno.

[20] Doyle van Meter, R.: "A brief survey of current work on network attached peripherals", *ACM Operating Systems Review*, Jan. 1996.

[21] Eberle H., Oertli E.: "Switcherland: a QoS communication architecture for workstation clusters", *Proceedings ISCA '98 – 25th annual Int. Symposium on Computer Architecture*, Barcelona, June 1998.

[22] von Eicken, T., Vogels, W.: "Evolution of the Virtual Interface Architecture", *Computer*, pp. 61-68, November 1998

[23] Estrin G.: "Organization of Computer Systems: The Fixed-plus Variable Structure Computer", *Proceedings of the Western Joint Computer Conference*, pp. 33-40, 1960.

[24] Flynn M.J.: "What's ahead in computer design?", *proceedings Euromicro 97*, pp. 4-9, September 1997.

[25] Gao B., Rees D.R.: "Communicating synchronous logic modules", *21th Euromicro conference*, September 1995.

[26] O'Hara, R.: "Microsoft Windows CE: History and Design", *Handheld systems 5.1*, Jan./Feb. 1997, available at http://www.cdpubs.com/Excerpts.html.

[27] Havinga P.J.M., Smit G.J.M.: "Rattlesnake – a single chip high-performance ATM switch", *proceedings International conference on multimedia networking (MmNet'95)*, pp. 208-217, Aizu, Japan, September 26-29, 1996.

[28] Havinga, P.J.M., Smit, G.J.M.: "Minimizing energy consumption for wireless computers in Moby Dick", *proceedings IEEE International Conference on Personal Wireless Communication ICPWC'97*, Dec. 1997.

[29] Havinga P.J.M., Smit G.J.M.: "Low power system design techniques for mobile computers", *CTIT technical report series 97-32*, Enschede, the Netherlands, 1997

[30] Havinga P.J.M., Smit G.J.M.: "The Pocket Companion's architecture", *Euromicro summer school on mobile computing '98*, Oulu, pp. 25-34, August 1998

[31] Havinga P.J.M., Smit G.J.M., Bos M.: "Energy efficient wireless ATM design", *proceedings wmATM'99*, June 1999.

[32] Hayter M.D., McAuley D.R.: "The desk area network", *ACM Operating systems review*, Vol. 25 No 4, pp. 14-21, October 1991.

[33] Helme A.: "A system for secure user-controlled electronic transactions", *PhD. thesis University of Twente*, August 1997.

[34] H.H. Houh, Adam J.F., Ismert M., Lindblad C.J., Tennenhouse D.L.: "The VuNet desk area network: architecture, implementation and experience", *IEEE Journal of Selected Areas in Communications* (JSAC), 13(4):710-121, May 1995 (see also: http://www.tns.lcs.mit.edu/ViewStation/src/html/publications/JSAC95.html)

[35] Hui J.: "Switching and traffic theory for integrated broadband networks", *Kluwer Academic Press*, 1990.

[36] C. Kantarjiev et al.: "Experiences with X in a wireless environment", *Mobile and location-independent computing symposium*, Cambridge MA, August 1993.

[37] Kozyrakis C.E., Patterson D.A.: “A new direction for computer architecture research”, *Computer*, Nov. 1998, pp. 24-32,

[38] Leijten J.A.J.: “Real-time constrained reconfigurable communication between embedded processors”, *Ph.D. thesis, Eindhoven University of Technology*, November 1998.

[39] Lempel, O., Peleg A., Weiser U.: “Intel’s MMX ™ Technology – a new instruction set extension”, *Proceedings IEEE COMPCON*, pp. 255-259, 1997.

[40] Leslie I., D. McAuley, D. L. Tennenhouse: “ATM Everywhere?”, *IEEE Network*, March 1993.

[41] Lettieri P., Srivastava M.B.: “Advances in wireless terminals”, *IEEE Personal Communications*, pp. 6-19, February 1999.

[42] Lorch J.R.: “A complete picture of the energy consumption of a portable computer”, *Masters thesis, Computer Science, University of California at Berkeley*, 1995.

[43] Mangione-Smith, B. 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 (ISLPED) 1996*, Monterey CA, USA, pp. 23-28, August 1996.

[44] Mangione-Smith W.H., et al.: “Seeking solutions in configurable computing”, *IEEE Computer*, pp. 38-43, December 1997.

[45] Mangione-Smith W.H., Hutchings B.L.: “Configurable computing: the road ahead”, *1997 reconfigurable architectures workshop*, 1997.

[46] Martin A.J., Burns S.M., Lee T.K., Borkovic D., Hazewindus P.J.: “The first asynchronous microprocessor: the test results”, *Computer Architecture News*, 17(4):95-110, June 1989.

[47] Mehra R., Rabaey J.: “Exploiting regularity for low-power design”, *proceedings of the international Conference on computer-aided design*, 1996.

[48] Mehra R., Lidsky D.B., Abnous A., Landman P.E., Rabaey J.M.: “Algorithm and architectural level methodologies for low power”, Section 11 in "*Low power design methodologies*", editors J. Rabaey, M. Pedram, Kluwer Academic Publishers, 1996.

[49] Microsoft: “OnNow and Power Management”, http://microsoft.com/hwdev/onnow.htm.

[50] 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).

[51] Mullender S.J., Smit G.J.M., Havinga P.J.M., Helme A., Hartvigsen G., Fallmur T., Stabell-kulo T., Bartoli A., Dini G., Rizzo L., Avvenuti M.: “The Moby Dick Architecture”, *CTIT Technical report series*, No. 98-18, Enschede, the Netherlands, 1998.

[52] Nieuwland A.K., Lippens P.E.R.: “A heterogeneous HW-SW architecture for hand-held multi-media terminals”, *proceedings IEEE workshop on Signal Processing Systems*, SiPS’98, pp. 113-122.

[53] Pedram M.: “Power minimization in IC design: principles and applications”, *ACM Transactions on Design Automation*, Vol. 1, no. 1, pp. 3-56, Jan 1996.

[54] PLX technology: “PCI9060, PCI Bus master interface chip for adapters and embedded systems”, datasheet, 1995, http://www.plxtech.com/download/9060/datasheets/9060-12.pdf.

[55] Prycker: “Asynchronous Transfer Mode”, 1991.

[56] Rambus Inc.: “Direct Rambus Technology Disclosure, http://www.rambus.com.

[57] Rathnam S., Slavenburg G.: “An architectural overview of the programmable multimedia processor, TM-1”, *Proceedings IEEE COMPCON*, pp. 319-326, 1996.

[58] Reiniger D., Izmailov R., Rajagopalan B., Ott M., Raychaudhuri D.: “Soft QoS control in the WATMnet broadband wireless system”, *IEEE Personal Communications*, pp. 34-43, February 1999.

[59] Rocket eBook, http://www.rocket-ebook.com.

[60] Sheng S., Chandrakasan A., Brodersen R.W.: “A Portable Multimedia Terminal”, *IEEE Communications Magazine*, pp. 64-75, vol. 30, no. 12, Dec., 1992.

[61] Smit G.J.M.: “The design of central switch communication systems for multimedia applications”, *Ph.D. thesis, University of Twente*, 1994.

[62] Smit G.J.M., Havinga P.J.M., et al.: “An overview of the Moby Dick project”, *1st Euromicro summer school on mobile computing*, pp. 159-168, Oulu, August 1998.

[63] Smit J., Bosma M.: “Graphics algorithms on Field Programmable Function Arrays”, *proceedings of the 11th EuroGraphics workshop on graphics hardware*, Eds. B.O. Schneider and A. Schilling, pp.103-108, 1996.

[64] Smit J., Stekelenburg M., Klaassen C.E., Mullender S., Smit G., Havinga P.J.M.: “Low cost & fast turnaround: reconfigurable graph-based execution units”, *proceedings 7th BELSIGN workshop*, Enschede, The Netherlands, May 7-8, 1998.

[65] SoftBook Reader, http://www.softbook.com.

[66] Srivastava M.: “Design and optimization of networked wireless information systems”, *IEEE VLSI workshop*, April 1998.

[67] Steenkiste P.A. Zill B.D., Kung H.T., Schlick S.J., Hughes J., Kowalski B., Mullaney J.: "A host interface architecture for high speed networks", *Proceedings 4th IFIP conference on high performance networking*, pp. A3-1 A3-16, December 1992.

[68] Steenkiste P.: “Design, implementation and evaluation of a single-copy protocol stack”, *Software – practice and experience*, January 1998.

[69] Texas Instruments, SMJ320C80 Digital Signal Processor, http://www.ti.com/sc/docs/products/sm320C80.html.

[70] Truman T.E., Pering T., Doering R., Brodersen R.W.: The InfoPad multimedia terminal: a portable device for wireless information access”, *IEEE transactions on computers*, Vol. 47, No. 10, pp. 1073-1087, October 1998.

[71] USRobotics PalmOS, URL: http://palmpilot.3com.com.

[72] Villasenor J., Mangione-Smith W.H.: “Configurable Computing”, *Scientific American*, June 1997.

[73] Waingold E, Michael Taylor, Devabhaktuni Srikrishna, Vivek Sarkar, Walter Lee, Victor Lee, Jang Kim, Matthew Frank, Peter Finch, Rajeev Barua, Jonathan Babb, Saman Amarasinghe, and Anant Agarwal: “Baring it all to Software: Raw Machines”, *IEEE Computer*, September 1997, pp. 86-93.

[74] Wireless Application Protocol Forum Ltd.: “Official Wireless Application Protocol”, *Wiley Computer Publishing*, 1999, http://www.wapforum.org.

[75] Yeap G.K.: “Practical low power digital VLSI design”, *Kluwer Academic Publishers*, ISBN 0-7923-80.

[76] Zhang H., Wan M., George V., Rabaey J.: “Interconnect architecture exploration for low-energy reconfigurable single-chip DSPs”, *Proceedings of the WVLSI*, Orlando, Fl, April 1999.

4

The Octopus switch

This chapter[1] discusses the interconnection architecture of the Mobile Digital Companion. The approach to build a low-power handheld multimedia computer presented here is to have autonomous, reconfigurable modules such as network, video and audio devices, interconnected by a switch rather than by a bus, and to offload as much as work as possible from the CPU to programmable modules placed in the data streams. Thus, communication between components is not broadcast over a bus but delivered exactly where it is needed, work is carried out where the data passes through, bypassing the memory. The amount of buffering is minimised, and if it is required at all, it is placed right on the data path, where it is needed.

A reconfigurable internal communication network switch called Octopus exploits locality of reference and eliminates wasteful data copies. The switch is implemented as a simplified ATM switch and provides Quality of Service guarantees and enough bandwidth for multimedia applications. We have built a testbed of the architecture, of which we will present performance and energy consumption characteristics.

4.1 Introduction

The interconnection structure of the *Mobile Digital Companion* is based on a switch, called *Octopus*, which interconnects a general-purpose processor, (multimedia) devices, and a wireless network interface. Although not uniquely aimed at the desk-area, our work is related to other projects (like [1][5] and [9]), in which the traditional workstation bus is replaced by a high speed network in order to eliminate the communication bottleneck that exists in current systems.

---

[1] Major parts of this chapter were presented at the fifth annual *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99)* 1999 [8], and at the *ProRISC'99 workshop on Circuits, Systems and Signal Processing*, 1999 [9].

The exact implementation of the interconnect is not a vital issue in the architecture of the *Mobile Digital Companion*. Just as rings, crossbars and busses have all been used in ATM switches [3], so may they be used in the Companion. It is the connection-oriented approach, using fixed sized cells and asynchronous multiplexing as key factors. In most computer systems a single shared link or a bus interconnects the components. The main attraction of a bus is its low cost and low complexity. Drawbacks are high energy consumption, limited extensibility and limited scalability (that is, the consequences of extending busses are high: increased complexity and high cost in many aspects). Due to the electrical properties of busses these limitations become more evident at high speed. Although an interconnect based on a switch cannot compete in terms of cost, it is attractive because of its high aggregate bandwidth, scalability, and low energy consumption. 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 performance.

The *n*-by-*n* switch interconnects *n* modules and provides a reliable path for communication between modules. Addressing is based on connections rather than memory addresses (see Chapter 3). This not only eliminates the need to transfer a large number of address bits per access, it also gives the system the possibility to control the QoS of a task down to the communication infrastructure. This is an important requirement since in a QoS architecture all system components, hardware as well as software, have to be covered end-to-end along the way from the source to the destination. In our infrastructure all connections are associated with a certain QoS.

In an ideal model all modules can communicate with each other over a communication channel of zero length and infinite speed. In our prototype the internal bandwidth will be much more than the maximal throughput of a device (e.g. the wireless interface is capable of transferring 2 Mb/s, and the interconnect has a capacity of 32 Mb/s per connection). By having such high bandwidths on the local interconnect, devices can access other devices (including the network) in much the same way as if the device had exclusive access. The bandwidth reservation on the wireless network, which will probably be the bottleneck in this architecture, can in this way be though of as extending into the mobile, all the way to the device. Since we are using ATM cells not only as basic communication mechanism on the network, but also internally in the architecture of the mobile, we do not need to have any packet conversion as well.

Switching theory

One of the main problems of network design is how to ensure sufficient bandwidth, and thus throughput, for all data streams [13]. A network may be blocking, which means that certain connections cannot be made, because of other connections created earlier. Basically there are two types of switches: *time division* (T) switches and *space division* (S) switches. In an S-switch, physical switches are used to connect input wires to output wires. Physical connections are thus created between input and output channels. In a T-switch, a single physical line is used to transport the different connections. Time is divided into periodic cycles, where each cycle consists of a fixed number of *time slots*. A

time slot is a periodically recurring time interval consisting of a fixed number of clock cycles. Each time slot represents a different channel.

A well-known three stage switching network is the TST network [13]. The first and third stage of this network consists of T-switches, whereas the second stage consists of a *time-shared* S-switch. In a time-shared S-switch new physical connections between input and output channels are created for each time slot.

The communication network is built according to the TST network in which the Octopus switch is a time-shared S-switch, and the modules can function as the T-switch. Such a configuration has important advantages over other configurations when used in processor networks [13]. Because a T-switch has intrinsic buffering, the concept of FIFO buffering to allow synchronisation between modules operating at different data rates is readily implemented in such a switch. The data that is to be produced or consumed by a module can be buffered until sufficient data is available and a time slot is granted to the connection.

Outline

Section 4.2 provides the architecture and design of the interconnection network, the Octopus switch. The switch provides the communication infrastructure between functional modules, and is based on connections of two service classes. Topics of special interest are the buffer organisation, the scheduling techniques used, and the internal communication protocol. In Section 4.3 the prototype implementation of the Octopus switch is described, and performance and energy consumption measurements are presented. Finally, we present the summary and conclusions in Section 4.4.

4.2 Architecture of the Octopus switch

In this section we will present the architecture of the Octopus switch. The key goals motivating the design has been simplicity, flexibility and energy efficiency.

4.2.1 Octopus architecture

The Octopus switch provides the interconnection infrastructure between the functional modules in the system of a Mobile Digital Companion. Figure 1 shows an architectural view of the system of a Mobile Digital Companion.

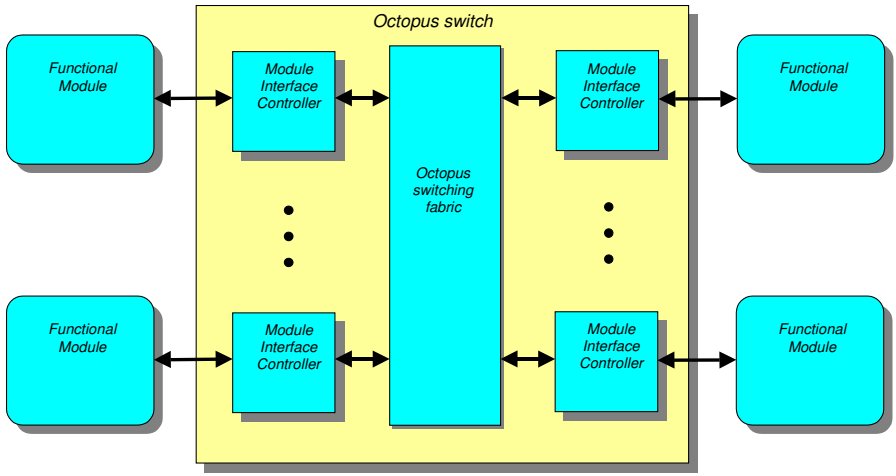


Figure 1: System architecture Mobile Digital Companion.

Around the Octopus switch there are several *Functional Modules* (like network module and video module). Inside the Octopus switch we have the *Module Interface Controllers* and the *Octopus switching fabric*.

In the communication we have three basic protocol layers: the *module layer* that connects the functional modules, the module interface layer, that interconnects the Module Interface Controllers, and the *physical layer* that is performed by the Octopus switching fabric. Figure 2 gives a schematic overview of the interconnection protocol layers.

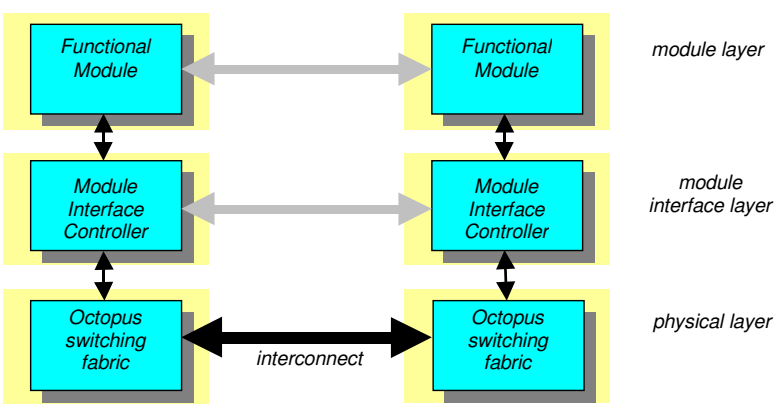


Figure 2: Interconnection protocol layers.

The grey arrows represent a connection at protocol level, and the black arrows represent physical connections. Flow control is done end-to-end rather than link-by-link, that is,

only the modules that send data and not the switch is responsible for controlling the flow of data.

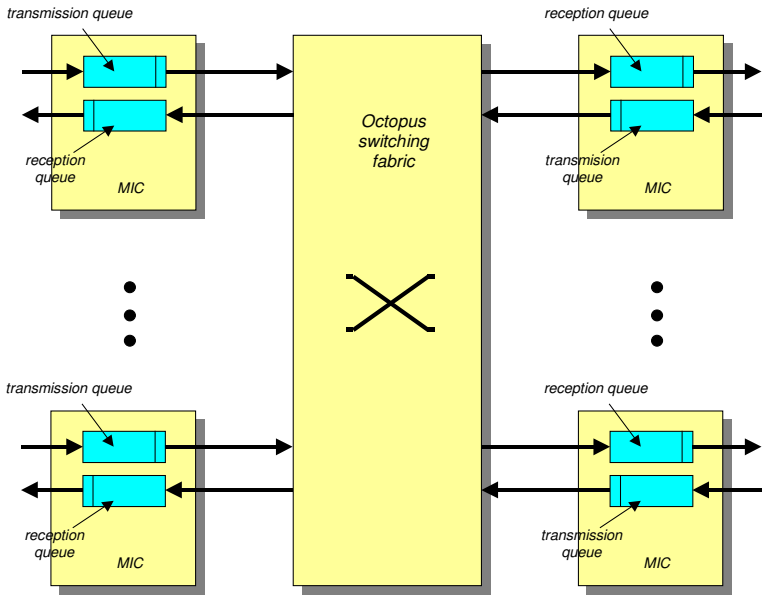


Figure 3: Basic architecture Octopus switch.

A high level schematic of the Octopus architecture is shown in Figure 3. At the heart of this architecture is the *Octopus switching fabric*. The fabric connects eight *Module Interface Controllers* (MIC) that interface to the attached modules. These MICs decouple the modules from the timing of the Octopus switch and the other modules. Each module can communicate with the others, so the switch is an 8-by-8 switch. The MICs contain small transmission and reception FIFOs that store ATM cells. The MICs further perform operations like connection setup and arbitration for the connections between the modules.

4.2.2 Packet size

The amount of data that is transported within a single time slot between functional modules is called a *packet*. Several factors determine a suitable packet size for the communication network.

- Firstly, as we want to minimise the buffering in the system, the packet size should be chosen as small as possible. However, the locality of reference principle suggests that we should preserve a sufficient amount of data correlation between the data items, so that the packet size should be sufficiently large. If we would have a too small packet size, of say one byte, then data items of different streams are sent alternating over the same physical link, thereby removing almost all data

correlation. This usually results in a higher transition activity, which causes higher energy consumption.

- Secondly, the minimum packet size is further determined by the overhead in transmitting a packet over the network. This overhead includes connection setup, arbitration of the connection network, error control and possibly also flow control. A small overhead (i.e. fast execution of the internal arbitration and protocol) allows for a smaller packet size, but it more expensive in area and energy consumption than a slower arbitration and protocol.
- Another aspect that determines the packet size is the amount of bits that can be transported over the network in parallel. A fully serial network is efficient in terms of the amount of wiring needed, and can therefore simplify layout. In contrast, a fully parallel implementation is more costly in terms of wiring and complicated layout, but can be very fast.
- Finally, since in a mobile computer the external (wireless) network places an important and prominent role, the packet size over the external network also determines the internal packet size. If the internal packet size were a multiple of the packet size that is transported over the external network then the interfacing would be practical, simple and efficient.

Asynchronous Transfer Mode (ATM) is used in communication systems. The ATM scheme is an advanced version of packet switching: small fixed-size cells are switched at high speeds. We will now only describe briefly the packet organisation of B-ISDN ATM, for a more comprehensive study on ATM we refer to [13]. A cell consists of an information field of 48 bytes and a 5 byte header. The primary function of the header is to identify cells that belong to a connection called the virtual channel, identified by a Virtual Channel Identifier (VCI). The VCI is used for routing ATM cells in a switching fabric. In addition ATM supports virtual paths, identified by Virtual Path Identifiers (VPI). A virtual path is a group of virtual channels. Further information contained in the header is the Payload Type, Cell Loss Priority, and a Header Error Control. The information field is transported transparently by the ATM layer, without any error detection or correction.

In the architecture we have chosen to adopt the structure of an ATM cell, i.e. a cell header and a small payload. This format has shown to be sensible for several reasons, among a practical one is that it allows for a simple connection to the ATM network that we are using. The size is small enough to allow for a fast and flexible scheduling of communication streams, and small enough to have a relatively small overhead. Within the system, the subdivision between virtual paths and virtual channels is not significant, and so the term circuit identifier is used to mean the whole of the header. The 5-byte header seems a bit overdone, but a practical implementation can easily implement a simple header compression mechanism that uses just a one-byte VCI header. A one-byte VCI header allows for 256 simultaneous virtual connections, which we expect to be large enough for our applications and system in a portable computer. Another format of the cell might be more efficient, e.g. a payload of 64 bytes and a header of 1 byte will

result in less overhead. The current testbed uses the B-ISDN ATM cell format, but future implementations might use a more efficient cell structure.

4.2.3 Buffer organisation

A problem that can occur on any communication network is blocking. Blocking results from sharing common resources such as common links or buffers. An important aspect of the architecture of a communication system is therefore the buffer organisation. To obtain a non-blocking communication network, sufficient bandwidth should be available on each communication path through the connection network.

A number of techniques can be applied to decrease the impact of blocking and to increase the throughput [15].

- First the system can provide an adequate buffering at the source, this is called *input buffering*.
- Secondly, it can try to buffer the traffic at the destination module, this is called *output buffering*.
- Finally, buffering can be provided *inside the Octopus switch*, using input buffering, output buffering, a congestion control mechanism, or a combination of these.

It is well known that output buffering yields much better performance than input buffering since cells can only be delayed when the bandwidth of the output link is saturated and never due to internal contention. Simulations showed that pure input buffering has a throughput of less than 60% compared to pure output buffering under a uniform workload [12]. This inferior performance is mainly due to head-of-line (HOL) blocking. HOL blocking occurs with FIFO queuing when a message at the head of an input queue is blocked, all messages behind it in the queue are prevented from being transmitted, even when the output link they need is idle. Link utilisation of an input buffered switch can be improved if the buffered cells can be randomly accessed rather than in FIFO order. This approach, however, requires a more complex buffer management and scheduling of the switching fabric [15].

When comparing the cost of an implementation, output buffering is more expensive than input buffering, in particular for large switches. For an output buffered n by n switch, the bandwidth b of the switching fabric and attached buffer memories grows as $O(n^2)$ since every output queue must be able to simultaneously receive data from n input ports. In contrast, the bandwidth of the fabric and the buffer memories for input buffering grows as $O(n)$ since an output port can only receive data from one input queue at a time.

The disadvantage of output buffering is thus the cost of the memory and the interconnection bandwidth. To overcome the congestion that can occur with input buffering, a scheduler can use a *congestion control mechanism* that (statically and/or dynamically) schedules the traffic in the switch.

Minimal buffering – The amount of buffering required in the system depends strongly on the traffic characteristics of a connection. It is, however, not always easy to know on beforehand the exact traffic characteristics. For example, video and audio data streams

are often bursty. Therefore the system must be able absorb large bursts and buffer the data during the scheduling latency interval. In the design of the Octopus architecture we have tried to minimise the amount of buffering that is required. Having no, or minimal buffering, can result in several advantages for both performance and energy consumption:

- *High performance* – Because the data is copied with no or just minimal buffering from the source module directly to the destination in the sink-module, the latency is reduced.
- *Energy efficient* – There is less energy needed for the storage of the data, and, if buffering can be omitted, also for the transfer to and from the buffers,
- *Simple flow control* – If buffering is used, then, in order to prevent overflow of the buffers, these buffers must be either very large or there must be a good and fast data flow mechanism. Large buffers require extra area and energy, which is wasted because most of the time they will not be used. Flow control induces also extra energy consumption and extra area because it requires extra communication and makes the design more complex.
- *Predictable QoS* – Having a large buffer also influences the Quality of Service because it increases the latency and jitter of the data packets between the modules.

In our model, the buffering of the data should thus be avoided as much as possible. Reduction in the required amount of buffering on the mobile can be achieved in several ways. Firstly, the applications that source the traffic can try to adapt the traffic rate to the rate the sink module can handle. For example in the case of a communication stream between a video camera and a display, the camera could lower its frame rate or use a smaller picture size. Secondly, the modules could try to adapt their implementation such that the system can handle the traffic rate. For example, the video source could use another coding mechanism. Thirdly, if the communication stream is between a mobile and an application or service that is running on a system with plenty of energy (in general on a wired network), then the required buffering could be migrated from the mobile to the fixed station. This implies that the data must be absorbed and processed as quickly as possible. Note that the same techniques can be used if the communication bandwidth is a bottleneck.

Octopus buffer organisation – In our model, the interconnection network is transparent and provides only a direct connection between two functional modules. The Octopus switch transmit and receive ports are simple, containing only minimal buffering and arbitration functionality. The buffering allows ATM cells to be read and written at a rate, which is independent of the functional modules. Each of the transmit and receive ports operates independently. This allows fast and slow modules alike to have a simple interface and see the interconnection network as a place to write and read ATM cells. Logically, the implementation of the Octopus switch represents a crossbar switch with both input and output buffering.

A buffering system can not be generic for all modules. Each individual module should use the buffering system that is dedicated to the traffic characteristics of that module. If the Octopus switch should provide an adequate buffering system and data-flow

mechanism that is capable of handling the buffering requirements of the modules, then the buffering system would become static, and designed for worst-case traffic characteristics that hardly occurs. It would need a large buffer capacity in the switch and an adequate flow control to prevent overflow of the buffers. However, in general most modules do not need such buffering system, and the introduced extra complexity and area requirements would be wasted.

In the Octopus architecture buffering can be performed at three logical units: the switching fabric, the MIC, and the modules. The most significant buffering is located in the functional modules, and with just a minimal buffering in the Module Interface Controllers. The switching fabric does not contain buffers.

- *Octopus switching fabric buffering* – We have omitted buffering of ATM cells in the switching fabric for the reasons mentioned before. Buffering in the fabric is only needed for synchronisation of a connection stream between two MICs.
- *Module Interface Controllers buffering* – These units provide a minimal amount of buffering, just enough to decouple the timing of the attached modules from each other and to buffer the data during the scheduling latency interval. Therefore, the rate at which the functional modules can handle traffic must thus be lower than the capacity of the MICs and the interconnection network.
- *Functional modules buffering* – This is the place where the most significant buffering can occur. Because the raw interconnect data rate of the Octopus switch is much higher than the rate of which the modules will probably handle the traffic, the main bottleneck will thus be not in the switch, but at the end-points of a connection between the modules. The responsibility of the buffer organisation is thus transferred to the module.

In some cases, like in traffic that uses the wireless communication network interface, buffering can be advantageous and reduce the energy consumption. In such cases the energy consumption required to buffer the data is lower than the energy savings that are achieved. However, this is only possible up to a certain limit due to buffer space limitations and Quality of Service requirements because for example the latency is increased [7].

4.2.4 Octopus switching fabric architecture

The Octopus switching fabric behaves like an 8-by-8 ATM switch. The switching provides a simple mechanism for the exchange of cells, regardless of their payload. The Octopus switch simply routes the traffic according to (a part of) the Virtual Channel Identifier (VCI) in the header. In contrast to full-blown ATM switching fabrics, the responsibility for ATM functions, such as VCI mapping and flow control, has been teased out of the switch fabric and assigned to the devices that plug into the switching fabric. The MICs are responsible for translating the VCI to the address of the destination module, and – when a connection has to be established – initialises the switching fabric with that address.

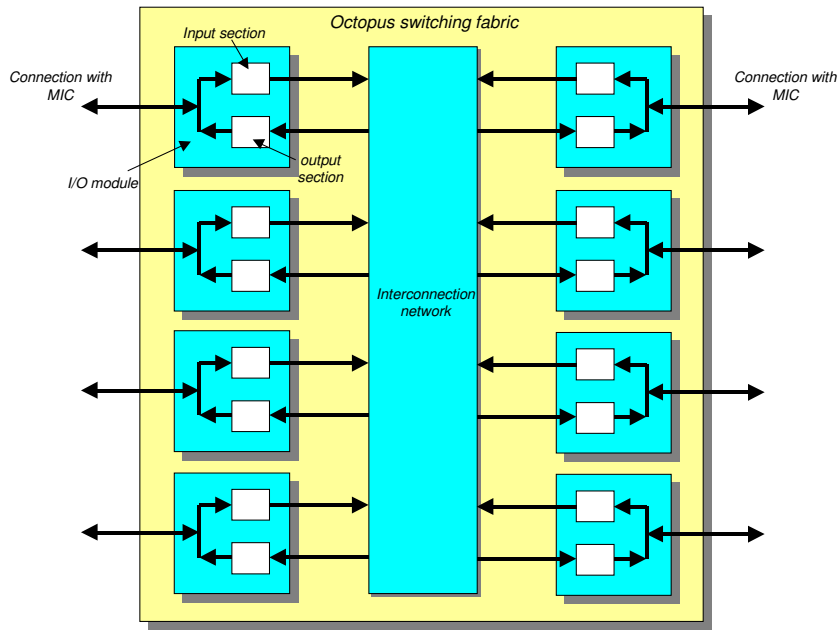


Figure 4: Architecture of the Octopus switching fabric.

The architecture of the switching fabric is therefore basically simple; most of the complexities are migrated to the Module Interface Controllers. As depicted in Figure 4, a switch consists of the following units:

- 8 input sections,
- 8 output sections, and
- an 8 x 8 interconnection structure.

A MIC is connected to an I/O module consisting of an input section and an output section. The connection between the MIC and an input and output section is shared, so the Octopus switch can support half-duplex connections only.

The interconnection network can be implemented in several ways as long as its capacity is large enough to support at least the maximum of four simultaneous connections at a rate that exceeds the data rate of an external module. In our prototype we use a fully connected crossbar. This is an energy efficient, simple and high performance architecture suitable for implementation in a Xilinx gate array. Although such a crossbar needs a large interconnection structure, the data rate on the individual connections can be relatively low.

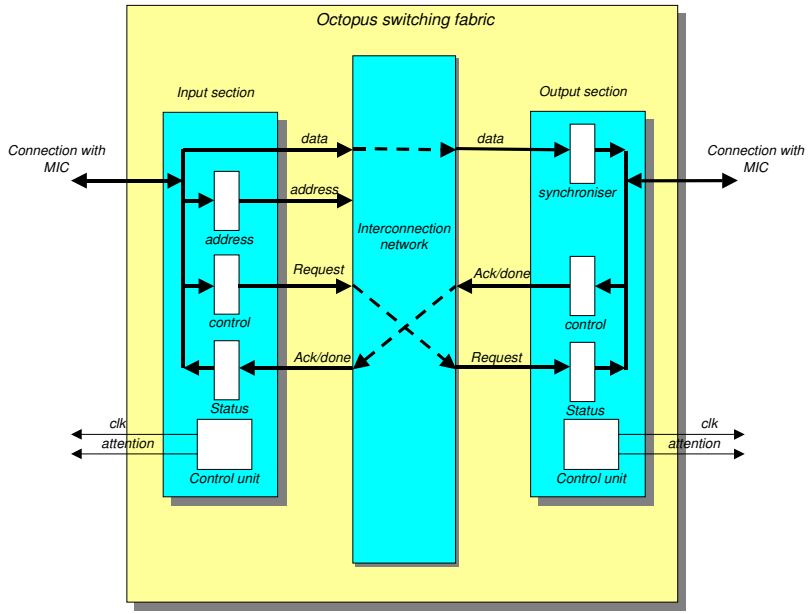


Figure 5: Structure of one input and one output section of the Octopus switch.

The input section basically only consists of three registers: an *address register* that determines the output section of the connection, a *control register* for energy management and general control, and a *status register*. The dataflow of a connection passes the input section transparently. The control register determines when a request for a connection will be made. The status register will contain status information about the current connection.

The output section contains two registers (*control register* and *status register*) and a *synchroniser* that synchronises the datastream to the timing of the receiver MIC. The status register is in fact shared with the input section. All requests for an output section are stored in this register and can be read by the MIC.

Both the input and the output section of an I/O module share the *control unit*. This unit generates the *clock signal* for the attached MIC, and generates an *attention signal* when the MIC has to do an operation. The attention signal can be used to wake-up the MIC from sleep mode.

Figure 5 shows one input section and one output section that are involved in one connection between two modules. The interconnection network uses the address that is stored in the address register of the input section to select the output section it wants to make a connection to.

4.2.5 Module Interface Controller architecture

The Module Interface Controller operates at the module interface layer. Its main task is to provide a data-path between the modules. The MIC basically contains the following units:

- A *transmission queue* that stores ATM cells originating from the module in transit to the switching fabric.
- A *reception queue* receives the ATM cells coming from the switching fabric
- A *VCI mapping table* determines the destination module, and is indexed by the VCI in the header of the ATM cell.
- An *arbiter* performs the actual establishment of a connection and performs scheduling in case multiple simultaneous requests are received via the switch.

Figure 6 shows the basic architecture of a Module Interface Controller with these basic units.

We will briefly describe the basic functions of these units using a typical communication stream between two MICs.

When a MIC receives an ATM cell from the module it is attached to, it will first store this ATM cell in its *transmission queue*. The output port to which the cell must be forwarded is determined by looking up in the VCI mapping table. The VCI mapping table contains the destination MIC of all previously announced virtual connections. The connection identifier contained in the VCI header of the ATM cell indexes the VCI mapping table. The MIC will then establish a connection with this destination MIC. Cells with a VCI that is not known will be forwarded to a default module, which in general will be the CPU-module. The CPU-module contains the *connection manager* (*ConMng*) that is responsible for the management of all connections in the system. The ConMng uses special *management cells* to communicate with the MICs and the modules in the system. The VCI mapping table will be initialised by the ConMng using these management cells.

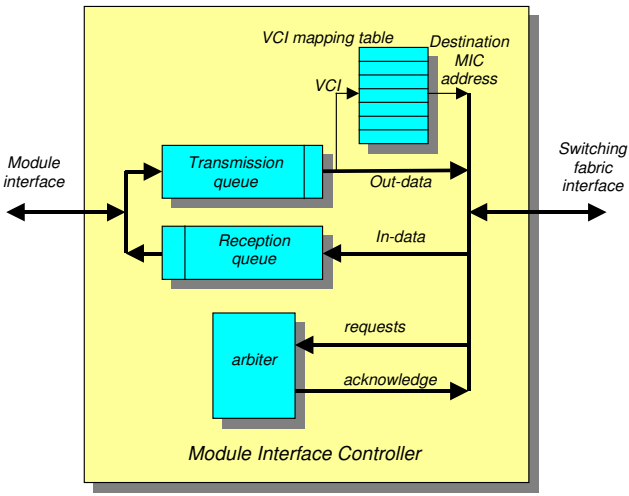


Figure 6: Module Interface Controller architecture.

When the destination MIC receives a request for a connection, the *arbiter* determines when and whether the connection can be established. If there are multiple simultaneous connection requests, it uses a scheduling algorithm to determine which request can be honoured first.

When the connection is established the source MIC forwards the data from its transmission queue over the Octopus switching fabric to the destination MIC. The destination MIC then first stores the ATM cells in its *reception queue* before it is further forwarded to the attached module. Note that the buffering of cells in the transmission and reception queue is not always required: when the module is capable to handle the cells at the required rate, then the buffering can be omitted.

4.2.6 Connections

All communication between modules are based on connections. There is a central Connection Manager (ConMng) that can be used to schedule traffic between the modules. All connections are uni-directional.

Connection types

There are basically two types of connections: *ad-hoc connections* that have no reservations made in the system, and *guaranteed connections* in which the flow through the switch is guaranteed by the connection manager (ConMng) located in the main CPU-module.

- *Ad-hoc connections* – Modules that need no guaranteed flow of data use the ad-hoc connections. These type of connections have no real-time guarantees, and a higher protocol is needed e.g. for flow control. Ad-hoc connections are also used during

the connection setup phase in which the module establishes a guaranteed connection with a different module.

- *Guaranteed connections* – Guaranteed connections are used to transfer data between modules that require bandwidth guarantees between the modules. Once a guaranteed connection is established, the actual data transfer still has to be announced by the source. In this way, the destination MIC can determine whether the reserved bandwidth is actually used, and otherwise assign that bandwidth to other connections.

Since the Octopus switch does not provide flow control between the modules, the modules either have to use some flow control mechanism (*end-to-end flow control*), or must be capable to source and sink the data at minimal the rate they both agreed upon. This is in line with our philosophy to have minimal buffering.

There is no flow control mechanism in the interconnection layer between two functional modules. Flow control mechanisms are generally used to control the stream of data between buffers. Our flow control can be much simpler than the ones required in for example ATM networks. In these systems, flow control is normally performed on a link-by-link basis. For guaranteed connections we do not need to have a flow control at the switch layer. End-to-end flow control suffices because

- guaranteed connections have been assigned a certain amount of bandwidth in the Octopus switch. The arbiter in the Octopus switch uses a distributed arbitration mechanism in which ad-hoc connections have a lower priority than guaranteed connections,
- the modules can be trusted in the sense that they never exceed their bandwidth allocation,
- the interconnect diameter (which is the maximum distance over all the nodes of a network) is small (internal in the system just one hop), and
- the Octopus interconnect provides a large bandwidth suitable for many simultaneous communication streams (the testbed it can support three simultaneous connections of 32 Mb/s, see Section 4.3.3).

All these reasons make that congestion internally in the switching fabric is not likely to occur.

Connection management

Prior to a communication transfer between two modules, a connection has to be set up. Individual MICs in the Octopus switch can be remotely configured by using special control cells. The operating system running on the CPU-module issues these control-cells. Once a connection has been setup, control can be taken over by a local processor that is responsible for controlling both the MIC as well as the locally connected device.

During *connection setup*, the source module contacts the *connection manager* (*ConMng*) that is located in the CPU-module that it requires a channel to a module with a certain amount of bandwidth. It therefore transmits an ATM cell containing the request to the

CPU module. Since no connection with this module has been established yet, it establishes an ad-hoc connection with the CPU-module. Since such a connection has no guaranteed bandwidth the connection request might take some time before it can be acknowledged. The CPU module will upon receiving the request determine whether there is enough bandwidth available between both modules involved and the connection can thus be established. The ConMng then might negotiate with the destination module to verify that this module is capable and prepared to connect. If the connection is not possible (because either the ConMng knows that there is not enough bandwidth in the Octopus switch, or the destination module cannot accept the connection), then the ConMng will reply to the requesting module that the connection is currently not possible. If the connection is possible, then it will inform the destination module and the source module that it a new connection has been set-up.

The aggregate bandwidth available inside the Octopus switch allows each module to communicate at a rate that is probably much higher than it can source or sink. The Octopus switch allows up to four parallel connections between eight modules. Given the small number of modules and the rate at which our present modules are able to generate traffic, this provides enough bandwidth.

4.2.7 Scheduling

In principle each MIC needs to be able to buffer just two ATM cells, both for receive traffic as for transmit traffic. The main philosophy is that the Octopus switch buffers as little as possible, and that the modules either have to provide the buffer capacity, or that they can adapt their traffic rate.

The input ports of the MICs can receive multiple simultaneous requests from all other MICs. A scheduling policy has to be applied to choose which input port will be acknowledged, and can forward an ATM cell to the module. The problem of scheduling in switches has been studied extensively in the design of ATM switching fabrics. Although the design of our internal ATM switch is much more limited, many of the peculiarities in full-blown ATM fabrics apply to the Octopus switch.

Basically there are two types of scheduling: static and dynamic scheduling [15].

- In *static scheduling* the bandwidth allocation is made on the basis of time slots in a service cycle. A *service cycle* is a periodically recurring time interval consisting of s time slots. In static scheduling each time a new request for a connection is made, the scheduler tries to allocate bandwidth on the path between source and destination by reserving resources for a number of timeslots. A *communication scheduling table* describes which connections can communicate at each time slot. When a new reservation is made, the scheduling table has to be updated, and possibly rearrange the previous schedules. The Slepian-Duguid theorem [11] states that a schedule can be found for any traffic pattern, as long as the number of cells for any input or any output is no more than the number of time slots in a service cycle. Computing the new schedule may require the number of slots in a service cycle s times switch size n (thus $s \cdot n$) number of steps for an n by n switch [2][15]. The main advantage of static scheduling is that it has a bounded latency and can therefore be used for real-

time multimedia traffic. Disadvantages are that it requires a centralised scheduler, and that for each new connection, the scheduling table has to be adapted. Furthermore, some bandwidth can be wasted when a reserved slot is not used. This, however, can be overcome with some special precautions.

- In *dynamic scheduling* the arbitration is more dynamic, and uses a scheduling mechanism to choose which cell to forward on the output link. Common arbitration schemes, such as round robin scheduling or priority based scheduling, are suitable. Which arbitration scheme is best for a specific set of applications depends on the characteristics of the tasks and on the cost of implementation of the arbitration scheme. Examples of scheduling mechanisms for switches are proposals of Hui [11] and parallel iterative matching [2]. A problem associated with dynamic scheduling is fairness because the scheduling is distributed at all the output ports. Dynamic scheduling is thus not really suitable for real-time traffic.

In the prototype of the Octopus switch we use a combination of both scheduling techniques: *static scheduling* for guaranteed connections, and *dynamic scheduling* for ad-hoc connections. A certain portion of the bandwidth is allocated to traffic of guaranteed connections, and any unused slot can be used for other traffic.

Static scheduling – The bandwidth allocation for the guaranteed connections is made on the basis of time slots in a service cycle. Each time a new guaranteed connection is required, the source MIC issues a connection request to the Connection Manager. The Connection Manager tries to allocate bandwidth on the path between source and destination by reserving resources for a number of timeslots. Given the communication network and a collection of connections, the question is in which time slots a certain connection may use the network, such that its throughput requirements are satisfied and no conflicts with other connections can occur. The *communication scheduling table* that describes which source MIC can communicate at each time slot is distributed to all MICs in the system using a management cell. Figure 7 shows an example of a communication scheduling table. The rows of the table represent the *s* time slots of a service cycle. Furthermore, the columns defines the source MIC, and each cell in the table defines the destination MIC. Cells that are not assigned a destination MIC represent time slots that can be used for ad-hoc connections.

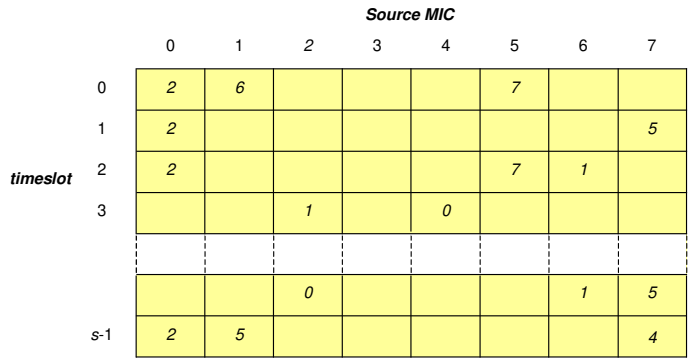


Figure 7: Communication scheduling table example.

Note that in the Octopus switch no distinction is made between connections from one MIC. A guaranteed connection request is issued by the source MIC as if it requires more time slots. When a MIC has multiple connections, it is free to use any slots that it is assigned to, even if they were originally not for that connection. In this way, the source MIC can easily adapt to small fluctuations in the connection streams.

When the guaranteed connection is established, the connection has reserved some bandwidth but these slots are not yet used for traffic. When the source has traffic on the guaranteed connection, it still needs to issue a request to the destination. The arbiter in the destination MIC will always honour this request. When the guaranteed connection has no traffic and does not use the slot, other connections may use the slot instead.

Dynamic scheduling – Dynamic scheduling is much more flexible and the traffic can easily be adapted to the current needs. In the prototype the scheduler of the arbiter in the MICs are based on a round-robin scheduling based on the source MIC (and thus not per connection). Fairness is not a big issue because most connections will be reserved in advance using the static scheduling. These guaranteed connections have a high priority. Ad-hoc connections, however, have a lower priority. We use two separate schedulers for both types of connections. Slots that are reserved for a guaranteed connection, but that are not used at some moment, can be used for other connections, both guaranteed connections as ad-hoc connections. In this way no bandwidth is wasted. Only when no requests are made for traffic on a guaranteed connection, the scheduler for the ad-hoc connections becomes active.

Although ad-hoc connections have no strict timing requirements, some bandwidth needs to be reserved in order to allow new connections to be established and also to allow progress in the processing of these connections. The Connection Manager can satisfy this requirement by ensuring that over a specified time interval sufficient time slots are available for ad-hoc connections. This time interval may be relatively large (e.g. span a large number of service cycles) because there will be sufficient time slots that remain unused. We can safely say this because the aggregate bandwidth is designed to be sufficient, and guaranteed connections are not always active since the bandwidth required by real-time connections may vary over time. During the time that no

bandwidth is required by such a connection, bandwidth can be used by other connections.

Note that traffic between two modules may consist of both a guaranteed connection, as an ad-hoc connection. For example, a video connection requires a guaranteed throughput to maintain some QoS, and more bandwidth only to improve the quality for which it uses the ad-hoc connection.

4.2.8 The stages of the internal communication protocol

During the operation of the switch, several phases are executed as an ATM cell is transmitted from one module to another. We identify the following phases: wake-up phase, arbitration phase, data phase, and release phase. Note that the basic transfer size is one ATM cell, but several cells can be grouped in a frame.

In the following figures the units and signals that are not used at a certain stage during the protocol are not shown. Units that are idle are coloured white, and fifos that are empty are also white.

Sleep phase – in the sleep phase most units are in a low-power mode, receiving no clock. Only a small part of the Octopus switch is active, waiting for an external event from a functional module indicating that it needs to communicate.

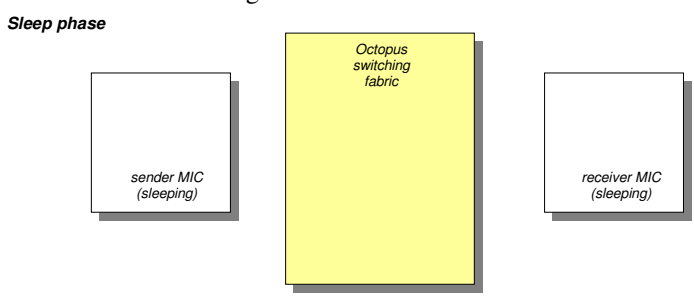


Figure 8: Sleep phase.

Wake-up phase – the first operational phase is the wake-up phase. Normally, i.e. when the Module Interface Controller has no communication to handle, the MIC is sleeping and does not receive a clock from the Octopus switch. If a module needs to transmit data to another module, it notifies the Octopus switch that in its turn wakes the Module Interface Controller by generating a wake-up event. The Octopus switch turns on the clock that goes to the MIC and gives an *Attention (att)* signal. The MIC can then receive an ATM cell from the module. Prior to the actual transfer, the destination MIC has to be notified that it has traffic waiting. It uses the VCI of the ATM cell to determine the destination module, and sends a *Request* to the receiver-MIC of the destination module. The output port of the Octopus switch that is connected to the receiver MIC contains a *Status register* that collects all outstanding requests (maximal 7). If the receiver-MIC is sleeping, the Octopus switch turns the clock on, and wakes the receiver-MIC by sending an *Attention* signal. Now both MICs that are involved in the communication are awake and the arbitration phase is entered.

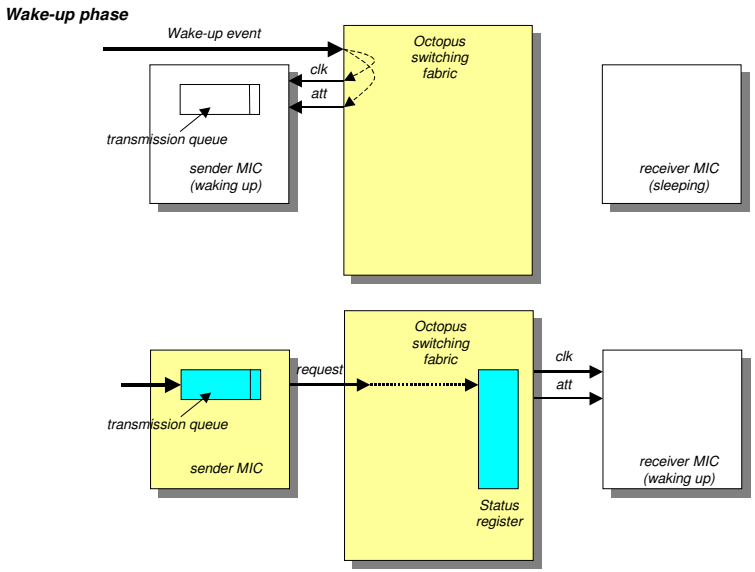


Figure 9: Wake-up phase.

Arbitration phase – The receiver MIC is signalled via the *Attention* signal that it needs to respond to a request. It reads the *Status register* that contains the outstanding requests to its module. It uses a scheduling mechanism to arbitrate between possible multiple requests and replies an acknowledge to the sender-MIC it has selected. Figure 10 shows the arbitration phase in which two simultaneous requests are received by the receiver MIC. The arbiter determines which sender-MIC will be granted the connection, and sends an acknowledge to that sender-MIC. The request of the other sender-MIC will not be honoured yet and has to wait.

The acknowledge is received in the Status register of the sender-MIC, and the MIC receives an attention signal that it should read the register. The attention signal allows the sender-MIC to enter sleep mode as soon as it has made a request. The attention signal will wake-up the MIC when it receives the acknowledge. This mechanism can save energy, since – especially for ad-hoc connections – it can take some time before the connection is established. The MIC reads the status register and can enter the data-transfer phase.

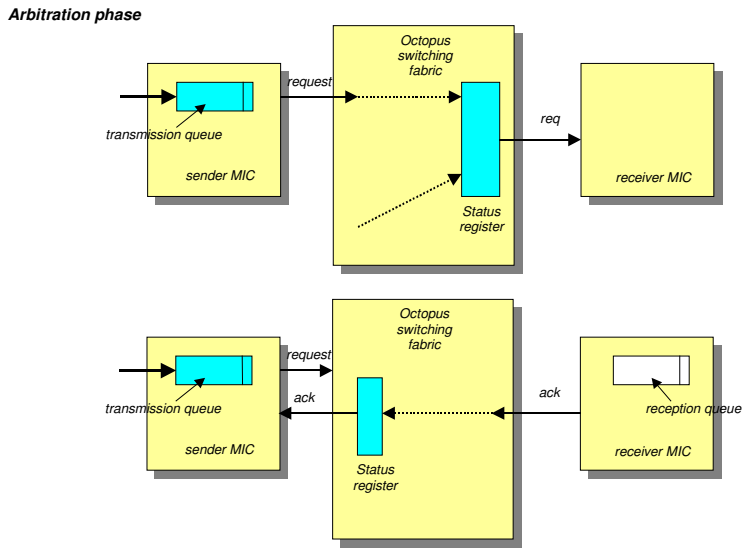


Figure 10: Arbitration phase.

Data-transfer phase – In this phase the transmission queue of the sender-MIC is read and transferred via the switching fabric to the reception queue of the receiver-MIC. The receiver module is notified that it should read the reception queue of its MIC.

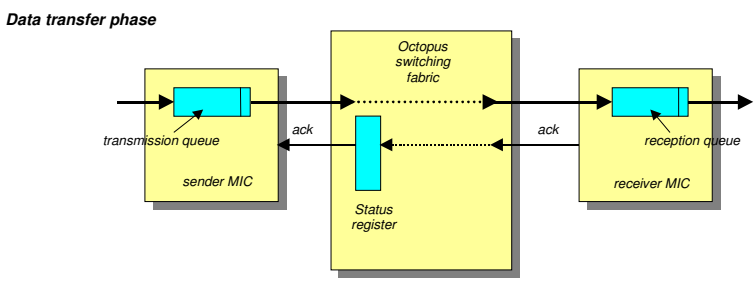


Figure 11: Data-transfer phase.

The acknowledge signal remains active during the communication. When the data-transfer is completed (which in general shall be after one ATM cell), then the release phase is entered.

Release phase – When all bytes of the ATM cell are transmitted, the source MIC releases its request, to signal to the destination MIC that it is ready. If the transfer was successful, i.e. the correct number of bytes were received, the destination MIC returns a *done* signal (i.e. it releases its acknowledge). This is the only error detection performed in the switch since we expect very little errors, and want to keep the forwarding delay as short as possible. This is mainly due to our software implementation of the data-flow. If we would have implemented the data-flow in the MIC with a hardware engine, then we

could more easily implement a better error detection, or even an error correction mechanism.

If the sender module has no more data to send, the sender-module can go to sleep mode.

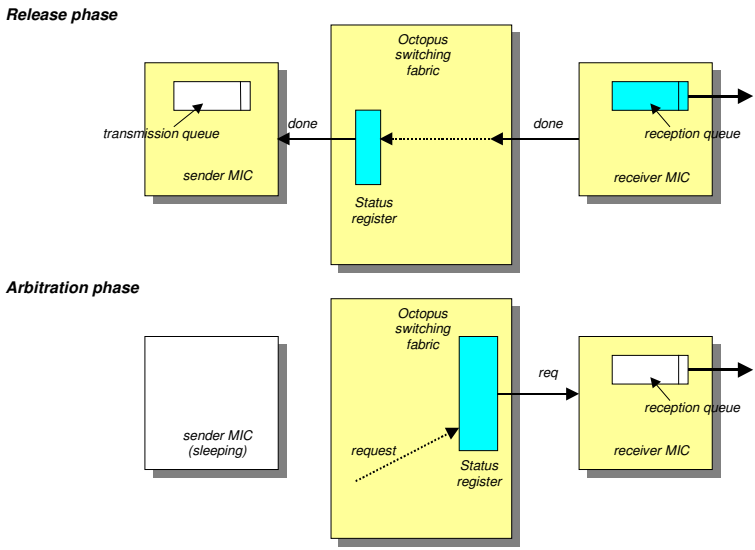


Figure 12: Release phase (and subsequent arbitration phase).

The receiver MIC can now enter the arbitration phase again. Figure 12 shows such a situation. The receiver-MIC schedules again and establishes a connection for the other still outstanding connection request.

4.2.9 Clock Gating

Chapter 3 already showed the locally synchronous, globally asynchronous timing methodology that was used in our architecture. The Octopus switch internally uses a synchronous design methodology. However, the local clock of various building blocks is generated under the control of incoming data tokens. Arrival of a data token restarts the clock signal, and when the processing is done, the clock signal is disabled. Such a mechanism is also know as *clock gating*, which is the most popular method for power reduction of clock signals [17]. When the clock signal of a hardware unit (which in general can be ALUs, memories, state machines, etc.) is not required for some period, the clock feeding the unit can be turned off. The generation of the enable signals increases the complexity of the circuitry, and the timing relation of the signals has to be evaluated carefully to avoid signal glitches at the clock output. Note that the gating signal should be enabled and disabled at a much slower rate compared to the clock frequency. Otherwise the energy required to drive the enable signal may outweight the energy saving.

In the design of the Octopus switch clock gating has been applied at several layers and with different timing granularity. The trade off is to justify the additional hardware and design complexity in managing the various functional units.

Clock gating at the architecture level

At the architecture level, clock gating is a particularly attractive technique because little hardware and design complexity is needed to achieve substantial energy saving. At the system architecture level layer the Octopus interface controllers can be left idle and be sleeping for an extensive period of time. The Octopus switch controls the clock signals of its attached interface controllers. When there is no traffic flow in the system, then all interface controllers are sleeping and receive no clock. When, due to an external event, the Octopus switch notices that it requires an interface controller, it enables the clock signal to the interface controller, and wakes the controller from its sleep. The effectiveness of clock gating in our testbed is described in Section 4.3.3.

Note that, although the design complexity of this method is low, and the energy savings can be high, the startup time can be significant in a particular implementation. The PIC micro-controller that is used in the prototype implementation has several energy saving modes. The intent is to put more functional modes of the controller in idle when the processor goes into a deeper sleep mode. However, it requires more latency to resume computation from a deeper sleep mode. Therefore, the tolerable latency of the attached module determines the sleep mode of the interface controller. In general a module that interfaces at a high data rate requires also a low latency, because otherwise it would require substantial buffering. The PIC controller requires 1024 clock signals (which is equivalent to 51 μs on 20 MHz) to power up from its deepest sleep (internal clock oscillator also turned off) and less than one μs when the internal clock oscillator keeps running.

Clock gating at logic level

Clock gating at the logic level in the Octopus switch is used in various blocks and at different hierarchies. The Octopus is divided into eight basic and identical blocks. Each block interfaces one functional module of the system. All blocks are interconnected to each other via a fully connected crossbar switch. If the block is not needed to handle traffic, then the whole block is inactive, and receives no clock. When a block takes part in the communication stream between two functional modules, then the block becomes partially active. Only those parts of the block that are required during a certain stage in the communication protocol are active and receive clocks.

This principle is applied at both the control flow (for state-machines) and at the data-flow. The advantages for the state-machines are mainly local: no energy is wasted just to stay in the same state. The advantages when applying this principle in the data flow can have more impact. The data-flow in the Octopus switch is based on a pipelined and *guarded* architecture. The basic principle of a guarded architecture is to identify logical conditions at some inputs to a logic circuit that is invariant to the output. To reduce the switching activities inside the Octopus switch, latches (registers) are added into the data

flow that guard switching activities to propagate further inside the switch. The latches are transparent when the data is to be used. Otherwise, if the outputs of a unit are not used, then they do not change.

4.3 Implementation of the Octopus switch

The previous section described the architecture of the Octopus switch. In this section we will present our testbed implementation of the Octopus switch that is used as the interconnection network of the Mobile Digital Companion.

A key goal motivating the design has been simplicity and flexibility. Our goal was to build a testbed from off-the-shelf VLSI components that was easy to design and test. With this testbed we are able to quickly explore the design space of the system. Therefore, the prototype interconnection module is build using a Field Programmable Gate Array surrounded by several low-end and low-power micro-controllers.

4.3.1 Basic components of the testbed

A prototype of the Octopus switch has been implemented on a single small printed circuit board with a standard Field Programmable Gate Array of Xilinx (i.e. XC4010XL) and six low-end micro-controllers (i.e. Microchip PIC 16C66). So, in the testbed we are able to interconnect only six modules instead of eight.

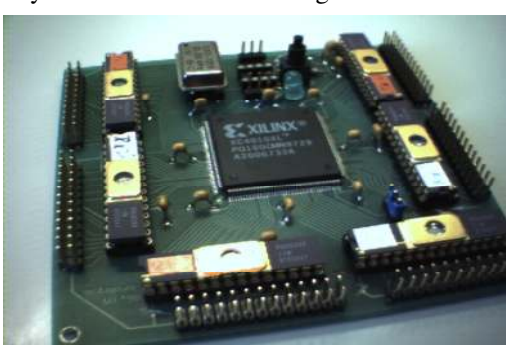


Figure 13: Testbed implementation *Octopus*.

Field Programmable Gate Arrays – Xilinx's FPGA architecture is similar to other gate arrays, with an interior matrix of configurable logic blocks and a surrounding ring of I/O interface blocks. A logic block consists of a combinatorial section and a sequential section. A logic block can also be configured as a small memory (32 by one bit), but this feature was not used in the prototype. The functions of the FPGA (logic, memory, and their interconnects) are stored in an on-chip memory. This technology allows gate arrays to be (re)programmed an unlimited number of times. With proper tools, the design cycle of implementing a design in an FPGA is short. In the last years traditional FPGAs have

gained new positions in the semiconductor industry moving from their initial use for 'glue logic' and fast prototyping purposes to be adopted for typical co-processor tasks. Still, speed area and power consumption are considerably lagging behind the more traditional ASIC designs.

The basic reason of the high energy consumption of an FPGA is due to the interconnect capability within an FPGA: the intrinsic need of routing flexibility implies that more than 90% of the total area is due to interconnect resources. The fact that general connection patterns are provided through switching points instead of dedicated wiring implies that the resulting routing is more resistive and capacitive. As a consequence the relative weight of interconnect contributes on average no less than 50%, and possibly as much as 80% to delay and power consumption.

The main reason for using FPGAs in our design is therefore *flexibility*. We use FPGAs as a dynamic programmable unit, whose function can be changed under program control. This approach creates a test-bed for interconnection structures, arbitration mechanisms, and clocking mechanisms.

Micro-controllers – With an FPGA it is relatively simple and efficient to implement the datapath of a system. However, the FPGA is not suitable for high control complexity. For example, it would require relatively large area if the FPGA has to implement the connection setup protocol, handle timeouts and perform retransmissions. Traditional microprocessors are much better equipped to deal with larger and more complex control structures, and although they are capable of handling the data-flow as well, they typically can only achieve this with a much lower performance than with a hardware implementation.

By combining an FPGA with a traditional processor, it is possible to build a system that uses a mixture of hardware and software in order to exploit the best features of both domains effectively. Another major advantage of using micro-controllers (at least of the type that we have used), is that they consume very little energy, and still provide a reasonable performance. The micro-controllers used have power savings modes in which they consume little energy (less than 1 μA typical standby current).

4.3.2 Implementation

The FPGA can be programmed to operate as the interconnection switch, that connects the modules. Each port of the switch is connected to one micro-controller, so in our prototype we have six micro-controllers. The basic function of the micro-controllers is to perform routing, to establish a connection, and to interface between the switch and the connected modules.

The micro-controllers implement the *Module Interface Controllers* (*MIC*) as described in the architecture. The datapath between the micro-controllers and the FPGA is eight bits wide. The internal datapath in the switch is also 8 bits wide. All data in the system is based on the size of an B-ISDN ATM cell (48 bytes, 5 bytes header). 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 and have a small latency, and large enough to keep the overhead small. Using an other frame format is quite well possible. E.g. a frame

format that uses 64 bytes of payload data with a one byte connection identifier would be more efficient, but would complicate the interface to a B-ISDN ATM network.

We have implemented the interconnection as a fully connected crossbar switch. The switch does not have ATM sized buffers, but just some synchronisation and pipeline registers.

In the current prototype the MICs perform several tasks:

- Connection establishment with the other MICs that are connected to the switch.
- Routing of traffic between the modules
- Scheduling of traffic at the output port of the switch destined for the module
- The actual data-transfer between the module's device and the input port of the switch

Note that the MICs also perform the actual data transfer. The reason for this is that they can perform the data transfer at a sufficient data-rate, and with very low energy consumption. We were able to achieve such a high data-rate because we added special synchronisation circuitry in the Octopus switch.

Connection synchroniser

The switch is capable of having three simultaneous data streams between three disjoint pairs of MICs. Since the data traffic to and from the switch is handled in software by the MICs, the data rate is determined by the rate at which the MICs can handle this. The switch should provide the connection between the two MICs involved in the connection. Both the sender-MIC as the receiver-MIC must participate in the communication at the same time. A complicating factor in the communication is that although the MICs operate at the same frequency and share the same clock signal, they operate completely asynchronously from each other and can be in another internal operating phase. For a reliable connection between the MICs we thus need a *handshake protocol*. However, to achieve the highest speed, it is not possible to have such a handshaking protocol that is implemented in software between the MICs. Therefore the Octopus switch contains some *synchronisation circuitry* at all output ports of the switch that synchronises the traffic between two modules.

When the actual traffic from the transmitting MIC arrives at the output port of the switch, the synchroniser can introduce some delay to adapt the timing of the traffic to the phase of the receiving MIC. The synchroniser therefore has to know the phase of both the sender-MIC and the receiver-MIC. When the receiving MIC acknowledges a connection request, the synchronisation circuitry knows the phase of the receiving MIC. Each ATM cell is preceded by a special *Start of Cell header*. The output port of the switch uses this header to determine when the sender starts transmitting an ATM cell. This header is also used as a synchronisation byte to determine the difference in phase between the sender MIC and receiving MIC and if needed introduces a little delay. In this way, the sender MIC can transfer the ATM cell that is in its transmit queue at the highest speed possible without having to care about any handshaking protocol. Figure 14

shows the datapath of a connection between two modules. The synchroniser is located at the output port of the switching fabric.

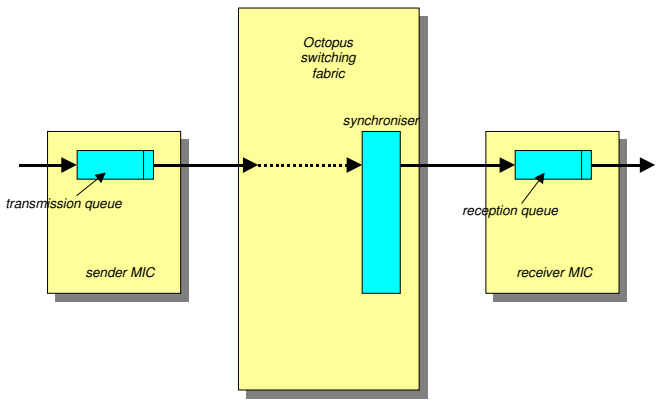


Figure 14: The datapath of a connection between two modules.

The MICs can thus receive and transmit cells at the highest speed possible. Figure 15 shows a simplified part of the code that is used to transmit an ATM cell in our testbed[2]. The ATM cells are stored in dedicated buffers (internal registers). During a communication between two MICs, they can perform no other tasks. The sender-MIC just has to read one byte from the buffer and then write it to the input port of the Octopus switching fabric. The receiver MIC does similar operations, but writes the data to a dedicated receive buffer. As can be seen, just two operations (or 8 clock ticks) are needed to transfer one byte. The achievable data-rate is thus equal to one bit per clock-tick.

[2] This code is only valid for our testbed, and it is shown here to give an indication of the complexity of the required functionality. When integrated in a chip-design, then the micro-controllers will be dedicated state-machines.

```
;*****************************************************************
Tx1Cell
; transmits an ATM cell from buffer 1 to receiver DEST
;*****************************************************************
       movf           DEST,w          ; DEST: the destination MIC
       call           sndReq          ; generate a request
       call           wait4ack        ; (busy) wait for the acknowledge
       ; now the acknowledge has been received start sending the cell
       bsf            PortA,0         ; command Write Data
       set_Octopus_O                  ; set Octopus port (B) to output
       movlw          B'10101010'     ; write sync
       movwf          PortB           ; to Octopus port
       nop
       movf           ATMb1,w         ; read 1st byte from buffer
       movwf          PortB           ; write to Octopus port
       movf           ATMb0,w         ; read 2nd byte from buffer
       movwf          PortB           ; write to Octopus port
       . . . .                        ; more bytes . . .
       movf           ATMb53,w        ; read last byte from buffer
       movwf          PortB           ; write to Octopus port
       set_Octopus_I                  ; set Octopus port to Input
       bcf            PortA,0         ; command Read Status
       call           Release         ; release the connection
       return
```

Figure 15: Code fragment to transmit one ATM cell.

The design methodology used is data driven and based on asynchronous methods, combined with synchronous parts. Each individual part of the switch that is not used at some time, does not receive any data and clock changes, thus minimising energy consumption. The attached modules can be similar to devices found on today's PDAs or notebook computers, but can also include multimedia devices like a camera. If the functionality of the attached module is small and requires little resources, then parts of this functionality can be integrated in the micro-controller as well.

The design has been implemented and tested. The design allows us to do experiment with and make performance measurements of various architectures and interconnection protocols. We used VHDL as a design tool.

4.3.3 Performance

We have measured the performance and energy consumption of the Octopus switch (implemented in an FPGA) including the MICs (implemented in six micro-controllers). All measurements are performed with a clock frequency applied to the switch and MICs ranging from 0.1 to 32 MHz. This is equivalent to a raw data-rate per connection of 0.1 to 32 Mb/s. The Octopus switch is capable to support up to three simultaneous active connections when the connections are disjoint. This makes the total maximal throughput to be 96 Mb/s. This data-rate is more than sufficient to support all our expected data

streams on the mobile computer. A full-custom implementation in an Integrated Circuit will give a higher throughput.

Performance and energy consumption during data transfer

In Figure 16 we have plotted the power consumption for varying frequencies, and with a different number of simultaneous data-flows. In this setup at most three disjoint connections were active, involving six modules, three sending and three receiving MICs. Since the connections are disjoint, no congestion can occur and all units are able to operate at maximum speed.

The graphs show clearly that the energy consumption increases linearly with the frequency, which was expected. It also shows that the required amount of energy depends strongly on the number of data flows in the switch. This effect is mainly due to our aggressive power management. All parts of the system that at some moment have no functionality, are in a low-power mode. The micro-controllers are in a very low-power mode when they don't have traffic, and their contribution to the energy consumption can be ignored. The switch, however, always has a large energy overhead due to the implementation in an FPGA.

Although most parts of the switch idle, it still requires a significant amount of energy. The parts of the switch that remain active are the clock generation and the wake-up circuitry.

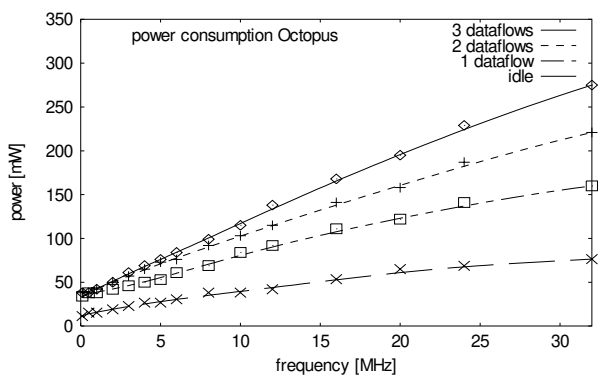


Figure 16: Power consumption Octopus switch with various simultaneous data-streams.

In these measurements the micro-controllers are actively transferring data from their modules to the switch. There are three transmitting modules, and three receiving modules. The transmitting micro-controllers operate in the following sequential phases:

phase 1: reading an ATM cell from the module (*module I/O phase*),

phase 2: establishing a connection with a destination module (*arbitration phase*),

phase 3: transferring the cell to the switch (*data-transfer phase*),

phase 4: waiting for an acknowledge and releasing the connection (*release phase*).

The receiving micro-controllers have three related phases:

Phase 1: arbitration and connection establishment with the source module,

Phase 2: receiving a cell from the switch (*data-transfer phase*),

Phase 3: sending an acknowledge and releasing the connection (*release phase*),

Phase 4: writing the received cell to the module (*module I/O phase*).

Note that the phases of both micro-controllers that take part in the connection operate in parallel, and that the effect is that there is a pipelined dataflow between the modules. During phase 1 of the transmitting module (i.e. an ATM cell is transferred from the module to the micro-controller), the receiving module is in phase 4 and transferes the just received ATM cell to its attached module. The synchronisation point is when a connection is being established.

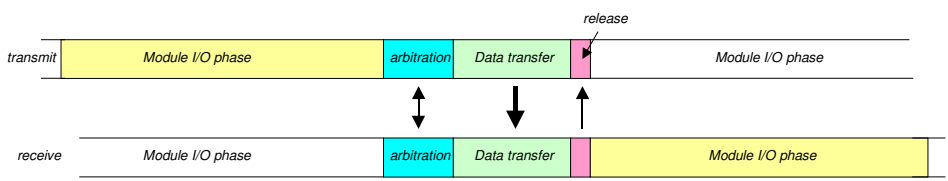


Figure 17: Phases during switch communication.

We have also measured the time in which the individual phases contributed to the total time needed to transfer one ATM cell. In the setup the arbitration phase took 9% of the time, the data-transfer between the micro-controller and the switch, including the release phase, took 27%, and the transfer between module and micro-controller took 64%. The contributions to the time of the receiving and transmitting controller were thus irrespective to their function, and only depended on the phase. Note that the interface with the switch is highly optimised, and that the interface with the module is more general, and requires more time. To determine the effect of the data-rate with the module to the energy consumption, we have made another setup. In this setup the data-rate with the module was infinite (in fact we removed the module I/O phase (1) from the transmitting controller, and the module I/O phase (4) from the receving module). Figure 18 show the result of these measurements with three simultaneous traffic flows.

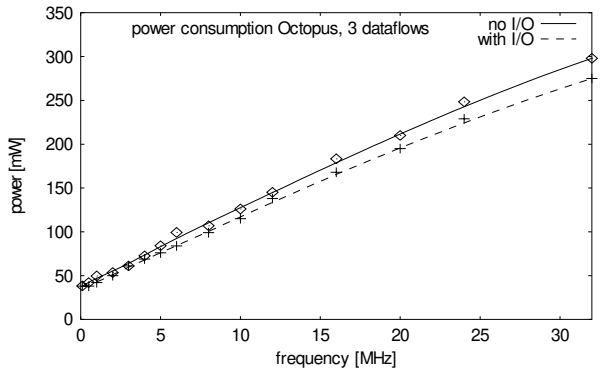


Figure 18: Power consumption Octopus switch with three data-streams, with and without I/O to the module.

Maybe somewhat surprisingly, the effect of having I/O with the module or not, is very small. If there is no I/O with the module, the energy consumption is even a little higher than when there is actual data traffic with the modules. This effect is due to the fact that the activity of the switch relatively to the activity of the MICs is much higher than in the previous setup. This requires more energy since the FPGA (i.e. switch) is less energy efficient than the microcontroller (i.e. MIC). The arbitration phase takes 31% of the total time needed to transfer one ATM cell, and the actual data-transfer phase takes 69% of the total time.

The effect of clock gating to the energy consumption

To determine the usefulness of our clock gating strategy and guarding of data on the energy consumption of the switch, we have made two versions of the switch: one optimised that uses clock gating and data guarding wherever possible, and one traditional design that did not use these techniques. Figure 19 shows the increase in power consumption of the traditional design versus the frequency. The figure plots two graphs, one when no data flow occurs and the switch is idle, and another when one connection between two modules is active.

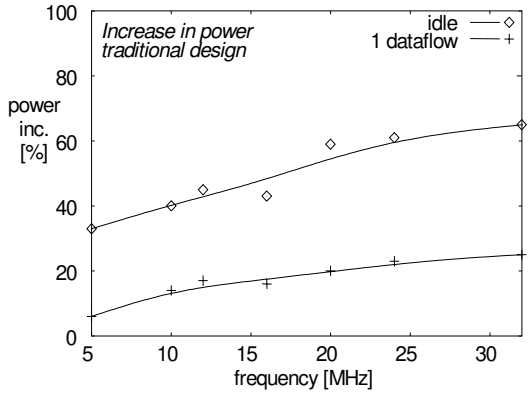


Figure 19: Effect of clock-gating to the power consumption of the Octopus switch.

When the switch is idle, the increase in energy consumption of the traditional design can be up to 65%. The optimised design is then in an energy efficient mode, and no useless clocks or data transitions occur. The main source of energy consumption that is left is due to the inherent energy consumption of the FPGA. This shows that – even for an FPGA that has a high energy consumption overhead – it is indeed worthwhile to optimise the clocking system.

When the switch is active, and there is one connection that communicates at maximum speed, the advantage is less: the increase in energy consumption of the traditional design is 25%. This can be expected since, when there is a connection, the attached MICs are also active and consume energy.

4.3.4 Conclusion

The realised prototype shows that it is relatively simple to implement an architecture that is suitable for becoming the interconnection structure of a mobile computer. The combination of a FPGA surrounded by several small micro-controllers proved to be a flexible prototyping testbed. The complexity of the architecture is low, which make it feasible to built the switch in a custom IC.

The measurements show that the costs of having a flexible dynamic scheduling can be significant. The overhead introduced by the arbitration and release phase is significant (i.e. 30%) when only one ATM cell is actually being transferred. A solution can be to have larger packets of multiple ATM cells, but this will lead to a bigger latency.

The performance of this prototype provides bandwidth guarantees and enough bandwidth for many multimedia applications. The power management that was used in the switch showed to be very effective. The energy consumption of the switch is strongly related to the number of communication streams in the switch. When the switch does not need to communicate, it is in an energy saving mode.

Clock gating and data guarding is being used in the switching fabric whenever possible. The effect of these techniques in energy savings is quite significant, and can be up to 65% when the switch is idle.

4.4 Summary and conclusions

In this chapter we discussed the design of an interconnection architecture for a handheld mobile multimedia computer. Energy management is the general theme in the design of the system architecture since battery life is limited and battery weight is an important factor. We have shown that there is a vital relationship between hardware architecture, operating system architecture and applications architecture, where each benefits from the others. In our architecture we have applied several supplementary energy reduction techniques on all levels of the system. Achieving high energy efficiency requires first of all the elimination of the waste that typically dominates the energy consumption in general-purpose processors. The second main principle used is to have a high locality of reference. The philosophy is that all operations that are required on the data should be done at the place where it the most efficient, thereby also minimising the transport of data through the system.

The interconnect of the architecture is based on a switch, called *Octopus*, which interconnects a general-purpose processor, programmable (multimedia) devices (called modules), and a wireless network interface. In our model, the interconnection network is transparent and provides only a direct connection between two functional modules. The Octopus switch transmit and receive ports are simple, containing only minimal buffering and arbitration functionality. The switch supports two basic connection types: ad-hoc connections for traffic with no hard real-time requirements, and guaranteed connections for traffic with (hard) real-time requirements. To assign the bandwidth in the switch we use static scheduling for guaranteed connections, and dynamic scheduling for ad-hoc connections. A certain portion of the bandwidth is allocated to traffic of guaranteed connections, and any unused bandwidth can be used for other traffic.

The architecture uses a locally synchronous, globally asynchronous timing methodology. The data paths through the switch only consume energy when data is being transferred, leaving most of the switch turned off nearly all the time. This is achieved by using clock gating and data guarding techniques in the switching fabric and energy management mechanisms in the module interface controllers.

We have built a prototype of this architecture from off-the-shelf VLSI components that was easy to design and test. A key goal motivating the design has been simplicity and flexibility. A Field Programmable Gate Array can be programmed to operate as the interconnection switch that connects the modules. Each port of the switch is connected to a micro-controller that performs routing, connection establishment, and provides the interface between the switch and the connected modules.

The performance of this prototype provides bandwidth guarantees and enough bandwidth for many multimedia applications. The power management that was used in

the switch showed to be very effective. The energy consumption of the switch is strongly related to the number of communication streams in the switch. When the switch does not need to communicate, it is in an energy saving mode. Clock gating and data guarding is being used in the switching fabric whenever possible. The effect of these techniques in energy savings is quite significant, and can be up to 65% when the switch is idle.

The prototype showed that it is already feasible with standard components to build an energy efficient architecture that allows many devices in the system to be turned off (including the CPU), while still providing enough performance to support multimedia applications. Having an energy efficient architecture that is capable to handle adaptability and flexibility in a mobile multimedia environment requires more than just a suitable hardware platform. First of all we need to have an operating system architecture that can deal with the hardware platform and the adaptability and flexibility of its devices. Optimisations across diverse layers and functions, not only at the operating systems level, is crucial. Managing and exploiting this diversity is the key system design problem. A model that encompasses different levels of granularity of the system is essential in the design of an energy management system and in assisting the system designer in making the right decisions in the many trade-offs that can be made in the system design. Finally, to fully exploit the possibilities offered by the reconfigurable hardware, we need to have proper operating system support for reconfigurable computing, so that these components can be reprogrammed adequate when the system or the application can benefit from it.

References

[1] Adam J.F., Houh H.H., Tennenhouse D.L.: “Experience with the VuNet: a network architecture for a distributed multimedia system”, *Proceedings of the IEEE 18th Conference on Local Computer Networks*, pp. 70-76, Minneapolis MN, September 1993.

[2] Anderson T.E., Owicki S.S., Saxe J.B., Tacker C.P.: “High speed switch scheduling for local area networks”, Proceedings *ACM ASPLOS V*, pp. 98-110, 1992.

[3] Barham P., Hayter M., McAuley D., Pratt I.: “Devices on the Desk Area Network”, March 1994.

[4] Benini L., De Micheli G.: “Dynamic Power Management, design techniques and CAD tools”, *Kluwer*

[5] Doyle van Meter, R.: “A brief survey of current work on network attached peripherals”, *ACM Operating Systems Review*, Jan. 1996.

[6] Eberle H., Oertli E.: “Switcherland: a QoS communication architecture for workstation clusters”, *Proceedings ISCA ’98 – 25th annual Int. Symposium on Computer Architecture*, Barcelona, June 1998.

[7] Havinga P.J.M., Smit G.J.M., Bos M.: “Energy efficient wireless ATM design”, *proceedings second IEEE international workshop on wireless mobile ATM implementations (wmATM’99)*, pp. 11-22, June, 1999.

[8] Havinga P.J.M., Smit G.J.M.: “Octopus: embracing the energy efficiency of handheld multimedia computers” , *proceedings fifth annual ACM/IEEE international conference on mobile computing and networking (Mobicom’99),* pp.77-87, August 1999.

[9] Havinga P.J.M., Smit G.J.M.: “Octopus – an energy-efficient architecture for wireless multimedia systems”, *Proceedings Program for Research on Integrated Systems and Circuits (ProRISC’99)*, pp. 185-192, November 1999.

[10] Hayter M.D., McAuley D.R.: “The desk area network”, *ACM Operating systems review*, Vol. 25 No 4, pp. 14-21, October 1991.

[11] Hui J.: “Switching and traffic theory for integrated broadband networks”, *Kluwer Academic Press*, 1990.

[12] Karol M., Hutchy M. Morgan S.: “Input versus output queueing on a space-division packet switch”, *IEEE transactions on communication*, 35(12), pp. 1347-1356, 1987.

[13] Leijten J.A.J.: “Real-time constrained reconfigurable communication between embedded processors”, *Ph.D. thesis, Eindhoven University of Technology*, November 1998.

[14] Prycker: “Asynchronous Transfer Mode”, 1991.

[15] Smit G.J.M.: “The design of central switch communication systems for multimedia applications”, *Ph.D. thesis, University of Twente*, 1994.

[16] Truman T.E., Pering T., Doering R., Brodersen R.W.: The InfoPad multimedia terminal: a portable device for wireless information access”, *IEEE transactions on computers*, Vol. 47, No. 10, pp. 1073-1087, October 1998.

[17] Yeap G.K.: “Practical low power digital VLSI design”, *Kluwer Academic Publishers*, ISBN 0-7923-80.

[18] Zhang H., Wan M., George V., Rabaey J.: “Interconnect architecture exploration for low-energy reconfigurable single-chip DSPs”, *Proceedings of the WVLSI*, Orlando, Fl, April 1999.

5

Energy-efficient wireless communication

In this chapter we present an energy-efficient highly adaptive network interface architecture and a novel data link layer protocol for wireless networks that provides Quality of Service (QoS) support for diverse traffic types[1]. Due to the dynamic nature of wireless networks, adaptations in bandwidth scheduling and error control are necessary to achieve energy efficiency and an acceptable quality of service.

In our approach we apply adaptability through all layers of the protocol stack, and provide feedback to the applications. In this way the applications can adapt the data streams, and the network protocols can adapt the communication parameters.

5.1 Introduction

As already observed before in the previous chapters, the energy consumption of portable computers like PDAs and laptops is the limiting factor in the amount of functionality that can be placed in these devices. More extensive and continuous use of wireless network services will only aggravate this problem. However, even today, research is still focused on performance and (low power) circuit design. There has been substantial research in the hardware aspects of mobile communications energy-efficiency, such as low-power electronics, power-down modes, and energy efficient modulation. However, due to fundamental physical limitations, progress towards further energy-efficiency will become mostly an architectural and software-level issue.

We have shown in Chapter 2 that it is more effective to save energy by a carefully designed architecture of the mobile, the communications device and wireless communication protocols that consider judicious use of the available energy [37].

---

[1] Major parts of this chapter have been presented at the Second *IEEE International Workshop on Wireless Mobile ATM Implementations (wmATM'99*), 1999 [32], and will appear in the *Journal on Mobile Networks and Applications (MONET),* 2000 [33].

Energy reduction should be considered in the whole system of the mobile and through all layers of the protocol stack, including the application layer. In this chapter we address the issue of *energy efficiency* in the data link network layer protocols for wireless networks. These protocols typically address network performance metrics such as throughput, efficiency, fairness and packet delay. This chapter addresses the additional goal of efficient energy usage of the mobiles. Considerations of energy efficiency are fundamentally influenced by the trade-off between energy consumption and achievable Quality of Service (QoS). The aim is to meet the required QoS, while minimising the required amount of energy.

The objective of this chapter is to present the design and analysis of a network interface and a medium access protocol, referred to as E^2MaC. The design is driven by two major factors. The first factor is that the design should be energy-efficient since the mobiles typically have limited energy capacity. The second factor is that it should provide support for multiple traffic types, with appropriate Quality of Service levels for each type.

Service model

Traditional communication networks provide a single service model that delivers packets on a best effort basis. The available bandwidth is shared by competing senders on a per packet basis. As a consequence, the packets experience an unpredictable – and possibly very long – delay in getting to their destination. For many traditional applications this is not a real problem as long as the overall delays are not excessive. Other applications, however, that e.g. transfer digitised voice, require a predictable service model. Circuit switched networks can offer such a service with a fixed slot of bandwidth allocated for use by a sender in each time period, and with equal delivery time for each slot.

It is expected that the new generation of wireless networks will carry diverse types of multimedia traffic. Multimedia services, like packet audio and video, and real-time services, e.g. for process control, have strict communication constraints. Multimedia services are typically sensitive to delay and jitter (variations in delay) and demand high bandwidths, but may be prepared to tolerate some data loss [36]. For example, dropping several pixels in a high-resolution image may not be noticeable. Even dropping one frame now and then from a video sequence at 25 frames per second can be tolerated. Hard-real-time applications usually have lower bandwidth requirements, but demand predictable delay and cannot tolerate any errors.

Quality of Service (QoS) is an attractive model for resource allocation and sharing, and is applied in communication networks like ATM [8]. QoS guarantees provide the basis for modern high-bandwidth and real-time multimedia applications like teleteaching and video conferencing. All the multimedia service types and the specific requirements can be expressed in terms of the QoS expected by the application. The notion of QoS service originally stems from communication, but because of its potential in the allocation of all scarce resources, it has found its way into other domains, e.g. operating systems [40]. QoS then involves all layers that are below the application. QoS based resource

allocation is based on services or users requesting a resource on some level of quality from a service provider.

In statically connected systems, the service provider will try to reserve resources (end-to-end) upon a request from a user. If the service provider grants the request (possibly after negotiation), the two parts have a QoS contract that gives some notion of guarantee that the service level in the contract shall be sustained. The service user will often rely on the availability of the resources specified in the QoS contract. However, in dynamically connected systems like wireless networks, the availability and quality of resources are generally unpredictable. Therefore, a service provider generally cannot issue a QoS contract that the service user can rely on. QoS based resource management in mobile systems therefore must take this fundamental difference into account.

Wireless system architecture

Wireless LANs can be classified as distributed (*ad-hoc*) or *centralised* systems. Essentially, the existence or lack of fixed wired infrastructure differentiates them.

- In *ad-hoc networks* the infrastructure is build up of mobiles which establish wireless links between them and build a network topology allowing multihop connectivity. Its key characteristics are that there is no fixed infrastructure, and that there is wireless multihop communication, dynamically set up and reconfigurable as mobiles move around.
- *Centralised systems* consist of base stations and mobiles. Its key characteristics are that there is some fixed wired infrastructure, which is always accessible through a single hop wireless link. The base stations are connected to the fixed network and support the communication of the mobiles in range of the base station's radio.

Ad-hoc networks provide more flexibility than centralised systems. However, in ad-hoc networks the data possibly has to pass multiple hops before it reaches its final destination. This leads to a waste of bandwidth as well as an increased risk of data corruption, and thus potentially higher energy consumption (due to the required error control mechanism). Only if the source and destination mobile are in each others reach, ad-hoc networking can be more efficient. However, the use of ad-hoc networks is limited because in general there is not much mobile-to-mobile communication, and in many situations a fixed network is still required.

Although ad-hoc networks are more flexible than centralised systems, they are less suitable for the design of low energy consuming mobiles. The assumption is that mobiles will always have a limited amount of energy, whereas the wired base-stations will have virtually unlimited energy. In a centralised system the base station can therefore be equipped with more intelligent and sophisticated hardware, that probably has a significantly higher energy consumption than the hardware required in the mobile. Portables can then be offloaded with some functionality that will be handled by the base station.

In centralised systems it is further much easier to provide a certain quality of service for applications or users. Since both energy requirements and QoS are our main targets, we will only consider a centralised system here.

Overview of the chapter

In this chapter we will consider an ATM based infrastructure network where a base-station co-ordinates access to one or more channels for mobiles in its cell. The channels can be individual frequencies in FDMA, time slots in TDMA, or orthogonal codes or hopping patterns in case of spread-spectrum. Hybrid TDMA/CDMA schemes benefit from both the capacity of TDMA schemes to handle high bit-rate packet-switched services, and the flexibility of CDMA techniques that allow smooth coexistence of different types of traffic [5]. In this chapter we will deal with three main aspects involved with energy-efficient wireless communication: Medium Access Control (MAC) design, error control, and network interface architecture.

Section 5.2 first presents the basics of wireless data link layer design issues are discussed, i.e. the wireless link limitations, the basic wireless networking functions needed, and introduces the concept of QoS renegotiations. Section 5.3 determines the main sources of energy consumption on wireless interfaces, which provide us the main principles of energy efficient MAC design. Then, Section 5.4 presents a short introduction to ATM and the peculiarities when applied to a wireless system. Section 5.5 presents various error-control alternatives and their consequences on energy consumption. Then, Section 5.6 describes the basic principles and mechanisms of the network interface architecture, and a new MAC protocol E^2MaC whose design is driven by energy consumption, diverse traffic type support, and QoS support considerations. Section 5.7 provides an evaluation of the performance of the E^2MaC protocol. Related work is presented in Section 5.8, and we will finish with some conclusions.

5.2 Wireless data link layer network design issues

The context in this section is data link-level communication protocols for wireless networks that provide multimedia services to mobile users. As mentioned before, portable devices have severe constraints on the size, the energy consumption, and the communication bandwidth available, and are required to handle many classes of data transfer over a limited bandwidth wireless connection, including delay sensitive, real-time traffic such as speech and video. This combination of limited bandwidth, high error rates, and delay-sensitive data requires tight integration of all subsystems in the device, including aggressive optimisation of the protocols to suit the intended application. The protocols must be robust in the presence of errors; they must be able to differentiate between classes of data, giving each class the exact service it requires; and they must have an implementation suitable for low-power portable electronic devices.

5.2.1 The ISO/OSI network design model

Data communication protocols govern the way in which electronic systems exchange information by specifying a set of rules that, when followed, provide a consistent, repeatable, and well-understood data transfer service. In designing communication protocols and the systems that implement them, one would like to ensure that the protocol is correct and efficient. The ISO/OSI model is a design guide for how network software in general should be built. In this model, protocols are conceptually organised as a series of layers, each one built upon its predecessor. Most network architectures use some kind of layering model, although the specific layers may not be an exact match with the layers defined in the ISO/OSI model.

The rationale behind this layering approach is that it makes in principle possible to replace the implementation of a particular layer with another implementation, requiring only that each implementation provide a consistent interface that offers the same services and service access points to the upper layer. Thus, the goal of service abstraction is modularity and freedom to choose the implementation that is best suited for a particular environment. However, while this model provides an excellent starting point for conceptually partitioning a set of protocol services, it has two implicit assumptions that fail to hold in many practical contexts [78]. First, there is the assumption that cost of abstraction and separation is negligible compared to the gained modularity and flexibility. Second, there is the assumption that interchanging layers that provide the same logical services – for example, a wired physical layer and a wireless physical layer – provide equivalent service.

These assumptions are in general not valid for mobile systems and can impose severe limitations. For example, although the TCP specification contains no explicit reference to the characteristics of the lower layers, implicitly in the timeout and retransmission mechanisms there are the assumption that the error rate is low, and that lost packets occur due to network congestion. TCP has no way of distinguishing between a packet corrupted by bit errors in the wireless channel from packets that are lost due to congestion in the network. The applied measures result on a wireless channel in unnecessary increases in energy consumption and deterioration of QoS. This example attests the need to tailor protocols to the environment they operate in. Separating the design of the protocol from the context in which it exists leads to penalties in performance and energy consumption that are unacceptable for wireless, multimedia applications.

The context of this section is mainly the data link layer. Data link protocols are usually divided into two main functional components: the *Logical Link Control* (LLC) and the *Medium Access Control* (MAC), that are responsible for providing a point-to-point packet transfer service to the network, and a means by which multiple users can share the same medium. The main task of the Data Link layer protocols on a wireless network is to provide access to the radio channel. Wireless link particularities, such as high error rate and scarce resources like bandwidth and energy, and the requirements to provide access for different connection classes with a variety of traffic characteristics and QoS requirements, makes this a non-trivial task. It requires a flexible, yet simple scheme that should be able to adjust itself to different operating conditions in order to satisfy all

connections and overall requirements like efficient use of resources like energy and radio bandwidth. The protocols have to support traffic allocation according an agreed traffic contract of a connection, but must also be flexible enough to adapt to the dynamic environment and provide support for QoS renegotiations. It further has to provide error control and mobility related services.

5.2.2 Wireless link restrictions

The characteristics of the wireless channel the Data Link protocol has to deal with are basically high bit error rate (BER), limited bandwidth, broadcast transmission, high energy consumption and half duplex links.

Wireless networks have a much higher *error rate* than the normal wired networks. The errors that occur on the physical channel are caused by phenomena such as signal fading, transmission interference, user mobility and multi-path effects. Typically, the bit error rates observed may be as bad as 10^{-3} or 10^{-4}, which is far more worse than assumed by networks with wired connections. Additionally, the errors show a dynamic nature due to movement of the mobile. In indoor environments propagation mechanisms caused by the interactions between electromagnetic fields and various objects can increase error rates considerably. Especially in the outer regions of the radio cell, the low signal-to noise ratio (SNR) makes wireless link errors a norm rather than an exception in the system.

The *available bandwidth* on a wireless channel is usually much less than offered by wired networks. Consequently, an important design consideration in the design of a protocol, is the efficient use of the available bandwidth.

Closely related to this is the amount of *energy* that is needed to transmit or receive data. The required amount of energy is high, and typically depends on the distance that the radio signal has to propagate between sender and receiver. Since wireless networks for mobile systems will be used more widely and more intensively, the energy consumption that is required to communicate will take a large part of the available energy resources (batteries) of the mobile. So energy consumption will be another main design constraint for the wireless data link protocol of the mobile. In general, saving energy for the base station is not really an issue, as it is part of the fixed infrastructure and typically obtains energy from a mains outlet. However, since the current trend is to have ever smaller area cell sizes, and the complexity of the base station is increasing, this issue might become more important in the future mainly because of economical and thermal reasons.

By their nature wireless radio transmission is a *broadcast medium* to all receivers within the range of a transmitter. This characteristic gives rise to several problems in a wireless environment with multiple cells and mobiles. A mobile that is in reach of more than one base station and communicates with only one of them can cause errors on the communication in the neighbouring cell. Even if the mobiles are just in reach of one base station, interference between mobiles in different cells can also cause errors. Solutions on the physical layer are possible (colouring schemes with multiple frequencies, spread spectrum technologies, near field radio [72], etc.) but are out of the scope of our research. However, provisions for handoff when a mobile moves from one

area cell to another, are important and have consequences for the design of a data link protocol.

A radio modem transceiver typically has one part dedicated to transmission, and the other part to reception. Consequently, the radio channel is generally used in *half duplex mode*. The only way to allow full duplex operation over the radio channel is to duplicate transceiver hardware and use two sub-bands in the frequency band, each of them being used for one-way transmission. Because such a solution is not economically viable, and also raises some technical problems, the data link protocol should be designed in such a way that connections in both directions are treated fairly.

5.2.3 Basic wireless networking functions

The challenge of a wireless data link protocol is to overcome the harsh reality of wireless transmission and to provide mobility and multimedia services. The data link layer of a wireless network has to provide assistance to several basic functions: *QoS management* when a connection is initiated or when the operating conditions have changed; *traffic and resource allocation* according to a traffic contract; *error control* to overcome the effect of errors on the wireless link, *flow control* to avoid buffer overflow and also to discard cells of which the maximum allowed delay is exceeded due to retransmissions; *security and privacy* for the mobile user, and *mobility features* to allow handover when a mobile moves to another area cell. In this section we will discuss these items briefly and describe the consequences for the data link layer.

QoS management

To support diverse traffic over a wireless channel, the notion of QoS of a connection is useful. Setting up a connection involves negotiation along a path from sender to receiver in order to reserve the required resources to fulfil the QoS needed. Due to the dynamic nature of wireless channels and the movement of the mobile the agreed QoS level in one or more contracts generally cannot be sustained for a longer period. These situations are not errors, but are modus operandi for mobile computers. Therefore, these situations must be handled efficiently, and QoS renegotiations will occur frequently. Multimedia applications can show a more dynamic range of acceptable performance parameters depending on the user's quality expectations, application usage modes, and application's tolerance to degradation.

Traffic and resource allocation

Each accepted connection has a certain traffic contract that describes the traffic type and required QoS parameters. A slot-scheduler is responsible to assign slots in a transmission frame according to the various traffic contracts. At the same time it must attain a high utilisation of the scarce radio bandwidth and minimise the energy consumption for the mobile.

Error control

Due to the high bit error rate (BER) that is typical for a wireless link, many packets can be corrupted during transmission. If this rate exceeds the allowable cell loss rate of a connection, an effective and efficient error control scheme must be implemented to handle such situations. At the radio physical level redundancy for detecting symbols reduces the bit error rate for the first time. However, it is usually inefficient to provide a very high degree of error correction, and some residual errors pass through. The residual channel characteristic is based on *erases*, i.e. missing packets in a stream. Erasures are easier to deal with than errors, since the exact location of the missing data is known. Then, integrated into the MAC layer (and possibly also into the higher layers), an error control scheme further enhances transmission quality by applying error correction and/or retransmission schemes.

Since different connections do not have the same requirements concerning cell loss rate and cell transfer delay, different error control schemes must be applied for different connection types [60]. The alternatives are Forward Error Correction (FEC), retransmission techniques like automatic repeat request (ARQ), or hybrid FEC/ARQ schemes. To reduce the overhead and energy involved the error control scheme can also be adapted to the current error condition of the wireless connection. The error control mechanisms should trade off complexity, buffering requirements and energy requirements (taking into account the required energy for both computation and communication) for throughput and delay.

Flow control

A connection involves buffering at several places on the path between sender and receiver. Traffic type requirements concerning delay, and implementation restrictions on the buffer capacity generally limit the amount of buffer space available to a connection. Due to the dynamic character of wireless networks and user mobility, the stream of data might be hindered on the way from source to destination. Therefore, flow control mechanisms are needed to prevent buffer overflow, but also to discard packets that have exceeded the allowable transfer time. Depending on the service class and QoS of a connection a different flow control can be applied. For instance, in a video application it is useless to transmit images that are already outdated. It is more important to have the 'fresh' images. For such traffic the buffer is probably small, and when the connection is hindered somewhere, the oldest data will be discarded and the fresh data will be shifted into the fifo. Flow control can cover several hierarchical layers, but in the context of link access protocols we mainly deal with the buffering required directly at both sides of the wireless link.

Security and privacy

Since eavesdropping of the data bits is a real threat because they will be transmitted over the wireless air interface, security and privacy are important issues in wireless systems. These items are important on two levels: protection of the data on the wireless link, and end-to-end application security. The MAC layer is only capable to provide some basic

protection of the data on the wireless link. Since it is hard to make this very secure, end-to-end security will be the most attractive and secure solution.

Mobility features

In a wireless environment the mobility of the mobile will enforce handover procedures when the mobile moves from one area cell to another. As the current trend is that the radius of an area cell decreases (because of the higher bandwidth density and lower energy requirements) handover situations will be encountered frequently.

The task of the link layer is to provide the higher layers of the mobile with information about which area cells are in range, and provide services to actually handle the handover. The radio link quality will be the first parameter to be taken into account for the handover initiation procedure. In the new area-cell a new connection has to be prepared and bandwidth reserved. When a mobile is being handovered to a new area cell, the connection will be dropped if there is insufficient bandwidth to support the connection. Since dropping connections is more undesirable than blocking new connection requests, some bandwidth can be reserved in neighbouring area cells in advance, before the mobile reaches that area cell. It is possible to provide a general pool of bandwidth that can be used for new connections. If it is possible to predict the movement of mobiles, then bandwidth can be saved since not in all neighbouring area cells bandwidth has to be reserved [75].

5.2.4 QoS renegotiation

In a wired network, QoS is usually guaranteed for the lifetime of a connection. In a wireless environment these guarantees are not realistic due to the movement of mobiles and the frequent occurrence of errors on the wireless link.

To prevent service interruptions in a proactive fashion, QoS renegotiations may be required to assure a lower, but deliverable level of service. The difficulty is to provide a mechanism with which QoS parameters of an active connection can be changed dynamically.

QoS control is important during the handover procedure as the mobile moves into a cell and places demand on resources presently allocated to connections from other mobiles. If a mobile faces a significant drop in bandwidth availability as it moves from one cell to another, rather than dropping the connection, the QoS manager might be able to reallocate bandwidth among selected active connections in the new cell. The QoS manager of the new cell selects a set of connections, called *donors*, and changes their bandwidth reservations to attempt satisfactory service for all. To quickly process handover requests, the QoS manager can use cached bandwidth reserves. This cache can then be replenished after the QoS manager has obtained the required bandwidth from the donor connections.

When the mobile moves to a cell where the traffic on the wireless link is much higher, it is not just the current connection that needs renegotiations, even other connections of applications in that area cell may become subject to the QoS renegotiations to allow the

new mobile access. The movement of the mobile also influences the (already poor) quality of wireless channels and can introduce dynamic changes in error rate. Especially an indoor environment with small rooms and corridors can cause interactions between the electromagnetic fields and various objects. These interactions can increase error rates considerably. To be able to guarantee an agreed QoS for – especially error sensitive – connections, error recovery techniques using error correcting codes or retransmission is required. In addition to this, before completely closing a connection on a faulty link, the link errors can be gracefully tolerated by renegotiations of QoS. Many multimedia applications can deal with varying bandwidth availability once provided with sufficient information about the operating conditions. For instance, video transmission schemes may adjust their resolution, their frame rates, and encoding mechanism to match the available bandwidth or deal with the current error conditions.

5.3 Energy-efficient wireless MAC design

The objective of an energy efficient MAC protocol design is to maximise the performance while minimising the energy consumption of the *mobile*. These requirements often conflict, and a trade-off has to be made.

Sources of unessential energy consumption

The focus of this work is on minimising the energy consumption of a mobile and in particular the wireless interface, the transceiver. Typically, the transceiver can be in five modes; in order of increasing energy consumption, these are off, sleep, idle, receive, and transmit. In transmit mode, the device is transmitting data; in receive mode, the receiver is receiving data; in idle mode, it is doing neither, but the transceiver is still powered and ready to receive or transmit; in sleep mode, the transceiver circuitry is powered down, except sometimes for a small amount of circuitry listening for incoming transmissions [50].

Several causes for unessential energy consumption exist. We will review in this section some of the most relevant sources of unessential energy consumption.

- First of all, most applications have low traffic needs, and hence the transceiver is *idling* most of the time. Measurements show that on typical applications like a web-browser or e-mail, the energy consumed while the interface is on and idle is more than the cost of actually receiving packets [54][74].
- Second, the typical *inactivity threshold*, which is the time before a transceiver will go in the off or standby state after a period of inactivity, causes the receiver to be in a too high energy consuming mode needlessly for a significant time.
- Third, in a typical wireless broadcast environment, the receiver has to be powered on at all times to be able to *receive messages* from the base station, resulting in significant energy consumption. The receiver subsystem typically receives all packets and forwards only the packets destined for this mobile. Even in a scheme in

which the base transmits a traffic schedule to a mobile, the mobile has to receive the traffic control information regularly to check for waiting downlink traffic. When the mobile is not synchronised with the base-station, then it might have to receive 'useless' data before it receives the traffic control.

- Fourth, significant time and energy is further spent by the mobile in switching from transmit to receive modes, and vice-versa. The *turnaround time* between these modes typically takes between 6 to 30 microseconds. The transition from sleep to transmit or receive generally takes even more time (e.g. 250 μs for WaveLAN). A protocol that assigns the channel per slot will cause significant overhead due to turnaround.
- Fifth, in broadcast networks *collisions* may occur (happens mainly at high load situations). This causes the data to become useless and the energy needed to transport that data to be lost.
- Sixth, the *overhead of a protocol* also influences the energy requirements due to the amount of 'useless' control data and the required computation for protocol handling. The overhead can be caused by long headers (e.g. for addressing, mobility control, etc), by long trailers (e.g. for error detection and correction), and by the number of required control messages (e.g. acknowledgements). In many protocols the overhead involved to receive or transmit an amount of data can be large, and may depend on the load of the network. In general, simple protocols need relatively less energy than complex protocols.
- Finally, the high *error rate* that is typical for wireless links is another source of energy consumption. First, when the data is not correctly received the energy that was needed to transport and process that data is wasted. Secondly, energy is used for error control mechanisms. On the data link layer level error correction is generally used to reduce the impact of errors on the wireless link. The residual errors occur as burst errors covering a period of up to a few hundred milliseconds. To overcome these errors retransmission techniques or error correction techniques are used. Furthermore, energy is consumed for the calculation and transfer of redundant data packets and an error detection code (e.g. a CRC). Finally, because in wireless communication the error rate and the channel's signal-to-noise ratio (SNR) vary widely over time and space, a fixed-point error control mechanism that is designed to be able to correct errors that rarely occur, wastes energy and bandwidth. If the application is error-resilient, trying to withstand all possible errors wastes even more energy in needless error control.

We define *energy efficiency* as the quotient between the intrinsic amount of energy needed to transfer a certain quantity of data and the actually used amount of energy (including all overheads). We will use this metric to quantify how well a MAC protocol behaves with respect to its energy consumption.

Main principles of energy-efficient MAC design

The above observations are just some of the possible sources of unessential energy consumption related to the medium access control protocol. We have no intention to provide a complete list. We can, however, deduce the following main principles that can be used to design a MAC protocol that is energy efficient for the mobile.

- *Avoid unsuccessful actions of the transceiver.* *(P1)*

  Two main topics cause unsuccessful actions: collisions and errors.

  Every time a *collision* occurs energy is wasted because the same transfer has to be repeated again after a backoff period. A protocol that does not suffer from collisions can have good throughput even under high load conditions. These protocols generally also have good energy consumption characteristics. However, if it requires the receiver to be turned on for long periods of time, the advantage diminishes.

  A protocol, in which a base-station broadcasts traffic control for all mobiles in range with information about when a mobile is allowed to transmit or is supposed to receive data, reduces the occurrence of collisions significantly. Collisions can only occur when new requests have to be made. New requests can be made per packet in a communication stream, per application of a mobile, or even per mobile. The trade-off between efficient use of resources and QoS determines the size to which a request applies. Note that this might waste bandwidth (but not energy) when slots are reserved for a request, but not used always. In such a reservation mechanism, energy consumption is further reduced because there is less need for a handshake to acknowledge the transfer.

  Errors on the wireless link can be overcome by mechanisms like retransmissions or error correcting codes. Both mechanisms induce extra energy consumption. The error control mechanisms can be adapted to the current error condition in such a way that it minimises the energy consumption needed and still provides (just) enough fault tolerance for a certain connection. Due to the dynamic nature of wireless networks, *adaptive error control* can give significant gains in bandwidth and energy efficiency [23][82]. This avoids applying error control overhead to connections that do not need it, and allows the possibility to apply it selectively to match the required QoS and the conditions of the radio link. Note that this introduces a trade-off between communication and computation [36]. Section 5.5 goes into more detail on this issue. A different strategy to reduce the effect of errors is to avoid traffic during periods of bad error conditions. This, however, is not always possible for all traffic types as it influences the QoS.

- *Minimise the number of transitions.* *(P2)*

  Scheduling traffic into bursts in which a mobile can continuously transmit or receive data – possibly even bundled for different applications –, can reduce the number of transitions. Notice, however, that there is a trade-off with QoS parameters like delay and jitter. When the traffic is continuous and can be scheduled for a longer period ahead, then the mobile does not even have to listen to the traffic

control since it knows when it can expect data or may transmit. The number of transitions needed can also be reduced by collecting multiple requests of multiple applications on a mobile, and by piggy-backing new requests on current data streams. Simple protocols can further reduce the required number of transitions due to the low amount of control messages needed.

- *Synchronise the mobile and the base station.* *(P3)*

  Synchronisation is beneficial for both uplink (mobile host to base station) and downlink (base station to mobile host) traffic. When the base-station and mobile are synchronised in time, the mobile can go in standby or off mode, and wake up just in time to communicate with the base-station. The energy consumption needed for downlink traffic can be reduced when the time that the receiver has to be on – just to listen whether the base-station has some data for the mobile – can be minimised. The premise is that the base has plenty of energy and can broadcast its beacon frequently. The application of a mobile with the least tolerable delay determines the frequency by which a mobile needs to turn its receiver on. If the wake-up call of the communication is implemented with a low-power low-performance radio, instead of the high-performance high-energy consuming radio, then the required energy can be reduced even more.

- *Migrate as much as possible work to the base-station.* *(P4)*

  In a centralised wireless system architecture, the base-station that is connected to the fixed network and a mains outlet, can perform many tasks in lieu of the mobile. The calculation of a traffic control that adheres the QoS of all connections is an example of such task. At higher levels, the base-station can also perform tasks to process control information, or to manipulate user information that is being exchanged between the mobile device and a network-based server (see Chapter 2).

Note that these principles can reduce the energy consumption of the wireless interface. The energy consumption of the mobile system is much more complex and comprises many issues. The total achieved energy reduction is thus based on many trade-offs. For example, grouping traffic in multimedia video streams to minimise the number of transitions requires the data to be buffered in the client's memory. The required amount of energy needed for buffering reduces the effect of the energy savings principle in some sense.

There are many ways in which these principles can be implemented. We will consider an environment suitable for multimedia applications in which the MAC protocol also has other requirements like provisions for QoS of real-time traffic, and to provide a high throughput for bulk data. Due to the dynamic character of wireless multimedia systems and time-varying radio channel conditions, flexibility and adaptation play a crucial role in achieving an energy efficient design.

We have chosen to adopt *Asynchronous Transfer Mode* (ATM) mechanisms for the wireless network. We have no intention to build the full-blown B-ISDN ATM protocol stack, but merely adopt the small, fixed size packet and the QoS mechanisms. In the next section we give a short introduction to ATM and motivate why it is suitable for building an energy-efficient wireless network.

5.4 ATM

The challenge of designing a network that can cope with all different service types led to the development of the *Asynchronous Transfer Mode* (ATM). ATM is able to support different kind of connections with different QoS parameters. ATM technology provides deterministic or statistical guarantees with connection-oriented reservations. The original intent of ATM was to form a backbone network for high speed data transmission regardless of traffic type. Later, ATM has been found to be capable of more. Today, ATM scales well from backbone to the customer premises networks and is independent of the bit rate of the physical medium. By preserving the essential characteristics of ATM transmission, wireless ATM offers the promise of improved performance and QoS, not attainable by other wireless communication systems like cellular systems, cordless, or wireless LANs. In addition, wireless ATM access provides location independence that removes a major limiting factor in the use of computers and powerful telecom equipment over wired networks [58].

ATM transports data in small, fixed size (in B-ISDN ATM 53-byte) packets called *cells*. Having a fixed cell size allows for a simple implementation of ATM devices, and results in a more deterministic behaviour. Small cells have the benefit of a small scheduling granularity, and hence provide a good control over queuing delays. This also allows rapid switching that supports any mix of delay-sensitive traffic and bursty data traffic at varying bit rates. ATM carries cells across the network on connections known as *Virtual Circuits*. With a Virtual Circuit the flow of data is controlled at each stage in its path from source to destination. In ATM, the QoS requirements of Virtual Circuits are a key element as it relates to how cells for a Virtual Circuit are processed. The connection-oriented nature allows the user to specify certain QoS parameters for each connection. Network resources are reserved upon the acceptance of a Virtual Circuit, but they are consumed only when traffic is actually generated.

5.4.1 ATM service classes

The ATM service architecture uses procedures and parameters for traffic control and congestion control whose primary role is to protect the network and end-system to achieve network performance objectives. The design of these functions is also aimed at reducing network and end-system complexity while maximising network utilisation. The *ATM service categories* represent service building blocks and introduce the possibility for the user to select specific combinations of traffic and performance parameters. Most of the requirements that are specific to a given application may be resolved by choosing an appropriate ATM Adaptation Layer (AAL). However, given the presence of a heterogeneous traffic mix, and the need to adequately control the allocation of network resources for each traffic component, a much greater degree of flexibility, fairness and utilisation of the network can be achieved by providing a selectable set of capabilities within the ATM-layer itself.

The ATM forum has specified the following ATM Service Categories (ASC). ATM Service Category relates quality requirements for a given set of applications and traffic characteristics to network behaviour.

- *Constant Bit Rate* (CBR). A category based on constant (maximum) bandwidth allocation. This category is used for connections that require constant amount of bandwidth continuously available during the connection lifetime. CBR is oriented to serve applications with stringent time delay and jitter requirements (like telephony), but is also suitable for any data transfer application which contains smooth enough traffic.
- *Variable Bit Rate* (VBR) for statistical (average) bandwidth allocation. This is further divided into real-time (rt-VBR) and non-real-time (nrt-VBR), depending on the QoS requirements. Rt-VBR is intended to model real-time applications with sources that transmit at a rate which varies in time (e.g. compressed images) and have strict delay constraints. Video-conference is a suitable application, in which the real-time constraint should guarantee a synchronisation of voice and image, and the network resources are efficiently utilised because of the varying bandwidth requirements due to compression. Nrt-VBR is for connections that carry variable bit rate traffic with no strict delay constraints, but with a required mean transfer delay and cell loss. Nrt-VBR can be used for data-transfer like response-time critical transaction processing (e.g. airline reservation, banking). The undetermined time constraints give the possibility to use large buffers.
- *Available Bit Rate* (ABR) where the amount of reserved resources varies in time, depending on network availability. The variation managed by the traffic control mechanisms is reported to the source via feedback traffic. Compliance to the variations from the feedback signal should guarantee a low cell loss ratio for the application. Generally, it is necessary to use large buffers to offer ABR service on the network due to the burst nature of the service. It has no guaranteed cell transfer delay, but just a minimum guaranteed bandwidth. This category provides an economical support to those applications that show vague requirements for throughput and delay and requires a low cell loss ratio. Applications are typically run over protocol stacks like TCP/IP, which can easily vary their emission as required by the ABR rate control policy.
- *Unspecified Bit Rate* (UBR) has no explicit resource allocation and does not specify bandwidth or QoS requirements. Losses and error recovery or congestion control mechanisms could be performed at higher layers, and not at lower network layers. UBR can provide a suitable solution for less demanding applications like data applications (e.g. background ftp) that are very tolerant to delay and cell loss. These services can take advantage of any spare bandwidth and will profit from the resultant reduced tariffs.

5.4.2 Admission control and policing

Setting up a virtual connection involves taking information on the required service class and QoS. Using this information the system negotiates along the path from source to

destination in order to reserve the necessary resources. A traffic contract specifies the negotiated characteristics of a virtual connection at an ATM User Network Interface (UNI). Each QoS parameter consists of a value pair, one representing the low end, and the other the high end. This is called the tolerable range.

Once admitted, the system continually checks that the virtual connection sends data according to its allowance, known as *policing*. When the value of the delivered QoS parameter falls outside the tolerable range, the contract is be violated.

Functions related to the implementation of QoS in ATM networks are usage parameter control (UPC) and connection admission control (CAC). In essence, the UPC function (implemented at the network edge) ensures that the traffic generated over a connection conforms to the declared traffic parameters. Excess traffic may be dropped or carried on a best-effort basis. The CAC function is implemented by each switch in an ATM network to determine whether the QoS requirements of a connection can be satisfied with the available resources.

5.4.3 Wireless ATM

At the moment there are already wireless LANs and wireless systems offering data services and mobile data. These mobile systems offer low bit rate wireless data transmission with mobility and roaming possibility. Wireless LANs offer mobility only in restricted, smaller areas of coverage without wide area roaming capabilities. The achieved bit rates are generally greater than with current mobile systems. The third generation mobile telecommunication systems, such as UMTS (Universal Mobile Telecommunication System) aim to achieve data services of up to 2 Mbit/s, which is a significant improvement over the second-generation mobile systems. However, the importance of speech service may overrun the 2 Mbit/s data service goals [58]. The third generation wireless networks will enable mobiles to carry integrated multimedia. Wireless ATM networks can be useful for these new generation wireless networks because of its ability to handle traffic of different classes and integrate them into one stream. A wireless ATM network consists generally of a cluster of base stations interconnected by a wired ATM network (see Figure 1).

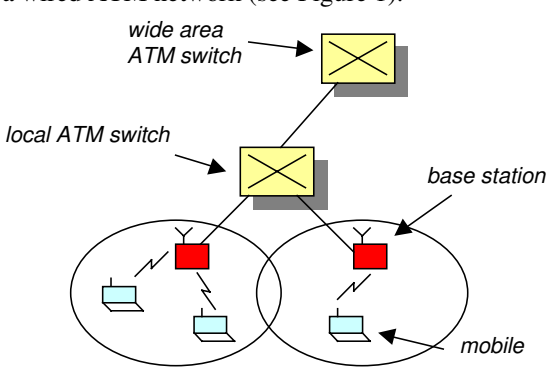


Figure 1: Wireless ATM architecture.

Originally, ATM was characterised by bandwidth on demand at megabits per second rates; it operates at very low bit error rate environments, supports packet switched transport, virtual circuit connections, and statistical sharing of the network resources among different connections.

Wireless networking is inherently unreliable and the bandwidth supported is usually lower than that of fixed networks. Various forms of interference on the wireless link result in high error rates, and thus introduces delay, jitter and an even lower effective bandwidth. Mobility of the user makes these problems even more dynamic and introduces the need for handover mechanisms when the user comes in reach of a different base-station.

The characterisation of ATM – that was designed for wired networks – seems rather contradictory with the operating conditions of wireless networks. Even with high redundancy introduced at several layers (i.e. physical, medium access control, transport and applications) the quality of service may not be guaranteed. Therefore, when adopting ATM in a wireless environment we need to adopt a more dynamic approach to resource usage. *Applications* must adapt their QoS requirements on the current operating environment. Explicit renegotiation of the QoS of a connection about the available resources between the application and the wireless system is essential in wireless ATM systems.

Since a connection typically involves both a fixed and a wireless part, the wireless link should support similar mechanisms as the fixed ATM network. Therefore it has to support all traffic types taking into account the characteristics of the wireless medium. The medium access protocol should be able to bridge the fixed and wireless world and provide ATM services transparently over a wireless link.

Wireless ATM is a topic on which many research activities are going on, e.g. Magic WAND [57], MEDIAN [18], NTT AWA [43]. Most projects aim to extend ATM to the mobile terminal. The main difference can be observed in air interface. No project explicitly addresses reduction in power consumption as a major issue.

5.5 Energy-efficient error control

Since high error rates are inevitable to the wireless environment, *energy-efficient error-control* is an important issue for mobile computing systems. This includes energy spent in the physical radio transmission process, as well as energy spent in computation, such as signal processing and error control at the transmitter and the receiver.

Error-control mechanisms traditionally trades off complexity and buffering requirements for throughput and delay [46][48][15]. In our approach we apply energy consumption constraints to the error-control mechanisms in order to *enhance energy efficiency under various wireless channel conditions*. In a wireless environment these conditions not only vary dynamically because the physical conditions of a communication system can vary rapidly, but they can also vary because the user moves from an indoor office

environment to a crowded city town. Not only the characteristics could have changed, it is even possible that a complete different infrastructure will be used [71]. The communication interface of the mobile must not only be able to adapt to these situations and provide the basic functionally, it must also do it energy efficient in all these situations. At the same time, the Quality of Service guarantees of the various connections should still be supported. In some cases it may be impossible to maintain the QoS guarantees originally promised to the application as the channel degrades, for example when the user moves into a radio shadow where the radio loses physical layer connectivity.

5.5.1 The error model

In any communication system, there have always been errors and the need to deal with them. Wireless networks have a much higher error rate than the normal wired networks. The errors that occur on the physical channel are caused by phenomena such as signal fading, transmission interference, and user mobility.

In characterising the wireless channel, there are two variables of importance. First, there is the Bit Error Rate (BER) – a function of Signal to Noise Ratio (SNR) at the receiver -, and second the burstiness of the errors on the channel. Figure 2 presents a graphical view of packets moving through this channel.

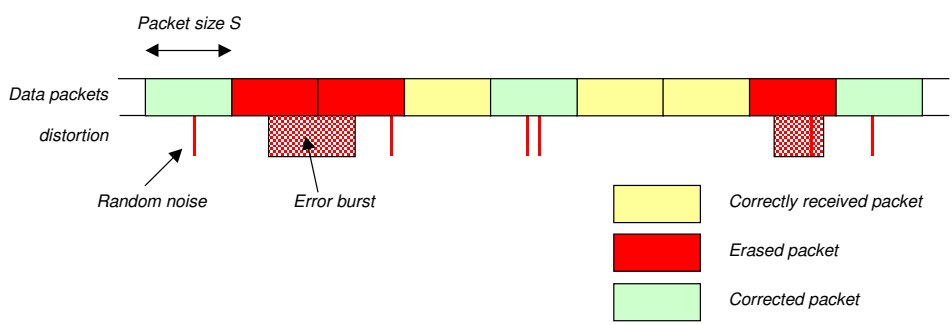


Figure 2: Error characteristics and packet erasures.

This leads to two basic classes of errors: packet erasures and bit corruption errors [21][83]. Error control is applied to handle these errors.

Note that when the bit errors are independent, the packet error rate (*PER*) is related to the size of the packet (*s*) and the bit error rate (*BER*) as

$$PER = 1 - (1 - BER)^s \qquad (1)$$

While this does not take into account the bursty nature of a wireless link, it gives an idea of the influence of the packet length on the error rate of a packet. Even one uncorrected bit error inside a packet will result in the loss of that packet. Each lost packet directly results in wasted energy consumption, wasted bandwidth, and in time spent. This loss

might also result in the additional signalling overhead of an ARQ protocol [45]. Because of this, it is important to simultaneously adapt the error control mechanism when the packet size is maximised to minimise the number of transitions. In Section 5.7.2 we will analyse the effects of packet length and energy efficiency in more detail.

5.5.2 Error-control alternatives

There are a large variety of error-control strategies, each with its own advantages and disadvantages in terms of latency, throughput, and energy efficiency. Basically there are two methods of dealing with errors: retransmission (Automatic Repeat reQuest (ARQ) and Forward Error Correction (FEC). Hybrids of these two also exist. Within each category, there are numerous options. Computer communication generally implements a reliable data transfer using either methods or a combination of them at different levels in the communication protocol stack. Turning a poor reliability channel into one with moderate reliability is best done within the physical layer utilising signal space or binary coding techniques with soft decoding. FEC is mainly used at the data link layer to reduce the impact of errors in the wireless connection. In most cases, these codes provide less than perfect protection and some amount of residual errors pass through. The upper level protocol layers employ various block error detection and retransmission schemes (see e.g. [67][39]).

- With *FEC* redundancy bits are attached to a packet that allow the receiver to correct errors which may occur. In principle, FEC incurs a fixed overhead for every packet, irrespective of the channel conditions. This implies a reduction of the achievable data rate and causes additional delay. When the channel is good, we still pay this overhead. Areas of applications that can benefit in particular from error-correction mechanisms are *multicast applications* [74][65][61]. Even if the QoS requirement is not that demanding, insuring the QoS for all receiving applications is difficult with retransmission techniques since multiple receivers can experience losses on different packets. Individual repairs are not only extremely expensive, they also do not scale well to the number of receivers. Reducing the amount of feedback by the use of forward error correction, leads to a simple, scalable and energy-efficient protocol.

  Several studies have shown that adaptive packet sizing and FEC can significantly increase the throughput of a wireless LAN, using relative simple adaptation policies (e.g. [21][24][60]). Note that, due to the burst errors, FEC block codes might require interleaving to spread the errors over the whole packet. However, burst error events on the indoor wireless channel caused by slow-moving interference may last for hundreds of milliseconds, rendering interleaving infeasible for time-critical (delay and jitter) applications [29].

- Using *ARQ*, feedback is propagated in the reverse direction to inform the sender of the status of packets sent. The use of ARQ results in an even more significant increase of delay and delay variations than FEC [66]. The retransmission requires additional buffering at the transmitter and receiver. A large penalty is paid in waiting for and carrying out the retransmission of the packet. This can be

unacceptable for systems where Quality of Service (QoS) provisioning is a major concern, e.g. in wireless ATM systems. These communications will include video, audio, images, and bulk data transfer, each with their own specific parameter settings regarding for example jitter, delay, reliability, and throughput [19]. Solutions to provide a predictable delay at the medium access control layer by reserving bandwidth for retransmission are possible [27], but spoil bandwidth.

ARQ schemes will perform well when the channel is good, since retransmissions will be rare, but perform poorly when channel conditions degrade since much effort is spent in retransmitting packets. Another often ignored side effect in ARQ schemes is that the round-trip-delay of a request-acknowledge can also cause the receiver to be waiting for the acknowledge with the receiver turned 'on', and thus wasting energy.

- *Hybrids* do not have to transmit with maximum FEC redundancy to deal with the worst possible channel. Under nominal channel conditions, the FEC will be sufficient, while under poor channel conditions ARQ will be used. Although more efficient than the pure categories, a hybrid system is still a rigid one since certain channel conditions are assumed.
- *Adaptive error control* allows the error-control strategy to vary as the channel conditions vary. The error control can be FEC, ARQ, or a hybrid. The wireless channel quality is a function of the distance of user from base station, local and average fading conditions, interference variations, and other factors. Furthermore, in packet data systems the bursty nature of data traffic also causes rapid changes in interference characteristics. In a wireless channel, link adaptations should occur frequently because of the rapid changes in signal and interference environment. In such a dynamic environment it is likely that any of the previous schemes is not optimal in terms of energy efficiency all the time. Adaptive error control seems likely a source of efficiency gain.

  Adaptive error control can be added fairly easily to a MAC protocol and link layer protocols. First of all, the adaptive error-control techniques have to be present in the sender and receiver.

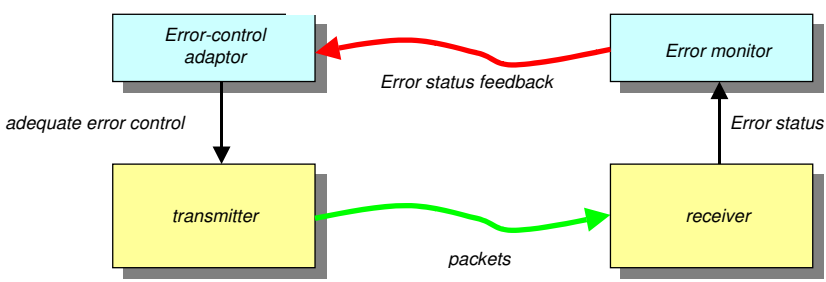


Figure 3: Feedback loop for adaptive error control.

Secondly, a *feedback loop* is required to allow the transmitter to adapt the error coding according to the error rate observed at the receiver. Normally, such

information consists of parameters such as mean carrier-to-interference ratio (C/I) or signal-to-noise ratio (SNR), standard deviation of SNR channel impulse response characterisation, bit error statistics (mean and standard deviation), and packet error rate. The required feedback loop limits the responsiveness to the wireless link conditions. Additional information can be gathered with a technique that performs link adaptation in an implicit manner by purely relying on acknowledgement (ACK/NACK) information from the radio link layer.

Depending on the application, the adaptation might not need to be done frequently. If, for example, the application is an error-resilient compression algorithm that when channel distortion occurs, its effects will be a gradual degradation of video quality, then the best possible quality will be maintained at all BERs ([3][56][76][77]).

A more detailed comparison of the performance of ARQ and FEC techniques has been made by many researchers (e.g. [44],[66] and [85]), and is not part of our research.

The choice of energy-efficient error-control strategy is a strong function of QoS parameters, channel quality, and packet size [44]. Since different connections do not have the same requirements concerning e.g. cell loss rate and cell transfer delay, different error-control schemes must be applied for different connection types. The design goal of an error-control system is to find optimum output parameters for a given set of input parameters. Input parameters are e.g. channel BER or maximum delay. Examples of output parameters are FEC code rate and retransmission limit. The optimum might be defined as maximum throughput, minimum delay, or minimum energy consumption, depending on the service class (or QoS) of a connection. Real-time traffic will prefer minimum delay, while most traditional data services will prefer a maximum throughput solution. All solutions in a mobile environment should strive for minimal energy consumption.

5.5.3 Local versus end-to-end error-control

The networking community has explored a wide spectrum of solutions to deal with the wireless error environment. They range from local solutions that decrease the error rate observed by upper layer protocols or applications, to transport protocol modifications and proxies inside the network that modify the behaviour of the higher level protocols [23].

Addressing link errors near the site of their occurrence seems intuitively attractive for several reasons.

- It is most efficient that the error-correcting techniques to be tightly coupled to the transmission environment because they understand their particular characteristics [31].
- Entities on the link are likely to be able to respond more quickly to changes in the error environment, so that parameters such as FEC redundancy and packet length are varied with short time.
- Performing FEC on an end-to-end basis implies codes that deal with a variety of different loss and corruption mechanisms, even on one connection. In practice this

implies that different codes have to be concatenated to deal with every possible circumstance, and the resulting multiple layers of redundancy would be carried by every link with a resultant traffic and energy consumption penalty [30]. End-to-end error control requires sufficient redundancy for the worst case link, resulting in a rate penalty on links with less impairment. Local error control requires only extra bandwidth where it is truly needed.

- Practically, deploying a new wireless link protocol on only those links that need it is easier than modifying code on all machines. Application-level proxies address this problem to some extend, but they are currently constrained to running end systems, whereas local error control can operate on exactly the links that require it [23].

Despite these attractions, trying to solve too much locally can lead to other problems. In the case of local error control for wireless links, there are at least three dangers [23].

- Local error control alters the characteristics of the network, which can confuse higher layer protocols. For example, local retransmission could result in packet reordering or in large fluctuations of the round-trip time, either of which could trigger TCP timeouts and retransmissions.
- Both local and end-to-end error control may respond to the same events, possibly resulting in undesirable interactions, causing inefficiencies and potentially even instability.
- End-to-end control has potentially better knowledge of the quality requirements of the connection. For example, a given data packet may bear information with a limited useful lifetime (e.g. multimedia video traffic), so error control that will cause the delay to exceed a certain value is wasted effort. It might be better to drop a corrupt video packet, than to retransmit it, since retransmission may make the next packet late.

Given the significant advantages of local error control, we will pursue a local approach for the lower layers of the communication protocol stack. However, while we propose that the primary responsibility for error control fall to the local network, there is no reason to dogmatically preclude the involvement of higher level protocols. In particular, the application should be able to indicate to the local network the type of its traffic and the QoS expectations.

The lowest level solution to local error-control is by using hardware error-control techniques such as adaptive codecs and multi-rate modems. While these are attractive in terms of simplicity, they may leave a noticeable residual error rate. In addition, while they reduce the average error rate, they cannot typically differentiate between traffic of different connections. A MAC and link-layer approach that is able to apply error control on a per-traffic basis is an attractive alternative. These protocols, such as IEEE 802.11 [41], MASCARA [5], and E^2MaC [33], are − or can be made − traffic-aware (rather than protocol-aware) by tailoring the level of error control to the nature of the traffic (e.g. bounding retransmission for packets with a limited lifetime).

5.5.4 Related work

Error control is an area in which much research has been performed. Books on error control, such as [46], cover the basic FEC and ARQ schemes well. More recently, much work has focussed on error control in wireless channels. Some error-control scheme alternatives and their implications have been discussed in Section 5.5.1.

Adaptive error-control is mainly used to improve the throughput on a wireless link [21][24]. Schuler presents in [66] some considerations on the optimisation and adaptation of FEC and ARQ algorithms with focus on wireless ATM developments. The optimisation, with respect to the target bit error rate and the mapping of the wireless connection quality to the ATM QoS concept, is discussed in detail. Eckhardt and Steenkiste [23] argue and demonstrate that protocol-independent link-level local error control can achieve high communication efficiency even in a highly variable error environment, that adaptation is important to achieve this efficiency, and that inter-layer coexistence is achievable.

In [82] it has been shown that classic ARQ strategies could lead to a considerable waste of energy (due to several reasons: more communication overhead, more transitions, longer communication time, etc.). They propose an adaptive scheme, which slows down the transmission rate when the channel is impaired. This scheme saves energy without a significant loss in throughput. Several other solutions have been proposed [3][55], but the focus with these solutions is mainly on increasing the throughput, and not on preserving QoS and energy efficiency.

Classic ARQ protocols overcome errors by re-transmitting the erroneously received packet, regardless of the state of the channel. Although in this way these retransmission schemes *maximise the performance* – as soon as the channel is good again, packets are received with minimal delay – the consequence is that they expend energy [82]. When the tolerable delay is large enough, ARQ outperforms error-correction mechanisms, since the residual error probability tends to zero in ARQ with a much better energy efficiency than error correction methods [85].

Most relevant work that relates the error coding strategy to energy consumption is by Zorzi and Lettieri. Zorzi describes in [82] and [85] an adaptive probing ARQ strategy that slows down the transmission rate when the channel is impaired without a significant loss in throughput. A modified scheme is also analysed, which yields slightly better performance, but requires some additional complexity. Lettieri ([45]) describes how energy efficiency in the wireless data link can be enhanced via adaptive frame length control in concert with adaptive error control based on hybrid FEC and ARQ. The length and error coding of the frame going over the air and the retransmission protocol are selected for each application stream based on QoS requirements and continually adapted as a function of varying radio channel conditions.

All error-control techniques introduce latency, a problem that is more prominent with limited bandwidth. This poses the problem that low latency (for interactivity) and high reliability (for subjective quality) are fundamentally incompatible under high traffic conditions [29]. Some multimedia applications might, however, be able to use the possibly corrupt packet. With *multiple-delivery transport service* multiple possibly

corrupt but increasingly reliable versions of a packet are delivered to the receiving application [29]. The application has the option of taking advantage of the earlier arriving corrupt packet to lower the perceived latency, but eventually replaces them with the asymptotically reliable version.

In the concept of *incremental redundancy* (IR) [60], redundant data, for the purpose of error correction, is transmitted only when previously transmitted packets of information are received and acknowledged to be in error. The redundant packet is combined with the previously received (errored) information packets in order to facilitate error correction decoding. If there is a decoding failure, more redundancy is transmitted. The penalty paid for increased robustness and higher throughput is additional receiver memory and higher delay.

5.6 Energy-efficient wireless network design

This section describes the basic principles and mechanisms of the network interface architecture implemented in our research, and our energy efficient medium access control protocol for wireless links, called E^2MaC. The protocol and the architecture are targeted to a system in which quality of service (including the incurring energy consumption) plays a crucial role. The ability to integrate diverse functions of a system on the same chip provides the challenge and opportunity to do system architecture design and optimisations across diverse system layers and functions [73].

As mentioned before, two key requirements in mobile multimedia systems are:

- *Requirement 1*: the need to maintain quality of service in a mobile environment and,
- *Requirement 2*: the need to use limited battery resources available efficiently.

We have tackled these problem by making the system highly adaptive and by using energy saving techniques through all layers of the system. Adaptations to the dynamic nature of wireless networks are necessary to achieve an acceptable quality of service. It is not sufficient to adapt just one function, but it requires adaptation in several functions of the system, including radio, medium access protocols, error control, network protocols, codecs, and applications. Adaptation is also a key to enhancing battery life. Current research on several aspects of wireless networks (like error control, frame-length, access scheduling) indicate that continually adapting to the current condition of the wireless link have a big impact on the energy-efficiency of the system [13][16][36][45][41]. In our work these existing ideas and several new ideas have been combined into the design of adaptive energy efficient medium access protocols, communication protocol decomposition, and network interface architecture [37] using the previously mentioned principles P1, P2, P3, and P4.

5.6.1 System overview

The goals of low energy consumption and the required support for multiple traffic types lead to a system that is based on reservation and scheduling strategies. The wireless ATM network is composed of several base-stations that each handle a single radio cell[2] possibly covering several mobile stations. We consider an office environment in which the cells are small and have the size of one or several rooms. This not only saves energy because the transmitters can be low powered, it also provides a high aggregate bandwidth since it needs to be shared with only few mobiles. The backbone of the base-stations is a wired ATM network. In order to avoid a serious mismatch between the wired and wireless networks, the wireless network part should offer similar services as the wired network.

The general theme that influences many aspects of the design of the data link protocol is adaptability and flexibility. This implies that for each connection a different set of parameters concerning scheduling, flow control and error control should be applied.

We do not intend to handle all aspects of a full-blown wireless ATM network that provides all possible services. We adapt some features of ATM because they can be used quite well for our purpose. To implement the full ATM stack would require a large investment in code and hardware. The QoS provisions of ATM fit quite well with the requirements of multimedia traffic. This provides much more possibilities for differentiating various media streams than an often used approach in QoS providing network systems with just two priority levels (real-time versus non-real-time) [17], or even multiple priority levels [33].

However, when adopting ATM in a wireless environment, we need a much more dynamic approach to resource usage. The small size packet structure and small header (in B-ISDN ATM 48 bytes data and 5 bytes header) allows for a simple implementation. Small cells have the benefit of a small scheduling granularity, and hence provide a good control over the quality of a connection. The fixed size also allows a simple implementation of a flexible buffering mechanism that can be adapted to the QoS of a connection. Also a flexible error control mechanism has advantage when these cells are adopted. When the base station is connected via a wired ATM network, then the required processing and adaptation can be minimal since they use the same cell structure and the same quality characteristics.

The system contains several QoS managers. Applications might need resources under control of several QoS managers. The QoS managers then need to communicate with each other via a wired network and wirelessly with applications on mobiles. The key to providing service quality will be the scheduling algorithm executed by the QoS manager that is typically located at the base-station. This QoS manager tries to find a (near) optimal 'schedule' that satisfies the wishes of all applications.

---

[2] Note that the term cell here is different from the term cell used to denote the basic transmission unit in ATM.

Each mobile can have multiple unidirectional connections with different Quality of Service requirements. Five service categories have been defined under ATM (see Section 5.4.1): constant bit rate (CBR), real time VBR, non-real time VBR, unspecified bit rate (UBR), and available bit rate (ABR). The scheduler gives priority to these categories in the same order as listed here possibly using different scheduling algorithms for each category.

The base-station receives transmission requests from the mobiles. The base-station controls access on the wireless channel based on these requests by dividing bandwidth into *transmission slots*. The key to providing QoS for these connections will be the scheduling algorithm that assigns the bandwidth. The premise is that the base-station has virtually no processing and energy limitations, and will perform actions in courtesy of the mobile. The main principles are (using the principles *P*1 to *P4* of Section 5.3): avoid unsuccessful actions by avoiding collisions and by providing provisions for adaptive error control, minimise the number of transitions by scheduling traffic in larger packets, synchronise the mobile and the base-station which allows the mobile to power-on precisely when needed, and migrate as much as possible work to the base-station.

The layers of the communication protocol are summarised in Figure 4. The column in the middle represents the layers used by the base-station; the columns on the left and right represents the layers used by the mobile.

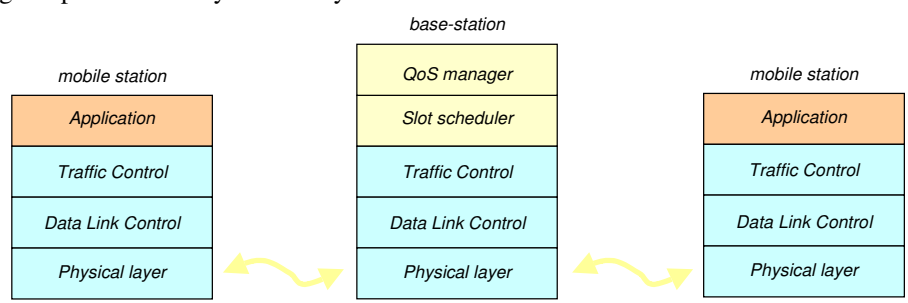


Figure 4: Protocol stack

The lower layers exist in both the mobile and the base station. The *Data link control* manages the data-transfer with the physical layer (using the E^2MaC protocol), and *Traffic control* performs error control and flow control. The base-station contains two additional layers: the *Slot Scheduler* that assigns slots within frames to connections, and the *QoS manager* that establishes, maintains and releases virtual connections.

The definition of the protocol in terms of multiple phases in a frame is similar to other protocols proposed earlier. The E^2MaC protocol goes beyond these protocols by having minimised the energy consumption of the mobile within the QoS requirements of a connection. The features of the protocol are support for multiple traffic types, per-connection flow control and error-control, provision of service quality to individual connections, and energy efficiency consideration.

5.6.2 E²MaC protocol

In the E²MaC protocol the scheduler of the base station is responsible for providing the required QoS for the connections on the wireless link and tries to minimise the amount of energy spend by the mobile. It uses the four main principles *P1* to *P4*. The protocol is able to provide near-optimal energy efficiency (i.e. energy is spent for the actual transfer only) for a mobile within the constraints of the QoS of all connections.

The protocol uses fixed-length frames of multiple slots. Each slot has a fixed size. A slot determines the time-frame in which data can be received or transmitted. The base-station and mobile are completely synchronised (the time unit is a slot), which allows the mobile to power-on precisely when needed. The base-station controls the traffic for all mobiles in range of the cell and broadcasts the schedule to the mobiles.

E²MaC frame structure

The frame is divided in time-slots that can have three basic types: *traffic control*, *registration request*, and *data*. Only the traffic control type has a fixed position at the start of the frame[3]. The number of slots needed for traffic control depends on the size of the frame and is thus implementation dependent. Typically one slot is sufficient. The other types are dynamic, have no fixed size and can be anywhere in the rest of the frame. The base-station controls the traffic for all mobiles in range of the cell and broadcasts the schedule in the traffic control slot. Only new connections may encounter collisions in the registration request slots, the traffic control slots and data slots are collision-less.

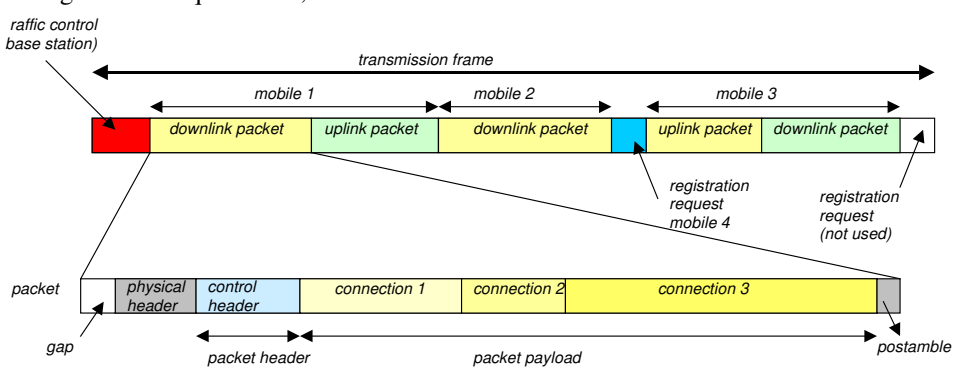


Figure 5: Example of a transmission frame.

The *traffic control slot* (TCS) contains information about the type and direction of each slot in the current frame, and the connection-ID that may use the slot. Since a corrupted traffic control slot can influence the QoS of all connections, these slots are protected with an error correction mechanism. The traffic control contains 1) the schedule of the slots for the connections assigned in the frame (connection-ID, slot number, length), 2)

[3] Implementation restrictions (that cause the time to interpret the traffic control to be significant) might cause the traffic control to be located somewhere in the previous transmission frame.

the connection queue status for all connections, and 3) two fields used for connection set-up for uplink and downlink connections (mobile-ID, connection-ID). This data-structure allows for 15 connections to be registered into a traffic control slot with the size of one ATM cell when we assume that a frame has maximal 256 slots and a mobile-ID is 16 bits. These numbers seem sufficient in a micro-cellular network in which the cells have the size of one or several rooms. To allow mobiles to power down completely for a while, it might be useful to have a timestamp in the traffic control slot as well. A mobile is not required to receive the TCS of each frame. Depending on the QoS of its connections, it might receive the TCS at a lower frequency.

The *registration request slots* are used by a mobile for two purposes: 1) to announce that it wants to connect to the base station, and 2) for connection management. This traffic is contention based. Because no data traffic is carried during this period, back-off values can be kept short [53]. All slots of a frame that are not used for a connection are registration request slots that can be utilised by a mobile to request a new connection or to update the status of the connection queues of the mobile (the buffer status). To be effective, the base station slot scheduler must know the state of each connection to avoid to assign in vain uplink slots to idle connections. The buffer status is generally forwarded to the base station in each uplink packet, but when no uplink connection is available, it can use a registration request slot to transmit a control connection message with the buffer status. Registration requests allow mobiles that have entered the cell to register at the base-station.

Both the mobile and the base-station use the *data slots* to send the actual data.

The overhead introduced in the physical layer can be significant, e.g. for WaveLAN it can be up to *virtual* 58.25 bytes (for guard space (gap in which the silence level is measured), interfacing delay (required to synchronise to the internal slotsync moments), preamble and postamble, see Figure 6). Moreover, with this interface that has a throughput of 2 Mbit/s, a transition time from sleep to idle of 250 µs already takes *virtually* 62.5 bytes (500 bits). The overhead required to power-up is thus already more than the transmission of one ATM cell.

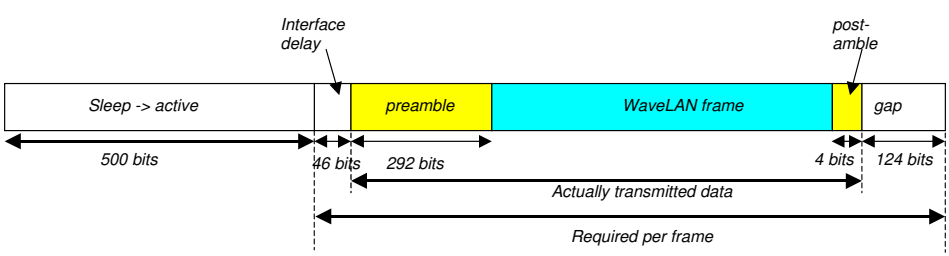


Figure 6: WaveLAN physical layer block format.

This shows that efficient data transmission (in terms of bandwidth utilisation and energy consumption) can only be achieved if the number of ATM cells transmitted is not too small. (In Section 5.7.2 the consequences of the transitions are analysed in more detail.) So, the data cells from one mobile are grouped together as much as possible within the QoS restraints. These cells form a packet that is a sequence of ATM cells possibly for

multiple connections. Each packet is constituted of a header, followed by the payload consisting of ATM cells generated by the same mobile. Control messages that do not need the payload can use the header only. Because in general the transition-overhead between transmit and receive modes is much less than the transition overhead between power down modes, transmission packets and reception packets for one mobile are placed right after each other in the frame. A mechanism in which a frame is divided into an uplink part and a downlink part uses the available bandwidth more efficiently, but requires more transitions for the mobile. More details about the slot scheduling can be found in Section 5.6.4.

The header of a packet contains 1) information about the actual length of the data for each connection, 2) the parameters of the error coding applied for each connection, and 3) flow control information about all transmission and reception queues of the network interface.

5.6.3 QoS manager

The *QoS manager* establishes, maintains and releases wireless connections between the base-station and the mobile and also provides support for handover and mobility services. Applications contact the QoS manager when setting up a connection. The QoS manager will inform the applications when they should adapt their data streams when the QoS of a connection has changed significantly. Figure 7 gives a schematic overview of the service model.

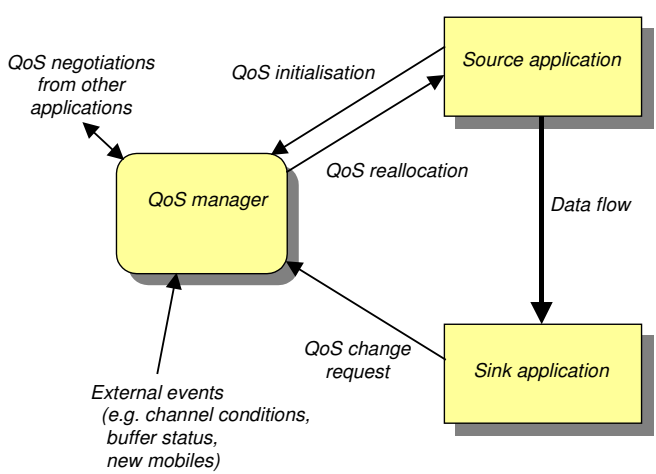


Figure 7: The service model for adaptive applications.

QoS support in wireless networks involves several considerations beyond those addressed in earlier work on conventional wireline networks. Wireless broadband access is subject to sudden variations in bandwidth availability due to the dynamics of the wireless channel and the service demand (e.g. mobiles moving in and out the base

station's coverage area, interactive multimedia connections). In traditional networks based on fixed terminals and high-quality/high capacity links it is feasible to provide 'hard' QoS guarantees to users. However, in the mobile environment, mobility and the need for efficient resource utilisation require the use of a 'soft' QoS model [64].

Multimedia networking requires at least a certain minimum QoS and bandwidth allocation for satisfactory application performance. This minimum QoS requirement has a wide dynamic range depending on the user's quality expectations, application usage modes, and application' tolerance to degradation. In addition, some applications can gracefully adapt to sporadic network congestion while still providing acceptable performance. The soft QoS model is suitable for adaptive multimedia applications capable of gracefully adjusting their performance to variable network conditions. The QoS manager matches the requirements of the application with the capabilities of the network. Figure 7 conceptually illustrates the role of adaptive applications in the QoS model.

The application requests a new connection for a certain *Service Class* that defines the media type (e.g. video, audio, data), interactivity model (e.g. multimedia browsing, videoconference), and various QoS traffic parameters (e.g. required bandwidth, allowable cell loss ratio). The service classes allow multimedia sessions to transparently adapt the quality of the connection when the available resources change marginally without the need to further specify details and without explicit renegotiations.

Network resource allocation is done in two phases. First, the QoS manager checks the availability of resources on the base-stations coverage area at connection setup. The necessary resources are estimated based on the required service. The new connection is accepted if sufficient resources are estimated to be available for the connection to operate within the service contract without affecting the service of other ongoing connections. Otherwise, the connection is refused. Second, while the connection is in progress, dynamic bandwidth allocation is performed to match the requirements of interactive traffic and the available resources. When the available bandwidth changes (because congestion occurs, or the error conditions change drastically), the QoS manager reallocates bandwidth among connections to maintain the service of all ongoing connections within their service contracts. The resulting allocation improves the satisfaction of under-satisfied connections, while maintaining the overall satisfaction of other connections as high as possible. In [64] a bandwidth reallocation algorithm is described that fits well to the QoS model used by the QoS manager.

Connection setup

When a new connection has to be made the service class and the required QoS of the wireless connection is passed to the QoS manager on the base-station. The required QoS is determined by classical parameters like *throughput*, *reliability*, *jitter* and *delay*. The quantitative QoS parameters used by the protocol are:

- The required bandwidth, expressed in the number of data slots required in a frame and the frequency that a connection will use slots in a frame.
- jitter, the allowable variation in delay in a frame.

The Traffic Control uses two additional parameters:

- allowable delay, expressed in number of ATM cells
- reliability, the percentage of cells that may be erased due to buffer overflow or errors

It is the task of the system to translate the original QoS parameters into these MAC level parameters. It thereby can also incorporate the expected error rate of the wireless link. For time-critical traffic it might use an error correcting code [34]. The base-station contains the central scheduler for the traffic of all mobiles in its range. The mobiles send requests for new connections or update information to the base-station. The base-station determines according to the current traffic in the cell whether it can allow the new connection. When the request is granted, the base-station assigns a connection-ID to the new connection and notifies this ID to the mobile in a dedicated field in the traffic control slot. The mobile will then create a queue for that connection.

Connection management

Control messages are exchanged between the functional entities. These messages are used 1) to perform data link layer flow control between the connection queues of the mobile and the base-station, and 2) to manage connections, i.e. to setup new connections, to update the QoS of current connections, or to release connections.

Just like any packet, a control connection packet contains the buffer status of the connection queues. This status is used by the slot scheduler to make a proper schedule.

Each mobile has at least one *control connection*. When a mobile enters a cell it uses a registration request slot to register itself to the base-station. In this slot contentions can occur with requests from other mobiles. If no collision occurred with other mobiles, the base-station receives the request and determines whether it can fulfil the request. If so, it initially assigns one data slot per frame for that connection. In the traffic control slot it indicates (using the ID of the mobile that it has acknowledged) the connection and assigns a connection-ID of the a bi-directional *control connection*. This connection is collision-less and can be used by the mobile to request new connections and acknowledge downlink traffic. The base-station can use this connection to request new downlink connections.

The control connection is scheduled at a rate corresponding to the most stringent requirement of all established connections of the mobile. This requirement can stem from maximal delay, jitter, etc. It will in general be mapped to a deadline time at which a control message or normal data connection of any connection needs to be established by the base station. We will name this interval the *maximum Cell Transfer Delay* (maxCTD).

The mobile can use control messages to change the parameters of existing connections, and request new connections. Possible commands are:

- **release connection**. To release the current connection and free all reservations.
- **sleep (s)**. This informs the base-station that the mobile will sleep for *s* frames. This allows the scheduler to re-assign the slots for other connections during *s* frames. The value of *s* is determined by the requirements of all connections. The connection with the most stringent requirement will in general dictate the value of *s*. A mobile can be forced to send a keep-alive message by indicating a sleep (*s*). In this way the base-station will know when the mobile has left the cell or is turned off, and can free the reserved resources.
- **update QoS**. This message can be used to change the current QoS of a connection.
- **new connection**. New connections can be made using the current control connection. In this way the mobile does not have to compete with other mobiles to access a connection request slot which reduces the occurrence of collisions. The data field is used to indicate the required service class and QoS of the new connection.

5.6.4 Slot scheduler

The notion of QoS over a wireless link has been the focus of much recent research, and several scheduling algorithms have been proposed [19][59][68]. This section describes a framework by which various scheduling mechanisms can be build that incorporates the QoS requirements and uses the four principles of energy efficient design (see Section 5.3)[4].

The *slot scheduler* on the base-station (*Principle P4*) assigns bandwidth and determines the required error coding for each individual connection. The QoS manager provides the service contracts used.

For a proper slot assignment, the slot scheduler needs to know the current state of each connection. For the downlink direction, the scheduler acquires this information directly by monitoring the corresponding queues in the base station. For the uplink direction, this information can be obtained through the implementation of a dedicated protocol, which can be a *polling* scheme or a *contention* scheme. The polling based mechanism, often proposed for its implementation simplicity, requires a polling interval based on *maxCTD*. The polling scheme introduces a maximal delay equal to the polling interval. The contention based scheme has a delay of one frame (when no contention occurs). Polling and contention are quite different also in the utilisation of the channel bandwidth. The polling mechanism uses a number of slots that linearly increases with the number of mobiles with a slope that grows as the required polling interval decreases. The number of contention slots is practically independent from *maxCTD*. The advantage of the polling scheme is that it can give better guarantees since no contention can occur.

[4] Up to now we have just implemented a simple scheduling algorithm [42].

In E^{2}MaC we use a combination of both polling and contention. In E^{2}MaC all packets include the buffer status of the connection queues. Thus, when there are enough uplink connections (either normal data packets or control connection packets), then the slot scheduler will receive the connection queue status frequently. If this is not sufficient (for example because a connection queue receives more data than anticipated), then a contention slot can be used to transfer a recent buffer status update to the slot scheduler using a control packet.

A schedule is broadcast to all mobiles so that they know when they should transmit or receive data (*Principle P1* and *P3*). In composing this traffic control, the slot scheduler takes into account: the state of the downlink and uplink queues, and the radio link conditions per connection. The slot scheduler is designed to preserve the admitted connections as much as possible within the negotiated connection QoS parameters. It schedules all traffic according to the QoS requirements and tries to minimise the number of transitions the mobile has to make (*Principle P2*). It schedules the traffic of a mobile such that all downlink and uplink connections are grouped into packets taking into account the limitations imposed by the QoS of the connections. The grouping of traffic in larger packets is also used by other protocols to increase the efficiency (both in terms of bandwidth and energy consumption) of the protocol. In general there are three phases: uplink phase, downlink phase, and reservation phase. In the downlink phase the base station transmits data to the mobiles, and in the uplink phase the mobiles transmit data to the base station. In the reservation phase mobiles can request new connections. We will refer to this mechanism as *phase grouping*.

In our protocol we have in principle similar phases, but these are not grouped together in a frame according to the phase, but are grouped together according to the mobile involved. In our protocol we thus group the uplink and downlink phase *of one mobile*. We will refer to this mechanism as *mobile grouping*.

Figure 8 shows the two grouping strategies. In mobile grouping the uplink and downlink packets for a mobile are grouped sequentially (if possible) so that the mobile can power down longer and make minimal transitions between power modes. The power consumption of the WaveLAN modem when transmitting is typical 1675 mW, 1425 mW when receiving, and 80 mW when in sleep mode [80]. Increasing the sleep time period of the radio thus significantly improves the energy efficiency of the wireless network. Moreover, due to the large power-transition times, this mechanism might give the mobile enough time to enter a power-down mode at all. This is shown in the figure where Mobile 2 with phase grouping cannot enter sleep mode after reception of the downlink packet, but is forced to idling[5]. Because the operating modes of phase grouping for a mobile are spread in the frame, the power-mode transition times T_{sleep} to enter sleep mode, and $T_{wake\text{-}up}$ to wake from sleep mode limits the time a mobile can stay in sleep mode.

[5] A power-optimised network interface could stop receiving the downlink packet after it has received data for mobile 2, and thus also enter sleep mode.

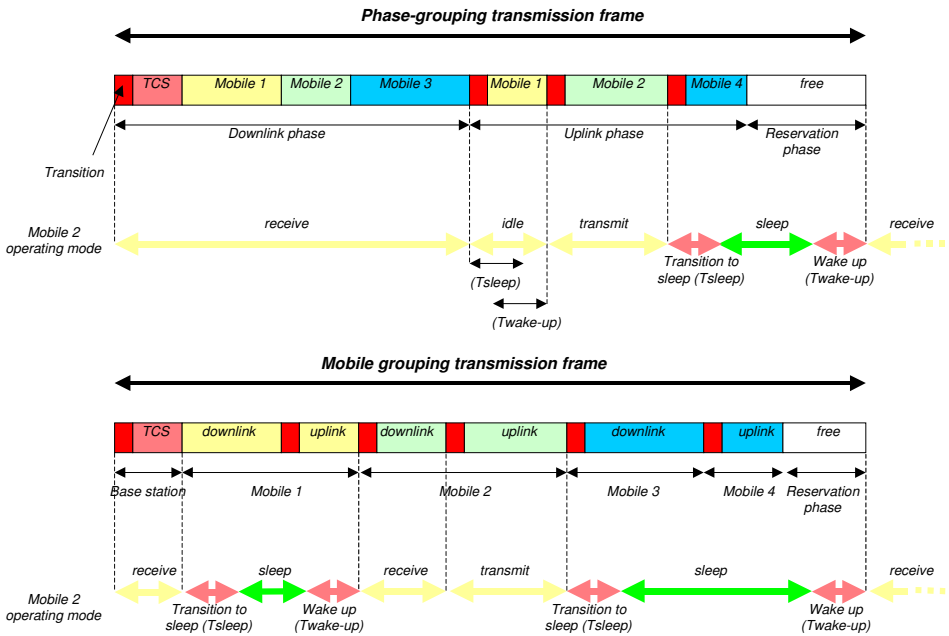


Figure 8: Grouping strategies in a transmission frame.

Notice that in the mobile grouping strategy there is more transition overhead (i.e. one transition per mobile) since the base station does not transmit its data to the mobiles in one packet during the downlink phase of phase grouping. The transition overhead consists of guard space (gap), interfacing delay, preamble, and postamble (see for an example Figure 6). The transition overhead involved with each transmission packet is the reason that the available bandwidth of mobile grouping is less than the available bandwidth in phase grouping. However, since the traffic of a mobile is grouped, the mobile can enter a low-power mode (sleep) for a longer time. In fact, with phase grouping, the mobile is in general forced to receive the complete downlink packet, and will ignore the data not destined for the mobile. The consequences of using mobile grouping on the channel efficiency and the energy consumption is analysed in detail in Section 5.7.2.

If the QoS of a connection allows jitter (like non-real-time bulk data transfer), then the scheduler has more flexibility to group the traffic. When a mobile requests a connection and indicates that it does not allow any jitter, then the scheduler is forced to assign the same data slots in each frame for that connection. In this case, only at connection setup, the scheduler is free to assign the slots. In this way the mobile can minimise its energy consumption, since it knows precisely when it is allowed to transmit data, or when it can expect data. It does not even need to listen to the traffic control. Only the drift of the clock might force the mobile once in a while to synchronise with the base-station.

The slot scheduler maintains two tables: a *request table* and a *slot schedule table*. The request table maintains several aspects of the current connections handled by the base station (like the connection type, the connection queue size and status, the error state of the channel with mobile, the assigned bandwidth, the requested reliability). The slot schedule table reflects the assigned number of slots to connections, and the error coding to be applied. This table is essentially broadcast as Traffic Control Slot (TCS) to the mobiles.

These two tables are used by the QoS manager and slot scheduler to assign bandwidth to connections. Since these entities are implemented as software modules on the base-station, their implementation can be adapted easily to other scheduling policies if needed.

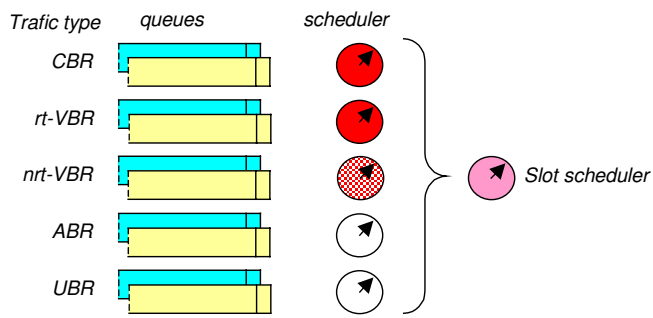


Figure 9: Scheduling per traffic type.

Each ATM service class is assigned a priority, from high to low: CBR, rt-VBR, nrt-VBR, ABR, and UBR. The scheduler gives high priority to CBR and VBR traffic. These traffic sources can reserve bandwidth that the scheduler will try to satisfy. CBR traffic is assigned a maximum bandwidth. If this is not used, the bandwidth will be used by other connections. VBR traffic (both real-time and non-real-time) bandwidth is assigned according to the current traffic flow, up to a specified (average) maximum. The bandwidth adjustment is depending on the current traffic load and the traffic generated by the VBR source. The reservation is updated dynamically in each frame. ABR and UBR traffic, on the other hand, is treated with lower priority and without reservation. Within the same traffic type, the different connections are treated using a scheduling scheme that incorporates the specific requirements of the traffic type (see Figure 9). Real-time traffic requires a fair queuing algorithm. Non-real time traffic can use a more simple scheduling like a round robin mechanism [59].

Dealing with errors

The slot-scheduler dynamically adapts error coding and scheduling to the current conditions in the cell. The error coding required for a specific connection is determined by the error rate observed at the receiver and the required quality of the connection. The slot scheduler retrieves the monitored channel status via a backward connection. It indicates to the network interface which error coding scheme to use. The slot scheduler

has to dynamically adapt its schedule when 1) connections are added or removed, 2) connections change their QoS requirements, and 3) the channel between mobile and base station has significant change in error condition.

The scheduler further tries to avoid periods of bad error conditions by not scheduling non-time critical traffic during these periods. Hard-real time traffic (CBR and rt-VBR) remains scheduled, although it has a higher chance of being corrupted. Note that the error conditions perceived by each mobile in a cell may differ. Since the base-station keeps track of the error conditions per connection, it can give mobiles in better conditions more bandwidth. This can lead to a higher average rate on the channel, due to the introduced dependency between connections and channel quality [16]. In Section 5.6.7 we will give more details on this adaptive error control.

5.6.5 Buffer status coding and flow control

Each connection has its own connection queues with customised flow control. Flow control is needed to prevent buffer overflow. ATM cells of a connection on which the maximum allowed delay is exceeded, for example due to bad error conditions, will be discarded by Traffic Control.

The connection queues of the connections can have a different size and replacement policy. The slot scheduler takes this into account in determining the schedule. The scheduler will be able to assign slots most effectively if it has an accurate notion of the transmission buffer status of each mobile.

A coding associates the number of cells N in the queue to a number of bits that represent the status. The coding of N in a number of bits determines the accuracy. There is a trade-off between the information accuracy and the cost of the information in terms of number of bits to be transmitted. The slot scheduler uses the status information to assign up to I slots to that connection. There are several alternatives for queue size coding.

A *linear coding* associates I with the number of cells in the transmission queues. The maximum number of cells (M) that can be indicated with linear coding is determined by the number of bits C used for the coding ($M=2^C-1$). At the base station, the scheduler can assign up to N cell-sized slots to the requesting connection using the relation $I=\min(M,N)$.

The simplest implementation of linear coding requires just one bit to code the connection queue status consists of a stop/run mechanism. The flow control information is also used by the slot-scheduler to assign slots for connections. If a transmission queue indicates **stop**, then this means that the queue is empty and does not need slots. If it indicates **run**, then the queue contains data and it needs slots. If a reception queue indicates **stop**, then it means that it cannot accept more data. A **run** on a reception queue means that it will accept more data. However, with this simple mechanism the scheduler does not know the buffer occupancy. Therefore, it either needs a threshold of multiple cells (and consequently introduces a delay), or the scheduler might assign too much bandwidth for a connection.

Adding more bits to the coding relieves this problem. For example, when four bits are used to code the buffer occupation, then the scheduler can assign slots using I=min(15,N). Since it is accurate, it minimises the number of assigned and unused slots. However, the effect of this algorithm is that – because of the upper-bound M=15 – this coding tends to reduce the differences among the connections. Therefore, it penalises connections with congested buffers.

A logarithmic coding introduces some inaccuracy, but allows a better representation of the buffer occupancy. The coding uses the following relation:

$$I = \begin{cases} 0 & N=0 \\ 1 + [\log_2(N)] & 1 \le N \le 2^{M-1} \\ M & N > 2^{M-1} \end{cases} \qquad (2)$$

Although this coding also has an upper-bound, it has a much larger range. Therefore, the scheduler can reveal connections with congested buffers.

5.6.6 The architecture of an energy efficient and adaptive network interface

One of the functional modules of a *Mobile Digital Companion* (MDC) is the network module. This module provides the interface between the external world and the different modules of the MDC. The processor on the MDC is responsible for the establishment of the connections between the modules, but also negotiates with the external infrastructure about the QoS of the connections between network module and the modules that are at the end-point of connections. Once a connection between modules is established, they autonomously communicate with each other in the Companion.

On the Network Module the *Data link control* manages the data-transfer with the physical layer, and *Traffic control* performs error control and flow control. Figure 10 depicts the basic blocks of the architecture of the Network Module. The number of connection queues is dynamic and the figure is just an example.

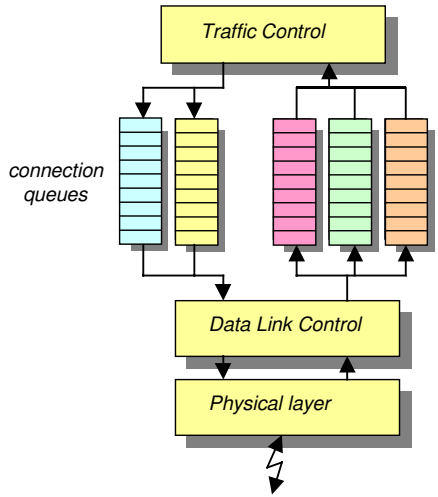


Figure 10: The network interface architecture.

The *Data Link Control (DLC)* performs the traffic allocation of data in the transmission queues. The actual admission decision of connections is made by the QoS management, which informs the Data Link Control using a traffic control packet (either transmitted over the air for the mobile or internally for the base-station). Data Link Control regulates the flow of ATM cells between the physical layer and a local *buffer*. The buffer is organised in such a way that it has a small queue for each connection. This buffer is only meant to store ATM cells for a short time, just enough to implement an effective error control mechanism. When the Data Link Control has to transmit data for a certain connection, it forwards the ATM cells from the transmission queue to the physical layer. On reception it will receive the ATM cells and store them in the queue assigned for that connection. The Data Link Control performs error detection on each ATM cell. The overhead required for error control will be fixed, so that the slot size will not vary.

The *Traffic Control (TC)* controls the flow of data from the connection queues to the corresponding end-points and applies an adaptive error control scheme that operates on individual virtual connections. The choice of an energy efficient error control strategy is a function of QoS parameters, radio channel quality, and packet length. Therefore the architecture of the network interface uses a dynamic error control adapted to these parameters. Each individual connection may use error control schemes that are both adaptive and customised. The selection of the error control scheme and the required size of the queue depend on the QoS constraints imposed on each connection, such as delay constraints or loss-less transfer constraints. This avoids applying error control overhead to connections that do not need it, and allows the possibility to apply it selectively to match the required QoS and the conditions of the radio link. The error control will be based on adaptive error *correcting* techniques. Although well designed retransmission schemes can be energy efficient, they are much more complex to implement (they require a protocol with control messages, sequence numbers, retry counters, etc.) and

can introduce intolerable low performance in delay, jitter and bandwidth to fulfil the required QoS of the connection [36]. The redundant data needed to implement the error correction, will be multiples of ATM cells, so that they fit well in a transmission frame. Status information about the channel conditions and the rate of not-correctable errors are fed-back to the Slot Scheduler at the base station. The Slot Scheduler will try to match the radio conditions to the required fault tolerance, and adapt the required error code and required bandwidth accordingly.

5.6.7 Adaptive error control

A wireless channel quality is dynamic because of the rapid changes in signal and interference environment. The wireless channel quality is a function of the distance of user from base station, local and average fading conditions, interference variations, and other factors. Furthermore, in packet data systems the bursty nature of data traffic also causes rapid changes in interference characteristics.

Due to the dynamic nature of wireless networks, adaptive error control can give significant gains in bandwidth and energy efficiency (see Section 5.5.2). The input parameters for an adaptive error-control system can be classified into two main groups: requirements by the upper protocol layers and momentary transmission quality. Adaptation of the error control can be influenced by three considerations [66]:

1. The FEC redundancy can be adapted to the channel bit error rate and induced energy consumption [36]. The error control system has to find a balance between the added redundancy and the bit error rate and energy consumption.
2. The error control algorithm can be adapted to the required quality. For a wireless connection that tolerates a specified cell loss rate, the error control parameters can be tailored to just meet the requirements.
3. The performance of various error-correcting methods depends on the actual error statistics of the transmission channel. While the FEC technique is generally more suitable for uniformly distributed bit errors, the ARQ technique is optimal for large error bursts, which can hardly be corrected by FEC.

Both the packet length and the BER determine the packet error rate (PER) according to Equation (1). Thus, adaptation is also required when the slot scheduler adapts its packet size in order to minimise the number of transitions. In fact, the Slot Scheduler and the Traffic Control need to work in concert to optimise the overall frame structure.

In our system the channel status information will be gathered by the receivers and forwarded to the Slot Scheduler at the base station. The scheduler determines then, incorporating the QoS requirements of the individual connections, and the observed error rate the changes that have to be applied.

Error control can be applied at multiple layers in the communication protocol stack (see Section 5.5). In our system we apply different error-control techniques at the various layers of the protocol stack. Figure 11 shows the error protocol stack of our system.

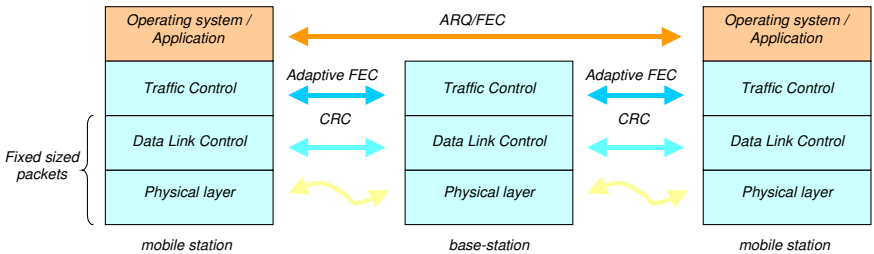


Figure 11: Error control protocol stack.

We do not design the physical layer, but concentrate on the higher layers. At the physical layer we assume that there will be some error correction. This, however, provides less than perfect protection and some amount of residual errors pass through.

At the Data Link Layer we use the E^2MaC protocol. An essential property of this MAC protocol is that it uses fixed-length frames of multiple fixed-length slots. This property allows the network interface to power-on their radio precisely when needed. It also simplifies the design of the data link control. The consequence, however, is that we cannot apply efficiently adaptive error control at this layer, since adaptive error control will change the size of a cell. Depending on the quality of the radio device that is being used, the Data Link Control Layer can use a fixed Forward Error Correction to reduce the number of corrupted cells that were caused by random noise. In our current implementation we only added a one byte CRC to each cell. This CRC is used by the Traffic Control to detect corrupted cells.

Traffic Control is able to apply adaptive error control since it operates with multiples of cells. The error correction mechanism then operates on relative large blocks. Any block error correction mechanism could be used. Generally, block codes such as Bose, Chaudhuri and Hockuenghem (BCH) and Reed-Solomon codes require a decoder capable of performing arithmetic operations in finite fields [51]. A comparison between application-specific integrated circuit (ASIC), FPGA, and digital signal processing (DSP) implementations of the decoder shows that the performance of FPGA-based designs lean more toward that of ASICs, but retain flexibility more like DSPs [11][28]. Unfortunately, good VLSI designs for codes using BCH or Reed-Solomon codes do not map well to FPGAs [4]. A code that does not require finite-field arithmetic, but basically only exclusive-OR operations, is the EVENODD code [8]. The EVENODD code was originally designed for a system of redundant disks (RAID). We have studied the EVENODD error correcting mechanism, and compared it with Reed-Solomon in Appendix A.

In the Traffic Control of the network interface we monitor the condition of the wireless channel of the receiver on three ways. The first method is to monitor the number of corrupted cells using the CRC of each cell. Second, we monitor the rate of corrupted cells that the Traffic Control was not able to correct. The last method is to use the information that is provided directly from the radio hardware. The measured channel condition is returned to the transmitter such that the adaptive mechanisms there can

make a determination of how to format outgoing packets. The status information that is gathered from these methods is forwarded to the Slot Scheduler at the base station (either using a special field in each uplink packet if the status originates from a mobile, or via an internal connection if the status originates from the base station). The Slot Scheduler can then decide to adapt the error control and simultaneously adapt the assigned bandwidth of a connection to the required fault-tolerance. The modification of the error-control parameters needs to be done synchronously at the base station and the mobile. The slot scheduler therefore indicates the error coding that should be applied for a connection in the traffic control slot that is transmitted in each frame.

Depending on the application, the adaptation might not need to be done frequently. If for example the application is an error-resilient compression algorithm that when channel distortion occurs, its effects will be a gradual degradation of video quality, then the best possible quality will be maintained at all BERs [56].

Note that with adaptive error control the energy efficiency is increased, but it cannot guarantee a reliable connection. Higher level protocols in the operating system or in the application are needed to ascertain this, if required. End-to-end error control has potentially better knowledge of the quality requirements of the application (see also Section 5.5).

To ensure a reliable operation, a confirmed service for the control protocol might be needed as well. This already indicates that adaptive error control introduces a significant increase in complexity. More research needs to be done to find a feasible implementation with low complexity and high efficiency. Simplifications in which only a minimal set of error-control mechanisms is used might quite well turn out to be the most optimal solution.

Avoiding bad periods – Above these error control adaptations, the slot scheduler can also adapt its scheduling policy to the error conditions of wireless connections to a mobile. The scheduler tries to *avoid periods of bad error conditions* by not scheduling non-time critical traffic during these periods. Note that the size of an error-burst may be up to 100 milliseconds, which will cause on a 2 Mbit/s wireless link that more than 400 ATM sized slots can be affected. Hard-real time traffic remains scheduled, although it has a higher chance of being corrupted. The base station uses this traffic to probe whether the channel is good again. When the mobile has no real-time connections, it will use a statistical backoff period. Note that the error conditions perceived by each mobile in a cell can be different. Since the base station keeps track of the error conditions per connection (and thus also per mobile), it can give other mobiles more bandwidth when these have better conditions. This can lead to a throughput that may even exceed the average rate on the channel, due to the introduced dependence between admitted connections and channel quality [16].

To ensure long-term fairness a special mechanism can be used that gives *credits* to connections that are not scheduled due to their error conditions. If a mobile is in error-state, the slot scheduler then adds credits for the appropriate connections. This credit mechanism is not applied to real-time traffic, since stale packets will be dropped. When

the error state conditions become better, the slot scheduler schedules the aggregate credits to slots for these connections.

5.6.8 Application interface

Multimedia applications typically communicate multiple streams of data with different types and QoS requirements. If multimedia applications want to achieve optimal performance in an efficient way, they must be aware of the characteristics of the wireless link. Simply relying on the underlying operating system software and communication protocols to transparently hide all the peculiarities of a wireless channel compromises energy consumption and achievable QoS.

By providing the application feedback on the communication, the application can take advantage of the peculiarities and the different data streams over the wireless link. The quality of service over the wireless link and the required energy consumption can be optimised by selecting appropriate parameters for the network interface and network protocols, and by adapting the data-streams.

Recent developments on the internet show streaming audio/video players (like RealPlayer [63]) that dynamically change the frame rate when available network bandwidth changes. The application notes these changes implicitly, i.e. the application senses that available bandwidth is too low because it gets data too late. Actually, only scaling down frame rates is automatic. Once an appropriate lower frame rate is chosen it will not be changed back when more bandwidth becomes available, as this cannot be noted implicitly.

When the link status is available to the applications, scaling in both ways becomes possible. As bandwidth can be used only once, it is better to have one authority that divides it instead of having for example two applications that note an increase in bandwidth and both of them start negotiating higher frame rates with the other end of their transmission. In the end this should average out, but a lot of energy will be wasted before changes settle. The operating system seems to be the right place to put the authority.

Although current audio and video codecs may not benefit from the information, the network interface can make notifications of interesting events. Examples are: 1) the bandwidth dropping below a certain level and 2) the latency in transmission of the last x frames being below a certain limit. When 1) is noted, the codec might drop sample rate accordingly or in case of video maybe even switch from color to black and white. In the case that 2) occurs the application could decide to do less buffering, which is more energy efficient.

Once new codecs that allow fast switching of resolution, frame rate, color/black and white become available, mobiles can take advantage of these notifications from the network interface. When mobile power reduction is taken seriously, these news codecs will emerge as chips with a billion transistors allow implementing them.

The system needs some mechanism in the operating system to tie hardware and user applications together. The MOBY DICK Project uses *Inferno* from Lucent Technologies

Bell Labs [20]. In Inferno communicating programs are multithreaded by nature. The mechanism to notify applications of hardware triggers is implemented as an entry point in the namespace of each application through which messages can simply be transferred. Threads block while reading from a channel until the other end writes a message.

To clarify the idea, here is a small code example of a video transmitter that can generate both color frames and black/white frames:

```
....
x:=sys->open(".../connctl",OREAD);     # open x as a control channel
spawn netwatcher(chan x,ref usecolor); # start a netwatcher thread
while not eof(v_in)                    # while not end of video stream
       generateandsendframe(v_in,usecolor); # send video frames
....

void
netwatcher(chan control,ref int usecolor) {
   while (1) {
         msg := <- control;            # read control msg from channel
         "parse message";
         usecolor = 0 or 1 depending on contents of message;
   }
}
```

5.6.9 Implementation

We have implemented a test-bed of the network interface that we can use to experiment with the various techniques and mechanisms for e.g. error control and MAC protocol. It is build with off-the-shelf components to allow a short design cycle.

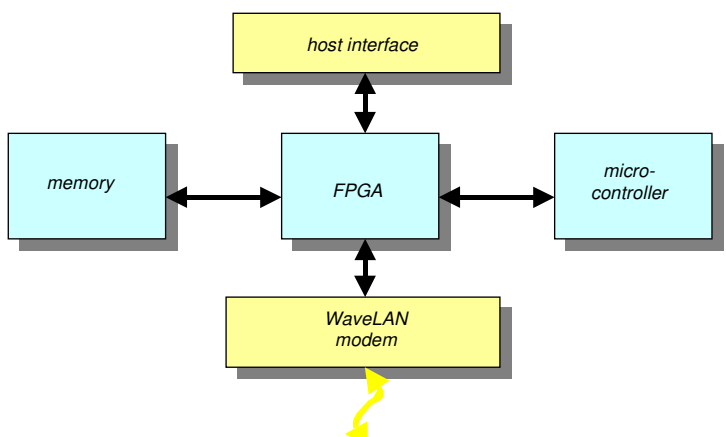


Figure 12: Network interface test-bed architecture.

Figure 12 shows the architecture of the network interface test-bed. The three basic components are:

- *memory* (512 kBytes SRAM) that will be used to implement the connection queues. The amount of buffering that is actually needed depends on the applied error control. Since retransmission is to be implemented by the applications (modules) we just need to have enough buffering to implement an error correction mechanism that is able to correct a small number of cells per connection[6].
- An *FPGA* (Xilinx XC4010) controls the dataflow between the radio and the host and provides basic error detection and error correction functions.
- A *microcontroller* (PIC 16C66) implements the Traffic Control and the Data Link Control (see Section 5.6.6). It controls the functions to be performed by the FPGA, controls the radio modem, and does the power-management. The queues that are stored in the memory are controlled by the microcontroller. It performs the control operations to setup, maintain and release the queues. It collects the status information of the queues and the radio, and transfers this to the QoS manager and slot scheduler in the base-station. Besides these basic functions it further provides miscellaneous operations like initialising the radio modem and gathering status information about the quality of the radio channel.

Figure 13 shows a photograph of the network interface implementation.

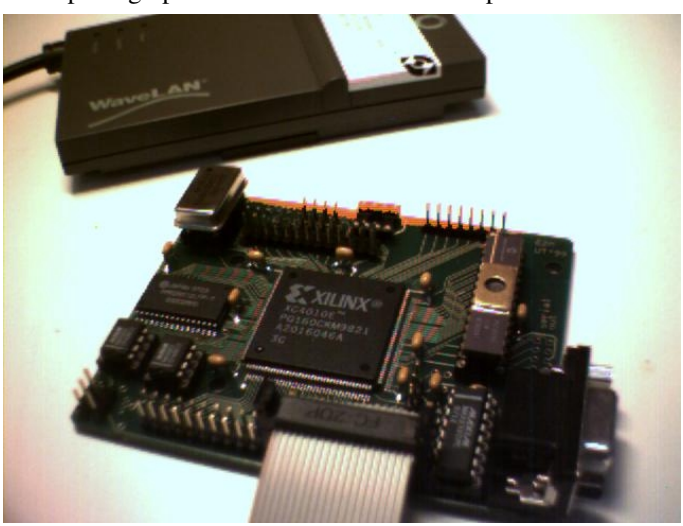


Figure 13: Network interface implementation.

We use a WaveLAN modem as the physical layer. The WaveMODEM is a RF module that converts a serial transmit data stream from the host into Radio Frequency (RF) modulated signals. When the RF signal is received, the RF signal will be demodulated into a serial data stream to the host. The raw data rate is 2 Mb/s. The WaveMODEM

[6] Note that this assumes that the connections have a guaranteed throughput (to the modules, and over the wireless channel).

operates half duplex, i.e. the modem is either transmitting or receiving. The modem provides the basic functionality to send and receive frames of data. It does not include a Medium Access Control Protocol, but provides signalling information like carrier sense.

The FPGA controls the data-flow between the radio and the host. It uses the memory to implement the queues. The FPGA does not perform any control-type operations, it just follows the instructions given by the microcontroller. The microcontroller controls the traffic-flow from the radio and from the host. It therefore performs the queue setup and administration. This administration is used to setup VCI mapping tables and queue address maps in the FPGA. It thereby uses the connection type to determine which flow-control and error control to use. It receives control messages (from either the base-station via the Traffic Control Slot, or from the host) when new connections have to be initialised, changed or removed, and when data has to be received or transmitted.

Cells arriving from the host can be protected against errors that might occur on the wireless channel by adding redundant cells. The FPGA provides the computation-intensive functions of the error control, that the microcontroller can use to build the packet that is protected by the required error correcting code. In this case, the FPGA works alongside a microcontroller that implements the remainder of the error control algorithm.

When the network interface has received traffic from the wireless link, it forwards this to the host using a previously established connection. On reception of an ATM cell, the FPGA simply looks up the VCI mapping table and the queue memory map to determine where to store the cell.

While transferring a cell to memory that originates from the wireless link, it performs *error detection* using a CRC check. Errors are reported to the microcontroller that can determine to initiate error correction on the received packet. Just like the error encoding mechanism, the error correction is being performed in a close collaboration between the FPGA and the microcontroller. The basic compute intensive operations are being performed by the FPGA, and the irregular control functions are being performed by the microcontroller.

The WaveLAN modem has a raw bandwidth of approximately 4830 ATM-cells per second. When we have a frame-rate of 100 Hz, then each frame is about 48 ATM cells large. The memory is capable to store 8000 64-byte cells, which is equivalent to about 1.6 seconds of continuous traffic.

5.6.10 Wireless communication with multiple radio's

The mobiles are expected to spend most of their time in sleep modes. This, however, also implies that they can neither transmit nor receive radio transmissions most of the time. As discussed before, a main source of unessential energy consumption is due to the costs of just being connected to the network. In E^2MaC we have tried to minimise this overhead, but still the receiver has to be switched on from time to time, just to discover whether the base-station has some messages waiting.

Another means of discovery is to use a low power RF detection circuit to wake the mobile out of sleep mode. Such a circuit can be quite small, but cannot be used to transfer bits. We could use a very low power receiver for the signalling only. This receiver can be used to wake-up a mobile and transfer connection setup requests or connection queue status information from the base station. It uses the same synchronisation mechanism between mobile and base-station, but uses a simple, low performance, low power receiver.

A further extension might be to use a dedicated *bi-directional* signalling network that could be used for the MAC protocol only and operates in parallel with the actual data-stream with another 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.

Note that the energy per bit transmitted or received tends to be lower at higher bit rates. For example, the WaveLAN radio operates at 2Mb/s and consumes 1.8 W, or 0.9 μJ/bit [79]. A commercially available FM transceiver (Radiometrix BIM-433) operates at 40 kb/s and consumes 60 mW, or 1.5 μJ/bit [61]. This makes the low bit rate radio less efficient in energy consumption for the same amount of data. However, there is a trade-off when a mobile has to listen for a longer period for a broadcast or wake-up from the base station, then the high bit rate radio will consume about 30 times more energy than the low bit rate radio. Therefore, the low bit rate radio must only be used for the basic signalling, and as little as possible for data transfer.

Another method to increase energy efficiency might be achieved by providing adequate support for broadcast or multicast. Energy can be saved when mobiles do not need to request a certain datum separately, but when the base station transmits it as a broadcast.

5.7 Evaluation of the E^2MaC protocol

The E^2MaC protocol is designed to provide QoS to various service classes with a low energy consumption of the mobile. The base-station which has plenty of energy performs actions in courtesy of the mobile. In the protocol the actions of the mobile are minimised. In the remainder we will thus only consider the energy efficiency of the mobile, and not of the base-station. The main restriction comes from the required QoS of the applications on the mobiles. The achieved energy efficiency depends on the implementation of the scheduler, the error rate, and also on the applications. The application, and also the user, must provide proper QoS requirements to the system. The E^2MaC protocol then offers the tools to the system to reduce the energy consumption that is needed for the wireless interface.

In the design of the protocol all main principles of energy efficient MAC design are used (see Section 5.3). Note that some principles interact, for example the synchronisation between base-station and mobile is not only used to power the transceiver just in time,

but also to avoid collisions. In this section we will show how these principles are used in the E^{2}MaC protocol and evaluate the attainable gain in energy reduction.

We define the energy efficiency *e* as the energy dissipation needed to transfer the a certain amount of data (e.g. a packet) divided by the total energy dissipation used for that.

$$e = \frac{\textit{Energy dissipation to transfer a certain number of bits}}{\textit{Total energy dissipation}} \qquad (3)$$

5.7.1 Synchronise the mobile and the base-station

When a mobile has a connection, then it is fully synchronised with the base-station and can – when it is idle – enter a minimal energy consuming mode, just enough to update its clock. The synchronisation is used for uplink and downlink connections. When the mobile wants to send data it first has to receive the Traffic Control Slot (TCS) to find the assigned slots to use in the frame. Since the mobile and base-station are synchronised in time, the mobile can power up the receiver on time. Note that the mobile does not need to receive the TCS of each frame.

After a connection has been set up, and the mobile has no data to send, it can simply tell the base-station that it will sleep for some time. The time is determined by the QoS of all connections of the mobile. The base-station will then release the slot and use it for other connections until the sleep period is over. When the mobile does not use the slot, then the base station will let the connection sleep again for the same period. This mechanism allows the mobile to sleep for a long period, and still be certain that it can acquire a slot within a bounded time. In this way the mobile reserves periodically a slot in a frame, and the bandwidth spent depends on the tolerable delay.

A mobile that just has to listen if there is *downlink traffic* waiting at the base-station wasts much energy. The E^{2}MaC protocol therefore tries to minimise the amount of energy needed by broadcasting such information in the Traffic Control slot of a frame. It is assumed that the mobile and base station can keep in sync for a reasonable time and thus can turn on their receiver just in time to receive the Traffic Control Slot. The moment at which the receiver has to be turned on depends on the accuracy of the clock. This allows a mobile to sleep when for some time a connection is not used. When the mobile wakes up, and the synchronisation with the base station has become unreliable, then it needs to scan for the TCS. The cost of just being connected is determined by the application of the mobile with the least tolerable delay or the drift in clocks between mobile and base station.

In reservation schemes like the E^{2}MaC protocol there is always an inevitable overhead due to the *traffic control*. In the E^{2}MaC protocol the required overhead for a mobile to receive the traffic control can be reduced when the traffic can be scheduled in advance. The mobile can request a static connection with no jitter in the frame. In this way the mobile has near-optimum energy efficiency since it does not need to listen to the traffic control: it knows when to expect the slot(s) assigned to the connection. This, however, is

selfish behaviour since it reduces the freedom of the slot-scheduler. Only when the load on the wireless channel is moderate, is the scheduler able to assign such connections. When the load is too high, the scheduler cannot fulfil all wishes. A strategy of a mobile can be to ask a best-fit connection with no jitter. The scheduler can then decide depending on the current load of the cell to honour the request or not.

5.7.2 Minimise the number of transitions

The number of *transitions* between transmitting, receiving, idle, sleep, and off is minimised by the system. The slot-scheduler tries to group the transmissions and receptions of a mobile as much as possible according to the service classes, QoS and current load. There are basically three effects that contribute to the required energy for a transition from sleep to transmission:

1. the required time and energy to change the power mode from sleep to idle. For example, the WaveLAN MODEM interface will become stable and operative within 250 μs after it was signalled to wake-up from sleep mode [79].
2. the required time and energy the interface has to be in idle and transmission mode, but not transmitting actual data. This is the overhead required to initiate and terminate the actual transmission. This time includes the required gap (guard time), interfacing delay, preamble, and afterwards the postamble. Also, as an example, for the WaveLAN interfaces this can take 466 bits per frame.
3. the required time and energy to enter the sleep mode after transmission (WaveLAN documentation does not specify this).

These effects greatly influence the required energy for the transmission of a packet. When we assume a wireless interface that has a throughput of 2 Mbit/s, then a transition time from sleep to idle of 250 μs already takes *virtually* 62.5 bytes. The overhead required to power-up is thus already more than the transmission of one ATM cell.

We will first evaluate in paragraph A the consequences on channel efficiency and energy consumption of the mobile grouping mechanism used in the E^2MaC protocol. We will compare this with phase grouping as commonly used in other MAC protocols. Then we will evaluate in paragraph B the consequences of the packet size being used.

A. Overhead

The maximal throughput of the network is determined by 1) the required guard space and physical overhead between slots, 2) the overhead in transmitting control information, and 3) by error control. The transition-overhead (see previous paragraph) to wake-up after a sleep can be done in parallel with a different communication stream and does not influence the throughput of the network. Higher-level protocol issues that might reduce the throughput (like reservation of bandwidth for e.g. mobility or error control) are not considered here.

We assume the wireless physical header and trailer to be a fact that cannot be changed or improved with a MAC protocol, although the protocol can try to minimise the number of times that these are required. Grouping of uplink and downlink traffic of one mobile

(*mobile grouping*) implicates that there is some space between sending and receiving to allow the transceiver to switch its operating mode from sending to receiving (i.e. guard space, preamble, postamble). This has a negative effect on the capacity of the wireless channel. The advantage is that it allows the mobile (i.e. the radio device) to turn its power off for a longer period, and that it makes less power-state transitions. If we would, in contrast to the mobile grouping of uplink and downlink traffic, group the downlink traffic from the base-station to all mobiles (*phase grouping*), then the space between sending and receiving that is required for mobile grouping, is not present for the downlink traffic (see Figure 8). Most MAC protocols group the traffic from the base-station, mainly because of its efficient use of the available bandwidth. However, there are consequences of phase grouping related to the energy consumption:

1) The receiver of the mobile must be on for a longer period (i.e. during the whole downlink period because it needs to synchronise using the preamble of the radio packet). If the radio would be capable of synchronising during the transmission of a downlink packet, then the mobile might be able to power down during the downlink phase. However, this will still cause an additional energy consuming power-state transition (from power-down/sleep/idle mode to an active transmission mode.
2) The period between two operations is too small to enter sleep mode. This period is determined by the time needed to enter sleep mode (T_{sleep}) and the time needed to wake-up ($T_{wake\text{-}up}$).

These two effects lead to higher energy consumption for the mobile. This shows that there is a trade-off between performance (channel efficiency) and energy consumption. The energy gained with mobile grouping depends on 1) the amount of data in the downlink phase that is not destined to the mobile, but must be received by the radio device because it is stored in the downlink transmission frame[7]. And 2) the amount of time between receiving the downlink packet and the uplink packet. Only when this time exceeds the time required to enter sleep mode (T_{sleep}) plus the time needed to wake-up ($T_{wake\text{-}up}$), then energy can be saved. Otherwise, the mobile must remain idle, waiting for its time to transmit its uplink packet.

We will now evaluate the effects of mobile grouping on the available bandwidth and on the energy consumption. We will compare this with the phase grouping mechanism. The properties of interest are:

TCS The size of the traffic control slot. In our implementation we use one ATM cell-sized slot.

O The overhead to transmit a packet. The overhead *O* consists of the overhead when the interface must be idle O_{idle} (required for guard space and interfacing delay), plus the overhead O_p required for preamble and postamble. The interfacing delay is caused by two factors. First, the delay caused by the

[7] If the network interface is able to skip packets in the downlink phase that are transmitted after the mobile has received its packet, then the downlink schedule order determines the amount of data to be received

wireless interface to synchronise to its internal syncslots. Second, we have an additional delay because we must also synchronise to the time slots that the MAC protocol uses. The MAC protocol uses fixed time slots, but since each packet is not a multiple of this slot size (because of the overhead in the wireless interface) we need to incorporate a delay with an average length of the size of a time slot divided by two.

T_{sw} This is the time needed by the wireless interface to enter sleep mode T_{sleep} plus the time $T_{wake\text{-}up}$ needed to wakeup from sleep mode to an active mode (idle, receive or transmit).

C The number of bytes used for the collision phase (reservation phase).

O_{total} The total overhead in a frame that is introduced to transmit the actual data over the wireless link.

F The size of a transmission frame.

TD The total size available for a mobile to transmit data packets. This can be expressed with:

$$TD = F - O_{total}$$

D The size of a packet (uplink and downlink) used by a mobile. We assume that the whole frame is used, and that all mobiles have an equal share. The size of an uplink and a downlink packet is thus dependent on the number of mobiles using the frame. It can be expressed with:

$$D = TD / 2M$$

M The number of mobiles, each with uplink and downlink packets.

All properties can be expressed in bytes. When a property is related to time, then we use the virtual overhead that expresses the number of bytes the wireless channel can transmit in that time.

In our analysis we will assume that each mobile has both uplink connections and downlink connections that both have similar bandwidth requirements. We further assume a packet length that allows a mobile to enter sleep mode. Thus, the packet length is greater than T_{sw}.

Evaluation phase grouping

Figure 14 shows a typical phase grouping transmission frame with three mobiles, each using downlink and uplink packets.

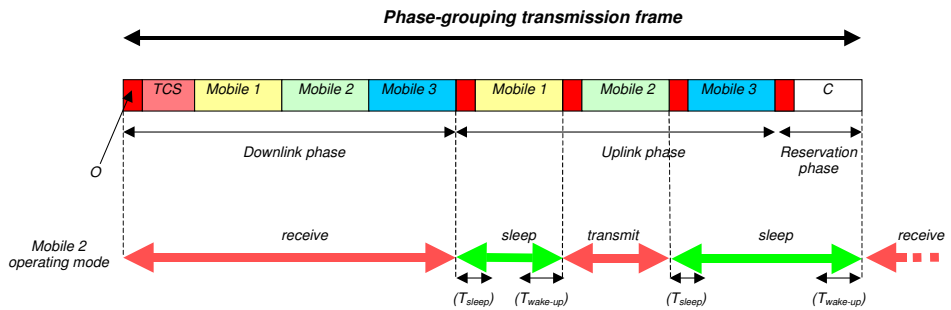


Figure 14: Phase grouping transmission frame

In general we have M mobiles, each with uplink and downlink packets. The total *overhead* O_{total} can be expressed with:

$$O_{total} = O + TCS + M.O + O + C$$

or,

$$O_{total} = TCS + (M+2).O + C \qquad (4)$$

We will now determine the total time a mobile can enter sleep mode. This time can be used to evaluate the energy consumption a mobile needs for its wireless interface.

We assume that a mobile is required to receive the whole downlink packet from the base station. Whether a mobile is able to transmit its uplink packet depends on the schedule made by the base station. In our analysis we evaluate the sleep period of a mobile that is scheduled as second in the uplink phase (e.g. Mobile 2 in Figure 14). We can then divide the uplink period in three phases: pre-uplink, uplink (in which the mobile transmits its data), and post-uplink.

When there is just one mobile, then we do not have a pre-uplink phase. The mobile can only sleep in the contention phase. The total sleep time T_{sleep} of the mobile in this situation is:

$$T_{sleep}(M=1) = O + C - T_{sw}$$

When there are more mobiles, then we have all phases. The pre-uplink sleep period is:

$$T_{sleep\text{-}pre} = (D + O) - T_{sw}$$

The post-uplink sleep period is during the remaining (M-2) data packets from the other mobiles:

$$T_{sleep\text{-}post} = (D + O) \,.\, (M\text{-}2) - T_{sw}$$

Together with the collision phase this gives a total sleep time for $M > 1$:

$$T_{sleep}(M>1) = (D + O) - T_{sw} + (D + O) \,.\, (M\text{-}2) + O + C - T_{sw}$$

Thus:

$$T_{sleep} = \begin{cases} O + C - T_{sw} & M = 1 \\ (D + O) \,.\, (M\text{-}1) + O + C - 2\,T_{sw} & M > 1 \end{cases} \qquad (5)$$

Evaluation mobile grouping

We will now evaluate mobile grouping using the same assumptions as applied for phase grouping. Figure 15 gives an example of a mobile grouping transmission frame.

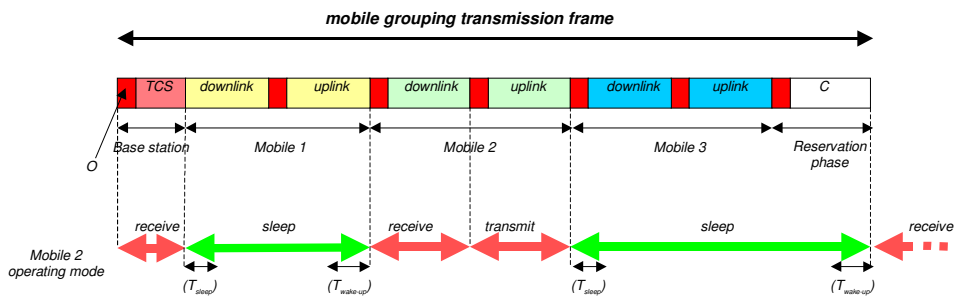


Figure 15: Mobile grouping transmission frame

In general we have M mobiles, each with uplink and downlink packets. The total overhead O_{total} can be expressed with:

$$O_{total} = O + TCS + O + 2M.O + O + C$$

or,

$$O_{total} = TCS + (2M+3).O + C \qquad (6)$$

We can divide the uplink period in three phases: pre-uplink, uplink (in which the mobile transmits its data), and post-uplink.

When there is just one mobile, then we do not have a pre-uplink phase. The mobile can only sleep in the contention phase. The total sleep time T_{sleep} of the mobile in this situation is:

$$T_{sleep}(M=1) = O + C - T_{sw}$$

When there are more mobiles, then we have all phases. The pre-uplink sleep period is:

$$T_{sleep\text{-}pre} = (2\,D + O) - T_{sw}$$

The post-uplink sleep period is during the remaining (M-2) data packets from the other mobiles:

$$T_{sleep\text{-}post} = 2\,(D + O)\,.\,(M\text{-}2) - T_{sw}$$

Together with the collision phase this gives a total sleep time for $M > 1$:

$$T_{sleep}\,(M>1) = (2\,D + O) - T_{sw} + 2\,(D + O)\,.\,(M\text{-}2) + O + C - T_{sw}$$

Thus:

$$T_{sleep} = \begin{cases} O + C - T_{sw} & M = 1 \\ (2M - 2)\,D - 2\,T_{sw} + (2M\text{-}2)O + C & M > 1 \end{cases} \qquad (7)$$

We will now apply the characteristics of the WaveLAN modem to these equations.

F 2544 bytes. (The transmission rate is 2 Mb/s. When we use a frame rate of 100 Hz, then the frame size is approximately 2544 bytes.)

TCS 53 bytes. (one ATM cell)

O 71 bytes. (O_p = 37 bytes, O_{idle} = 22 bytes. The internal time slots are 24 bytes, which results in an average synchronisation delay of 12 bytes)

C 53 bytes. (one ATM cell)

T_{sw} 73 bytes (T_{sleep}=10 bytes (unspecified by specs), $T_{wake\text{-}up}$=63 bytes)

The results are shown in Figure 16 and Figure 17.

Figure 16 shows the total overhead O_{total} caused by the two mechanisms. As expected, phase grouping induces less overhead than mobile grouping. When there are many mobiles using the frame (both in the uplink and in the downlink direction), then the overhead constitutes a significant part of the total available bandwidth. We can increase the frame size by lowering the frame rate frequency of 100 Hz. This would reduce the overhead that is required to transmit a certain amount of data, but will increase the latency.

Also shown in the figure is the packet size D (under the assumption that the whole frame is used). This clearly shows that the packet size D becomes rather small when the number of mobiles using the frame increases. The packet size of mobile grouping is a little bit smaller than of phase grouping because of the larger overhead.

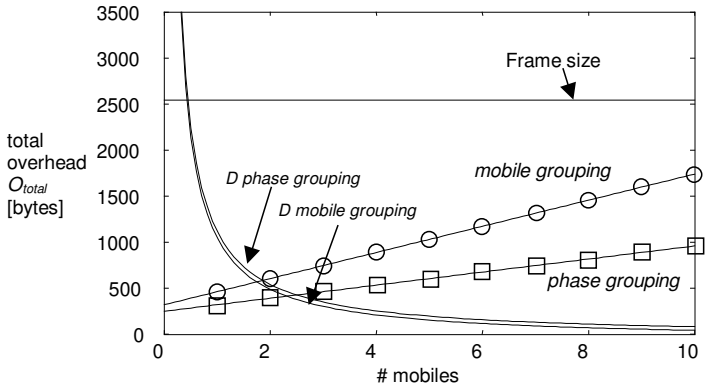


Figure 16: Total overhead versus number of mobiles.

When we look at the consequences for energy consumption, then mobile grouping is more advantageous. This is shown in Figure 17. The sleep period per mobile is larger when using mobile grouping compared to phase grouping.

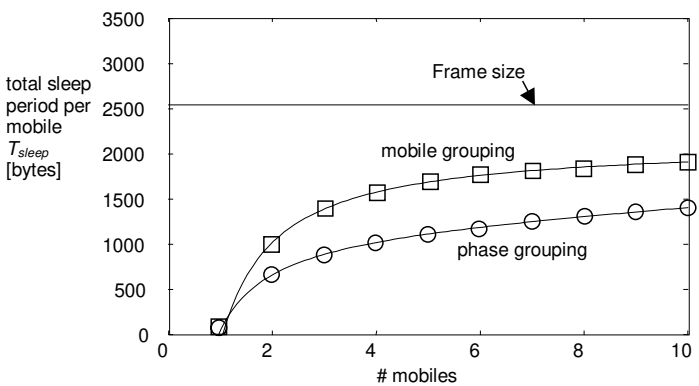


Figure 17: Total sleep period per mobile versus number of mobiles.

The figure show that the increase in the total sleep period is already high with a small number of mobiles. As the overhead increases with the number of mobiles using the wireless channel, mobile grouping seems particularly attractive for systems with a small cell size (e.g. pico-cellular with the size of an office-room). In these systems the number of mobiles in one cell will in general be small, and the available bandwidth high. Mobile grouping strategy will then have a small overhead while allowing a large sleep period.

Note that the assumptions we have made are conservative for two reasons. First, we assumed that the whole frame is used. If this is not the case, then the sleep period can be larger. However, when using phase grouping, a mobile is in general forced to receive the whole downlink packet, and cannot enter sleep mode in that phase; whereas mobile grouping only needs to receive the TCS. Second, we assumed that the amount of uplink traffic is equal to the amount of downlink traffic. This might be true for voice

applications (mobile phone), but is in general not true for applications running on a mobile computer. For these applications the downlink traffic in general will use more bandwidth. The disadvantage of having one large downlink packet (phase grouping) then becomes even more apparent.

The overhead caused by the transmission of the traffic control (TCS) depends on the frame-length, which is implementation dependent. The length is restricted by the amount of buffer-space in the base-station and the mobile, but also by the introduced latency. Figure 20 shows the effect of the frame size on the energy efficiency versus the load per mobile. The overhead for error control (i.e. to transmit a CRC and redundant data) also reduces the throughput. The required guard space between slots influences the throughput, but has no effect on the energy consumption. The size of the guard space depends on the hardware of the transceiver.

B. Packet size

Figure 18 shows the effect on the energy efficiency with respect to the packet size s and the overhead introduced with one transition. The overhead O that is required to transmit a packet consists of the virtual overhead O_v to wake-up from sleep mode, plus the overhead when the interface must be idle O_{idle} (required for guard space and interfacing delay), plus the physical overhead O_p that is required before the interface can transmit the actual data. The energy consumption when transmitting is E_{tx}, when waking-up $E_{wake-up}$, and when idling E_{idle}. The energy efficiency e_{packet} of transmitting one packet of size s is then:

$$e_{packet} = \frac{E_{tx}.s}{E_{tx}.s + E_{wake\text{-}up}.O_v + E_{idle}.O_{idle} + E_{tx}.O_p} \qquad (8)$$

We use a simplified energy model in which the energy consumption is equal in all states (waking up, idling and transmitting). For WaveLAN we have O_v = 62.5 bytes, O_{idle} = 21.25 bytes, and O_p = 37 bytes, which gives to a total overhead of 120 bytes (2.27 ATM cells).

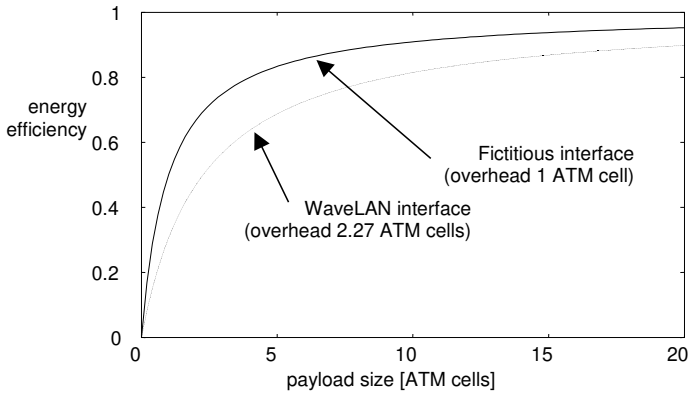


Figure 18: Energy efficiency vs. payload size as a function of overhead.

Figure 18 shows that the packet size has a big influence on the energy efficiency. It also shows the energy efficiency if we would have used a fictitious interface that has an overhead of one ATM cell. Small packet sizes are not efficient because the total overhead is large. So, when the protocol bundles communication in bursts from one mobile, much energy can be saved when compared with a scheme that for example requires two transitions per ATM cell (i.e. change the mode from idle to transmission, and back to idle).

The discussion above shows that the large transition times of a wireless interface make a large packet size profitable. However, this is valid for ideal situations in which no errors occur only. According to Equation (1) the packet error rate (PER) depends on the bit error rate (BER) and the packet size. The overhead imposed is caused by two main factors: overhead in time (power up, power down, inter-frame gap (which is the required guard space between two transmissions), transmission mode transitions) and overhead in bits transmitted over the air (preamble, MAC control header, postamble).

When we consider the *goodput,* which is the throughput a user will see, as a function of BER and packet size, then we only need to incorporate the overhead where errors influence the transmission. Since power up/down and transmission mode transitions of different mobiles can occur in parallel in time, they do not influence the goodput as well.

When we study the WaveLAN modem characteristics that are also depicted in Figure 6, then we can specify various quantities of interest. Let:

I = inter-frame guard space, 15.5 bytes

P = length of preamble (36.5) plus postamble (0.5), 37 bytes

M = length of MAC control header, in E^2MaC, 48 bytes

D = number of data bytes

The goodput g normalised to the raw bit rate of the radio can be specified as:

$$g = \frac{D}{I + P + M + D} \; (1 - BER)^{D+M} \qquad (9)$$

We have plotted this equation with the goodput g versus packet size s for various bit error rates in Figure 19. This figure clearly shows that when the channel conditions are bad, large packet sizes lead to a low goodput. If the QoS of the connection requires a better goodput, then the error control mechanism has to be adapted, or the packet size has to become smaller. The issues of packet size and error control coding are intertwined, since the amount and kind of coding needed will depend on similar factors as with packet sizing.

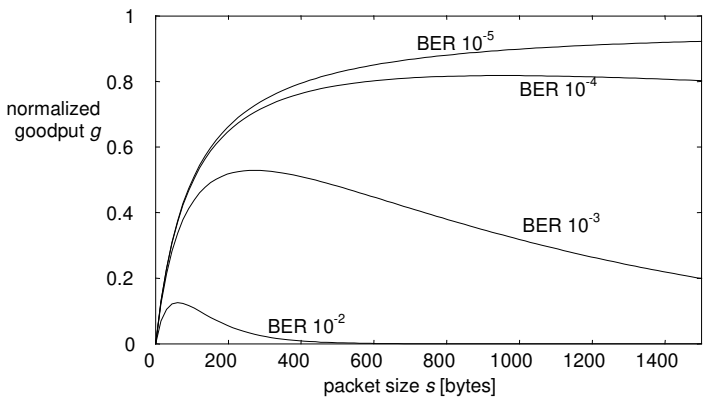


Figure 19: Goodput vs. packet size on WaveLAN for various BER.

There seems to be a trade-off concerning the packet size between energy efficiency (minimal transitions thus large packet size) and goodput (adequate packet size, not too large). However, the Traffic Control module can also adapt the error control mechanism, such that it divides the packet into smaller segments that each has their own error control. Both possible adaptations (either more redundancy on large packets or several smaller packets with less redundancy) require extra energy that is needed for the error control.

5.7.3 Avoid unsuccessful actions

In our approach unsuccessful transfers are minimised because the chance of a collision is small and the base station tries to avoid periods of bad error conditions.

- *Errors*

The error control is applied on individual connections and is tailored to the traffic type and required QoS of the connection. This structure, combined with an adaptive error control allows for an error control scheme that does not perform error control when it is not needed, but on the other hand does not give a too low reliability for a connection.

The scheduler at the base-station plays an active role in the error control. It not only determines the required error coding for each connection, it also tries to avoid periods of bad-error conditions by not scheduling non-time critical traffic during these periods. How profitable this latter approach is, depends on factors like the typical size of an error burst and on how fast the slot-scheduler can react (which also depends on the frame-length). Energy will be saved in any case (and will be maximal the amount of energy that otherwise would have been wasted during the bad-error period), but the consequence for the throughput in a cell is more complicated, because other – error free – connections will use the bandwidth instead. As already stated, this can lead to a throughput that may even exceed the average rate on the channel.

- *Collisions*

The chance of a *collision* in the E^2MaC protocol is small since 1) it can only occur when a mobile enters the cell and requests a connection, and 2) because many slots (i.e. all not used slots in a frame) can be used to request the connection. When a mobile has a connection, then it has reserved slots, and no collisions occur.

In this section we will compare the energy efficiency of a mobile with *uplink traffic* using the E^2MaC protocol and Slotted Aloha, a collision based protocol that is often used as a reference and is also used in many systems as the basis of the access mechanism. Downlink traffic is not considered since Slotted Aloha (just like many other protocols) does not care about the energy consumption, and just assumes that mobiles turn their receiver on to find out about downlink traffic. We will not incorporate insignificant details, but will concentrate on the main issue to show the difference in energy efficiency between a reservation and a collision protocol. Energy saving properties like avoiding periods of bad error conditions are not incorporated.

Energy efficiency of the access mechanism in Slotted Aloha

In Slotted Aloha, time is divided into slots [2]. Each slot is accessed with probability p by each mobile. When we assume that the aggregate network load does not change when a single station goes in backoff, then we can state that whether or not backoff is used, the probability of success of each data transmission does not change. Therefore, a backoff procedure is of no concern in the analysis of energy consumption [48]. The energy dissipated is determined by the time that the transmitter and receiver must be on. We neglect the energy needed to receive the identification message from the base station since this happens only once when entering a cell. When the mobile sends a message, the probability π that it is successfully received by the base station is:

$$\pi = (1 - p)^{n-1} \qquad (10)$$

where p is the probability that a station sends a message in a slot and n is the number of active mobiles in a cell. The average number of transmissions υ needed to send a message successfully is given by:

$$\upsilon = \frac{1}{(1-p)^{n-1}} = (1-p)^{1-n} \qquad (11)$$

Every time the mobile attempts to send a message, the receiver is switched on to receive possible positive acknowledgements. Using υ, the average time T_{tx} that the transmitter is active for a successful packet transmission is determined to be:

$$T_{tx} = \upsilon . T_{data} \qquad (12)$$

Similarly, the time T_{rx} per packet that the receiver is switched on in order to receive an acknowledgement is given by:

$$T_{rx} = \upsilon . T_{ack} \qquad (13)$$

The total energy dissipation is given by the time that the transmitter is on, plus time that the receiver is turned on to receive the possible acknowledgements, multiplied by the power dissipations of each of these functions.

The energy efficiency e_{sa} is thus determined by:

$$e_{sa} = \frac{T_{data} . P_{tx}}{T_{tx} . P_{tx} + T_{rx} . P_{rx}} \qquad (14)$$

in which T_{data} is the time to transmit one packet, T_{tx} is the time to successfully transmit a packet, T_{rx} the time to receive the acknowledge, P_{tx} the power dissipation for transmission and P_{rx} the energy dissipation for reception.

Energy efficiency of the access mechanism in E²MaC

The access mechanism used in E²MaC is based on a TDMA structure where the base station assigns time (slots) to mobiles in which they are allowed to transmit. Since in the E²MaC protocol collisions can only occur when the mobile enters a cell, their contribution to the average energy consumption per message can be neglected. When a connection has been set-up, the overhead is determined by the reception of the traffic control.

When a mobile indicates that it has continuous traffic and does not allow any jitter, then the slot-scheduler will reserve the same slots in each frame for that connection. The mobile thus only needs to receive the traffic control once. This situation has almost optimal energy efficiency and will not be analysed further. When not each frame is used by the connections of a mobile, then the mobile does not need to receive the TCS either. We will only analyse the worst case in which a mobile needs to receive the traffic control once per frame.

The time that the receiver has to be on per frame to receive the Traffic Control Slot is determined by the number N of slots in a frame. Since no collisions can occur,

acknowledgements are not needed on this level. The energy efficiency of the access mechanism e_{e2mac} is thus:

$$e_{e2mac} = \frac{p \,.\, T_{tx} \,.\, P_{tx}}{p \,.\, T_{tx} \,.\, P_{tx} + (T_{rx} \,.\, P_{rx}) / N} \qquad (15)$$

where N is the number of slots in a frame, p the probability that a mobile sends a packet in a frame, T_{rx} is the time needed to receive the traffic control slot, T_{tx} the time to transmit the packet, P_{tx} the power dissipation for transmission and P_{rx} the energy dissipation for reception.

Comparison

In our analysis we will assume that the energy consumption for transmission is equal to reception, thus $P_{tx} = P_{rx}$. This approximates the power consumption characteristics of the WaveLAN 2.4 GHz modem [79]. In our analysis of Slotted Aloha we will further assume that the acknowledgement uses the same channel as used for data transfer and that the receiver needs to be on for 1/8 of the time to transmit one data message (which is optimistic when the size of a slot is one ATM cell). So we will use $T_{rx} = 1/8\ T_{tx}$.

Figure 20 shows the energy efficiency characteristics of Slotted Aloha for a various number of mobiles, and for E^{2}MaC for two frame-sizes.

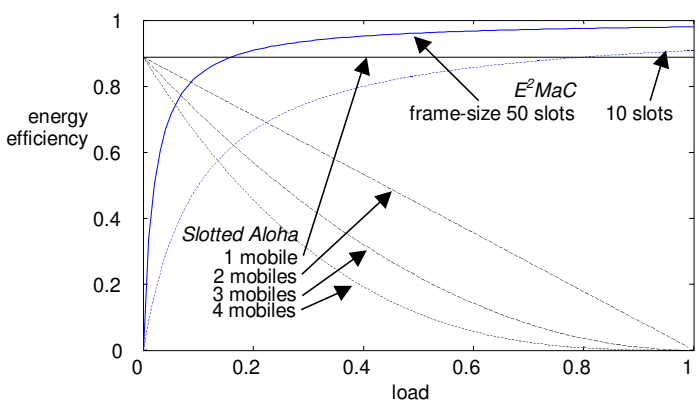


Figure 20: Energy efficiency vs. load for uplink traffic on Slotted Aloha and E^{2}MaC.

It is difficult to make a fair comparison for several reasons. Slotted Aloha requires an explicit acknowledgement as part of the MAC protocol. In E^{2}MaC no acknowledgements are required at that level, but there will be acknowledgements at higher layers of the protocol stack. In our comparison we will only incorporate MAC level issues.

Further, the energy efficiency of the E^{2}MaC protocol is independent of the activity of the users, in contrast to Slotted Aloha where the efficiency strongly depends on the activity

of other users. Therefore, the indicated load for Slotted Aloha is the total load in a cell, and for E^{2}MaC it is the load per mobile.

The energy efficiency of the E^{2}MaC protocol is much better than the Slotted Aloha protocol. Only when the load in the cell is very low (e.g. when there is only one mobile in the cell communicating with a small packet size at a low rate), then the energy consumed by the E^{2}MaC protocol is more than with Slotted Aloha that does not have this overhead. However, when a connection was requested with a best-fit option, or when the load in the cell is low, then the scheduler could have decided to establish a connection that uses the same slots in all frames. This gives the mobile a near-optimal efficiency because it does not even have to receive traffic control. Furthermore, with Slotted Aloha, when the load is higher the chance of collisions grow, leading to retries and unpredictable delays. QoS provisions are thus not possible using Slotted Aloha.

The figure also shows the consequences for the energy efficiency of the E^{2}MaC protocol of various frame-sizes. When the frame-size is larger the energy efficiency is better because the traffic control will be averaged over more data.

In many cases the receiver hardware consumes less energy than the transmitter, thus $P_{tx} > P_{rx}$. This, however, has little influence on the characteristics, and the conclusions remain the same.

5.8 Related work

In recent years much research has been done in providing QoS for the wireless link. Access protocols for these systems typically address network performance metrics such as throughput, efficiency, and packet delay. However, thus far little attention is given to energy conserving protocols, and mainly focuses their effort on energy reduction by circuit design. For example current designs for cellular phones have set aggressive goals for standby time, though all of their efforts are focused on supply voltage and circuit design [54]. The few that showed some attention to low power protocol design uses one or few principles to minimise the energy consumption and cannot provide QoS to end-users.

Lettieri [45] shows that there is much to be gained from variable frame length in terms of user seen throughput, effective transmission range, and transmitter power for wireless links. This is interesting, since we have shown that using a *fixed* frame size can save energy. Their point of view, however, was inspired by the high error rate on wireless links, where a high error rate on a large frame might not be efficient. A similar effect is reached when the error control would adapt to the current error condition of the radio channel. This is the approach that we have taken in E^{2}MaC. A similar point of view to reduce energy consumption (i.e. incorporating the error condition of the radio channel) is used in [16]. They try to avoid transmission during bad channel periods in order to reduce the number of unsuccessful transmissions. Both protocols, however, lack QoS provisions. In E^{2}MaC the scheduler of the base station tries to avoid only non-time critical traffic during these periods, thereby not affecting traffic with demanding QoS.

The 802.11 protocol [41] addresses energy consumption explicitly. In this approach the mobile is allowed to turn off and the base station buffers data destined for the mobile meanwhile. The mobiles have to be synchronised to wake up at the same time the base station announces buffered frames for the receiver. Afterwards the mobiles request the frame from the base station. This mechanism saves energy but also influences the QoS for the connections drastically. It also uses a traffic control for inter-frame synchronisation, but does not guarantee that it will not be delayed since any packet is being transmitted using a CSMA-like technique, and collisions can incur an indefinite delay.

HIPERLAN [25] is the wireless LAN specified by the ETSI. Its energy saving is based on two mechanisms: a dual data rate radio, and buffering. Because HIPERLAN is based on a broadcast channel, each station needs to listen to all packets in its range. To decide whether the station is the destination of a packet, each packet is divided into a low-power low bit-rate (1.4706 Mb/s) part to transmit acknowledgement packets and the packet header, and a high power high bit-rate (23.5294 Mb/s) part to transmit the data packet itself. HYPERLAN does not need a dedicated base station, but any station can become a so-called forwarder. Forwarders use a forwarding mechanism to build the infrastructure. The physical size of a HIPERLAN is thus a function of the current position of all stations. Power saving is based on a contract between at least two stations. The station that wants to save power is called the power-saver, and the station that supports this is the power-supporter. Power-supporters have to queue all packets destined for one of its power-savers. Forwarders and power-supporters are not expected to be mobiles since they have to receive, buffer, and forward packets sent to one of its clients. The p-saver is active only during pre-arranged intervals. Since this interval is minimal 500 ms, it cannot be used for most time-bounded traffic [81].

The R-TDMA protocol is shown to be energy efficient [48], but this is mainly due to the reservation scheme that is used to provide QoS for real-time connections. This clearly shows the advantage of having a time slotted reservation scheme. Other energy saving techniques are not applied.

The protocol design of the energy-conserving medium access control (EC-MAC) protocol [70][69] is related to the E^2MaC protocol in the sense that it provides QoS and uses a fixed frame length to allow transceivers to turn their radio on exactly in time. Their protocol, however, does not provide the close QoS relationship E^2MaC has with its **sleep** command (so that the mobile does not need to receive all Traffic Control Slots). It further does not apply some of the energy saving mechanisms such as dynamic flow control and dynamic adaptations to varying error conditions. The protocol uses phase grouping of traffic.

The principle of synchronisation between mobile and base station has been used for some time in paging systems [54]. Paging systems increase battery life by allowing the receiver to be turned off for a relatively long time, while still maintaining contact with the paging infrastructure using a well designed synchronous protocol using various forms of TDM.

The LPMAC protocol [53] uses a similar approach, but it requires that the mobile always receives the traffic control. It also allows bulk data transfers, but provides no QoS guarantees, and has no explicit mobile grouping of traffic.

5.9 Conclusions

In this chapter we have first pointed out that separating the design of the protocol from the context in which it exists, leads to penalties in performance and energy consumption that are unacceptable for wireless, multimedia applications. Then, we have presented an architecture of a highly adaptive network interface and a novel MAC protocol that provides support for diverse traffic types and QoS while achieving a good energy efficiency of the wireless interface of the mobile. The main complexity is moved from the mobile to the base station with plenty of energy. The scheduler of the base station is responsible to provide the connections on the wireless link the required QoS and tries to minimise the amount of energy spend by the mobile. The main principles of the E^2MaC protocol are: avoid unsuccessful actions, minimise the number of transitions, and synchronise the mobile and the base-station. We have shown in this chapter that considerable amounts of energy can be saved using these principles. The protocol is able to provide near-optimal energy efficiency (i.e. energy is spent for the actual transfer only) for a mobile within the constraints of the QoS of all connections, and only requires a small overhead. Most of the resulting energy waste comes from the relatively long transition times between the various operating modes of current wireless radio's. Minimising these transition times in future radio designs will be beneficial and will further reduce the energy consumption significantly.

A particular novel mechanism of the E^2MaC protocol is the mobile grouping of traffic. This mobile grouping strategy reduces the number of operating mode transitions between transmitting, receiving, active, and sleep, and maximises the possible sleep period of the transceiver. We have made the involved trade-off between performance and energy efficiency in favour of the latter because energy efficiency is one of our main concerns, and because the overhead in our system will be small (because we group all traffic as much as possible and we use small cell size with only a few mobiles per cell). Future work can be found in the development and analysis of wireless scheduling algorithms to provide QoS bounds to the various traffic types that incorporates the energy efficiency principles as determined in this chapter.

This protocol is not suited for ad-hoc networks with multiple mobiles, since much of the complexity and energy requirements is moved to a base station to provide a high energy efficiency for the mobile. Furthermore, the typical traffic on an ad-hoc network is quite different from a network with a base station. Therefore, a hybrid MAC protocol that can operate in two modes and that is optimised for both network types will probably be the most efficient.

We have shown that energy-awareness must be applied in almost all layers of the network protocol stack. Instead of trying to save energy at every separate layer, like

trying to implement TCP efficiently for wireless links [5], we have shown that applying energy saving techniques that impact all layers of the protocol stack can save more energy. To achieve maximal performance and energy efficiency, *adaptability* is important, as wireless networks are dynamic in nature. Adaptability cannot be effectively implemented in one separate layer. Furthermore, if the application layer is provided with feedback on the communication, advantage can be taken of the differences in data streams over the wireless link. To allow this, feedback is needed from almost all layers: the physical layer provides information on link quality, the medium access layer on effectiveness of its error correction, and the Data Link Control layer on buffer usage and error control. Also, if the transport layer is provided with proper feedback, it can make better differentiation between the needs for congestion control and retransmission.

Migration of some functionality from the mobile, for example to the base-station, allows reduction of the complexity of mobiles. Only a few simple components are now needed for the implementation of the network interface. Added complexity in the base-station or other parts of the fixed network is justified because they can be better equipped and are not battery powered.

The programming paradigm of *Inferno* is well suited for transparent distribution and migration of functionality. Inferno also allows easy implementation of feedback through layers of the network protocol stack up to the level of the applications.

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] Abramson, N.: “Development of the ALOHANET”, *IEEE transactions on Information Theory*, vol. IT-31, pp. 119-123, March 1985.

[3] Agrawal P., Chen J-C, Kishore S., Ramanathan P., Sivalingam K.: “Battery power sensitive video processing in wireless networks”, *Proceedings IEEE PIMRC’98*, Boston, September 1998.

[4] Ahlquist G.C., Rice M., Nelson B.: “Error control coding in software radios: an FPGA approach”, *IEEE Personal Communications,* August 1999, pp. 35-39, 1999.

[5] Akyildiz I.F., McNair J., Martorell L.C., Puigjaner R., Yesha Y.: "Medium Access Control protocols for multimedia traffic in wireless networks", *IEEE Network*, pp.39-47, July/August 1999.

[6] Balakrishnan H., et al.: “A comparison of mechanisms for improving TCP performance over wireless links”, *Proceedings ACM SIGCOMM’96*, Stanford, CA, USA, August 1996.

[7] Bauchot F., Decrauzat S. Marmigere G., Merakos L., Passa N.: “MASCARA, a MAC protocol for wireless ATM”, *proceedings ACTS Mobile Summit*, pp. 556-562, Granada, Spain, Nov. 1996.

[8] Blaum M., et al.: “EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures”, *IEEE Transactions on computers*, Vol. 44, No 2, pp. 192-201, February 1995.

[9] Birk Y. and Keren Y.: “Judicious Use of Redundant Transmissions in Multi-Channel ALOHA Networks with Deadlines”, *proceedings IEEE Infocom’98*, pp. 332-338, March 1998.

[10] Borriss, M. “QoS support in ATM and selected protocol implementations”, *technical report TU Dresden*, IBDR, http://www.inf.tu-dresden.de/~mb14/atm.html, Oct. 1995.

[11] Bowers H., Zhang H.: “Comparison of Reed-Solomon codec implementations”, *Technical rep. UC Berkeley*, http://infopad.eecs.berkeley.edu/~hui/cs252/rs.html.

[12] Chen T.-W., Krzyzanowski P., Lyu M.R., Sreenan C., Trotter: “A VC-based API for renegotiable QoS in wireless ATM networks”, *Proceedings IEEE ICUPC’97*, 1997.

[13] Chen T.-W., Krzyzanowski P., Lyu M.R., Sreenan C., Trotter: “Renegotiable Quality of Service – a new scheme for fault tolerance in wireless networks”, *Proceedings FTCS’97*, 1997.

[14] Chen, et al. “Comparison of MAC Protocols for Wireless Local Networks Based on Battery Power Consumption”, *IEEE Infocom’98*, San Francisco, USA, pp. 150-157, March 1998.

[15] Cho, Y.J., Un, C.K.: “Performance analysis of ARQ error controls under Markovian block error pattern”, *IEEE Transactions on Communications*., Vol. COM-42, pp. 2051-2061, Feb-Apr. 1994.

[16] Chockalingam, A., Zorzi, M.: “Energy consumption performance of a class of access protocols for mobile data networks”, *VTC’98*, Ottawa, Canada, May 1998.

[17] Choi S., Shin K.G.: “A cellular wireless local area network with QoS guarantees for heterogeneous traffic”, *Mobile networks and applications* 3, pp. 89-100, 1998.

[18] Ciotti C., Borowski J.: "The AC006 Median Project – Overview and State-of-the-Art", *ACTS Mobile Summit*, Granada, Nov. 96, http://www.imst.de/mobile/median/median.html.

[19] Colombo G., Lenzini L., Mingozzi E., Cornaglia B., Santaniello R.: Performance evaluation of PRADOS: a scheduling algorithm for traffic integration in a wireless ATM network”, *Proceedings of the fifth annual ACM/IEEE international conference on mobile computing and networking (MobiCom’99)*, pp. 143-150, August 1999.

[20] Dorward S., Pike R., Presotto D., Ritchie D., Trickey H., Winterbottom P.: “Inferno”, *Proceedings COMPCON Spring’97*, 42nd IEEE International Computer Conference, 1997, URL: http://www.lucent.com/inferno.

[21] Eckhardt D., Steenkiste P.: “Measurement and analysis of the error characteristics of an in building wireless network”, *Proceedings of the SIGCOMM ’96 Symposium on Communications Architectures and Protocols*, pp. 243-254, Stanford, August 1996, ACM.

[22] Eckhardt D., Steenkiste P.: “A trace-based evaluation of adaptive error correction for a wireless local area network”, *Journal on Special Topics in Mobile Networking and Applications (MONET)*, special issue on Adaptive Mobile Networking and Computing, 1998.

[23] Eckhardt D.A., Steenkiste P.: “Improving wireless LAN performance via adaptive local error control”, *Sixth IEEE International conference on network protocols (ICNP’98)*, Austin, October 1998.

[24] Elaoud, M, Ramanathan, P.: “Adaptive Use of Error-Correcting Codes for Real-time Communication in Wireless Networks”, *proceedings IEEE Infocom’98*, pp. 548-555, March 1998.

[25] ETSI: "High Performance Radio Local Area Network (HIPERLAN)", *draft standard ETS 300 652*, March 1996.

[26] Ferrari, D.: “Real-Time Communication in an Internetwork”, *Journal of High Speed Networks*, Vol. 1, n. 1, pp. 79-103, 1992

[27] Figueira, N.R., Pasquale, J.: “Remote-Queueing Multiple Access (RQMA): Providing Quality of Service for Wireless Communications”, *proceedings IEEE Infocom’98*, pp. 307-314, March 1998.

[28] Goslin G.R.: “Implement DSP functions in FPGAs to reduce cost and boost performance”, EDN magazine, 1996, http://www.ednmag.com/reg/1996/101096/21df_05.htm.

[29] Han R.Y., Messerschmitt: “Asymptotically reliable transport of multimedia/graphics over wireless channels”, *Proc. Multimedia Computing and Networking*, San Jose, Jan. 29-31, 1996.

[30] Haskell P., Messerschmitt D.G.: “In favor of an enhanced network interface for multimedia services”, *IEEE Multimedia Magazine*, 1996.

[31] Haskell P., Messerschmitt D.G.: “Some research issues in a heterogeneous terminal and transport environment for multimedia services*”, Proc. COST #229 workshop on adaptive systems, Intelligent Approaches, Massively Parallel Computing and Emerging Techniques in Signal Processing and Communications*, Bayona, Spain, Oct. 1994.

[32] Havinga P.J.M., Smit G.J.M., Bos M.: “Energy efficient wireless ATM design”, *proceedings second IEEE international workshop on wireless mobile ATM implementations (wmATM’99)*, pp. 11-22, June 1999.

[33] Havinga P.J.M., Smit G.J.M., Bos M.: "Energy efficient wireless ATM design", to appear in *ACM/Baltzer Journal on Mobile Networks and Applications (MONET), Special issue on Wireless Mobile ATM technologies, Vol. 5, No 2., 2000.*

[34] Havinga P.J.M., Smit G.J.M.: "Low power system design techniques for mobile computers", *CTIT technical report 97-32*, the Netherlands, 1997.

[35] Havinga P.J.M., Smit G.J.M.: "The Pocket Companion's Architecture", *Proceedings Euromicro Summer School on Mobile Computing '98*, pp. 25-34, Oulu, Finland, August 1998.

[36] Havinga, P.J.M., "Energy efficiency of error correcting mechanisms for wireless communication", *CTIT technical report 98-19*, 1998, the Netherlands.

[37] Havinga, P.J.M., Smit, G.J.M.: "Minimizing energy consumption for wireless computers in Moby Dick", *proceedings IEEE International Conference on Personal Wireless Communication ICPWC'97*, Dec. 1997.

[38] Hettich A., Evans D., Du Y., Lott M., Fifield R.: "Fast uplink signalling for an ATM radio interface using energy burst with random access", *proceedings wmATM'99*, pp.167-176, June, 1999.

[39] Huitema, C.: "The case for packet level FEC", *Proceedings 5th workshop on protocols for high speed networks*, pp. 109-120, Sophia Antipolis, France, Oct. 1996.

[40] Hyden E. A., "Operating System support for Quality of Service, *Ph.D. thesis, University of Cambridge*, 1994.

[41] IEEE, "Wireless LAN medium access control (MAC) and physical layer (PHY) Spec." P802.1VD5, *Draft Standard IEEE 802.11*, May 1996.

[42] Klein Gebbink J.P.A., Nienhuis M.L.: "An energy efficient wireless Communication system with Quality of Service", *Ms. Thesis University of Twente*, July 1999.

[43] Kohiyama, K., Hashimoto A.: "Advanced Wireless Access System", *Telecom'95*, Geneva, October 1995.

[44] Lettieri P., Schurgers C., Srivastava M.B.: "Adaptive link layer strategies for energy efficient wireless networking", ACM WINET.

[45] Lettieri, P., Srivastava, M.B.: "Adaptive Frame Length Control for Improving Wireless Link Throughput, Range, and Energy Efficiency", *IEEE Infocom'98*, San Francisco, USA, pp. 307-314, March 1998.

[46] Lin S., Costello D.J. Jr.: "Error control coding: fundamentals and applications", *Prentice-Hall*, 1983.

[47] Lin, S., Costello, D.J., Miller, M.: "Automatic-repeat-request error-control schemes", *IEEE Comm. Magazine*, v.22, n.12, pp. 5-17, Dec 1984.

[48] Linnenbank, G.R.J.: "A power dissipation comparison of the R-TDMA and the Slotted-Aloha wireless MAC protocols", *Moby Dick technical report*, http://www.cs.utwente.nl/~havinga/papers/macenergy.ps, 1997.

[49] Liu, H., El Zarki, M.: "Delay bounded type-II hybrid ARQ for video transmission over wireless networks", *proceedings Conference on Information Sciences and Systems*, Princeton, March 1996.

[50] Lorch, J., Smith, A. J.: "Software strategies for portable computer energy management", *IEEE Personal Communications Magazine*, 5(3):60-73, June 1998.

[51] MacWilliams, F.J., Sloane, N.J.A.: "The theory of error-correcting codes", *North-Holland Publicing Company*, Amsterdam, 1977.

[52] Makrakis D.M., Mander R.S., Orozco-Barbosa L., Papantoni-Kazakos P.: "A spread-slotted random-access protocol with multi-priority for personal and mobile communication networks carrying integrated traffic", *Mobile Networks and Applications* 2, pp.325-331, 1997.

[53] Mangione-Smith, B. et al.: "A low power architecture for wireless multimedia systems: lessons learned from building a power hog", *proceedings ISLPED 1996*, Monterey CA, USA, pp. 23-28, 1996.

[54] Mangione-Smith, B.: "Low power communications protocols: paging and beyond", Low power symposium 1995, http://www.icsl.ucla.edu/~billms/Publications/pagingprotocols.pdf.

[55] Mathis, M., et al., "RFC2018: TCP selective acknowledgement option", Oct. 1996.

[56] Meng T.H., Hung A.C., Tsern E.K., Gordon B.M.: "Low-power signal processing system design for wireless applications", *IEEE Personal communications*, Vol. 5, No. 3, June 1998.

[57] Mikkonen J., Kruys J.: "The Magic WAND: a wireless ATM access system", *proceedings ACTS Mobile Summit*, pp. 535-542, Granada, Spain, Nov. 1996.

[58] Mikkonen J.: "Wireless ATM overview", *Mobile Communications International*, Issue 28, pp. 59-62, Feb. 1996.

[59] Moorman, J.R., Lockwood J.W.: "Multiclass priority fair queuing for hybrid wired/wireless quality of service support", *Proceedings of the second ACM international workshop on Wireless Mobile Multimedia (WoWMoM'99),* pp. 43-50, August 1999.

[60] Nobelen R. van, Seshadri N., Whitehead J., Timiri S.: "An adaptive radio link protocol with enhanced data rates for GSM evolution", *IEEE Personal Communications*, pp. 54-64, February 1999.

[61] Nonnenmacher, J., Biersack, E.W.: "Reliable multicast: where to use Forward Error Correction", *Proceedings 5th workshop on protocols for high speed networks*, pp. 134-148, Sophia Antipolis, France, Oct. 1996.

[62] Radiometrix, "Low Power UHF Data Transceiver Module", http://www.radiometrix.co.uk/products/bimsheet.htm

[63] RealPlayer, http://www.realplayer.com

[64] Reiniger D., Izmailov R., Rajagopalan B., Ott M., Raychaudhuri D.: "Soft QoS control in the WATMnet broadband wireless system", *IEEE Personal Communications*, pp. 34-43, February 1999.

[65] Rizzo, L.: "Effective Erasure Codes for Reliable Computer Communication Protocols", *ACM Computer Communication Review*, Vol. 27- 2, pp. 24-36, April 97.

[66] Schuler C.: "Optimization and adaptation of error control algorithms for wireless ATM", *International Journal of Wireless Information Networks*, Vol. 5, No. 2, April 1998.

[67] Shacham, N., McKenney, P.: "Packet recovery in high-speed networks using coding and buffer management", *Proceedings IEEE Infocom'90*, San Fransisco, pp. 124-131, May 1990.

[68] Shakkottai S., Srikant R.: "Scheduling real-time traffic with deadlines over a wireless channel", *Proceedings of the second ACM international workshop on Wireless Mobile Multimedia (WoWMoM'99),* pp. 35-42, August 1999.

[69] Sivalingam, K.M., Chen J.C., Agrawal, P., Srivastava, M.B.: “Design and analysis of low-power access protocols for wireless and mobile ATM networks”, *Journal on special topics in mobile networking and applications (MONET)*, June 1998.

[70] Sivalingam, K.M., Srivastava, M.B. Agrawal, P.: “Low power link and access protocols for wireless multimedia networks”, *Proceedings IEEE Vehicular Technology Conference*, Phoenix, AZ, pp. 1331-1335, May 1997.

[71] Smit G.J.M., et al.: “Overview of the Moby Dick project”, *Proceedings Euromicro Summer School on Mobile Computing '98*, pp. 159-168, Oulu, Finland, August 1998.

[72] Smit, G.J.M., Havinga, P.J.M., van Opzeeland, M., Poortinga, R.: “Implementation of a wireless ATM transceiver using reconfigurable logic”, *proceedings IEEE wmATM'99*, pp. 241-250, June 2-4 1999.

[73] Srivasta M.: "Design and optimization of networked wireless information systems", *IEEE VLSI workshop*, April 1998.

[74] Stemm, M. et al.: “Reducing power consumption of network interfaces for hand-held devices”, *Proceedings MoMuc-3*, 1996.

[75] Su W., Gerla M.: “Bandwidth allocation strategies for wireless ATM networks using predictive reservation”, *IEEE Globecom '97*, 1997.

[76] Swann R., Kingsbury N.: “Error resilient transmission of MPEG-II over noisy wireless ATM networks”, *IEEE proceedings of the International Conference on Image Processing*, Santa Barbara, October 1997.

[77] Swann R.: “Bandwidth efficient transmission of MPEG-II Video over noisy mobile links”, *Signal Processing*, Vol. 12, No. 2, pp. 105-115, April 1998.

[78] Truman T.E.: “A methodology for the design and implementation of communication protocols for embedded wireless systems”, *Ph.D. thesis, University of California, Berkeley*, spring 1998.

[79] “WaveLAN/PCMCIA network adapter card”, http://www.wavelan.com/support/libpdf/fs-pcm.pdf.

[80] WaveMODEM 2.4 GHz Data Manual, Release 2, AT&T 1995.

[81] Woesner H., Ebert J., Schläger M., Wolisz A.: "Power-saving mechanisms in emerging standards for wireless LANs: The MAC level perspective", *IEEE Personal Communications*, Vol. 5, No. 3, June 1998.

[82] Zorzi, M., Rao, R. R.: “Error control and energy consumption in communications for nomadic computing”, *IEEE transactions on computers*, Vol. 46, pp. 279-289, March 1997.

[83] Zorzi, M., Rao, R. R.: “On the impact of burst errors on wireless ATM”, *IEEE Personal Communications,* August 1999, pp.65-76.

[84] Zorzi, M., Rao, R. R.: “On the statistics of block errors in bursty channels”, *IEEE transactions on communications, 1998.*

[85] Zorzi, M: “Performance of FEC and ARQ Error control in bursty channels under delay constraints”, *VTC'98*, Ottawa, Canada, May 1998.

6

Concluding remarks

In this chapter we will first evaluate the effectiveness of our approach in designing an energy-efficient architecture for hand-held multimedia computers. In particular we will compare the power dissipation of the test-bed of Mobile Digital Companion with a traditional architecture using a typical multimedia application. Then we give some suggestions for future research. We conclude with some general conclusions about the main issues discussed in this thesis.

6.1 Evaluation of power dissipation

The connection-centric approach with application domain specific modules gives a number of advantages like (energy) efficient processing, high performance, elimination of useless data copies, relieve of the general-purpose processor, and the possibility of an adequate energy management. We already gave some practical and theoretical background on the amount of energy that can be saved using such an architecture.

In this section we will show the effectiveness on the energy consumption of our architecture using a typical multimedia application. We will do this by comparing the power consumption of wirelessly receiving and playing MP3 music on a traditional architecture and on a Mobile Digital Companion. We will further compare the power dissipation of these architectures when idling, and waiting for an external event.

Two considerations have to be made when interpreting the resulting power dissipation:

- The architecture of a Mobile Digital Companion as described in this thesis encompasses various levels in the design cycle. The design approach is vertical oriented, which implies that all layers of the system are involved, and an optimal effect on the performance and efficiency is reached only when all layers co-operate. This thesis only covers the general system architecture, the interconnection architecture, and the wireless interface.

- In the comparison we will use the actual power consumption measured on the test-bed, and for those parts that have not been implemented, we will use the power consumption numbers from datasheets. The power consumption of the traditional architecture will be based solely on datasheets, since measuring the power consumption of a notebook computer would be too flattering and not fair.

  Further note that the test-bed is designed to evaluate the *energy efficiency* of designs, and is *not* designed to be low power! The actual implementation of the test-bed is therefore primarily designed to be *flexible*, and suitable for doing experiments with various design alternatives. Because of this, the implementation test-bed used various flexible, but certainly not low-power components (i.e. we have used Xilinx FPGAs).

With these above mentioned considerations in mind, we will now compare the power consumption of wireless receiving and playing MP3 music on a traditional architecture and on a Mobile Digital Companion.

The power consumption of the various hardware modules involved are gathered from datasheets and from measurements on our testbed prototype. We have only included the main components, and have neglected the parasitic power consumption due to glue logic and the energy consumed for the actual data transfer (except for the bus). The audio module is invariant in our comparison, since both setups are assumed to use the same hardware in the same way. The wireless network interface is in both setups also the same, although a different MAC protocol is being used.

We assume that there are three basic operating modes: *active* in which the device is fully operational, *sleep/idle* in which the device is idling and ready to become active, and *off* in which a device is completely powered down. It is further assumed that the mobile system uses a dynamic power management that has powered down (off state) all other components of the system that are not in use.

We will indicate with τ_{active} the percentage of time being active and with τ_{idle} the percentage of time the part is idle. The power consumption P of each part can then be calculated using:

$$P = \tau_{active} \cdot P_{active} + \tau_{idle} \cdot P_{idle} \qquad (1)$$

Because all the components that are used in the application we consider remain powered on (i.e. sleep/idle or active mode), we can state that

$$\tau_{active} = 1 - \tau_{idle} \qquad (2)$$

6.1.1 Setup traditional architecture

In a traditional (CPU-centric) architecture the general-purpose processor controls the media streams of an application. The general-purpose processor in such an architecture is responsible for the communication protocol to receive the MPEG frames from the

wireless interface, it needs to decode the frames, and also must transfer the data to an audio module.

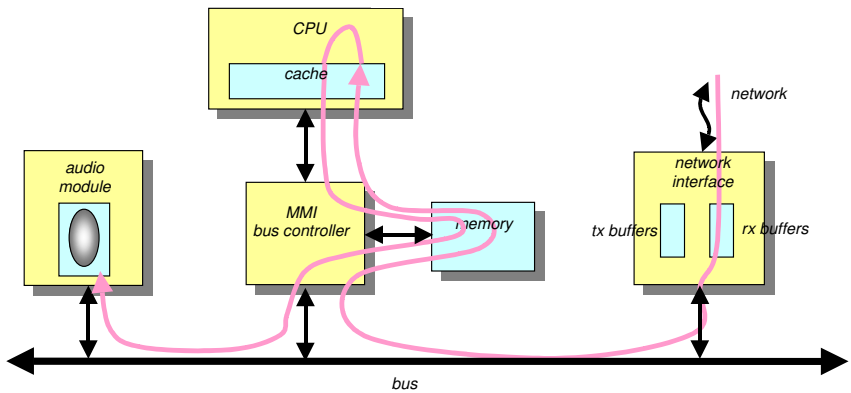


Figure 1: Data-flow through a traditional architecture.

The *traditional architecture* of a mobile, shown in Figure 1, is centered around a general-purpose processor with local memory and a bus that connects peripherals to the CPU. The long arrow in the figure indicates the data stream through the system when data arrives from the network, is transferred through the receive buffers on the network interface, copied to the 'main' memory, and then processed by the application (i.e. MP3 decoding). After the data is processed by the application, the data will traverse via 'main' memory, over the bus, to the output device (the audio module). In general additional bus transfers between CPU and memory are introduced while traversing several protocol layers (e.g. for data conversion of the packets like Ethernet to IP, and subsequently IP to TCP).

The power consumption *P* of the various parts of concern can be found in Table 1.

Table 1: Power dissipation of various parts in a traditional architecture.

Device	Specification	P active [mW]	P sleep/idle [mW]
Processor	Mobile Pentium II/400 [10]	7500	500
Network	WaveLAN modem 2.4 GHz [11]	1800 (RX)/1825 (TX)	180
Memory	SDRAM 2x32 Mb (Micron) [9]	1188	30
Bus	Theoretical bus, similar performance as Octopus switch	1344	1344

The table shows the power dissipation of the Mobile Pentium II processor. This processor is designed for portable applications, and has special features for a low-power consumption. The wireless network interface is based on the WaveLAN I modem. This is the same module as currently being used in the Mobile Digital Companion's network interface. The module does not include the MAC protocol implementation and bus

interface logic. The additional power consumption for these parts will be ignored. The energy required to transfer data over a bus is based on the theoretical values derived in Chapter 3. This theoretical bus has a similar performance as the Octopus switch. If we would have used for example a PCI-bus, then the required energy would be much higher. For example, the 'low-power' PCI9060 PCI bus master from PLX technology requires 680 mW [11].

6.1.2 Setup Mobile Digital Companion

In the architecture of the Mobile Digital Companion as discussed in Chapter 3 the general-purpose processor does not take part in the actual application. A dedicated MPEG audio decoder is being used to decode the MP3 data traffic. The general-purpose processor is only being used to initialise the connections, and setting up the Octopus switch. The processor will thus not be incorporated in the calculations of the power consumption. The network interface used is the testbed-interface board as described in Chapter 5. The medium access protocol is E^2MaC. Data coming from the wireless network interface is packet by the base-station into ATM cells, with a Virtual Connection Identifier (VCI) indicating the destination of the data. The Octopus switch will use this VCI to forward received packets of that connection from the network interface to the audio module.

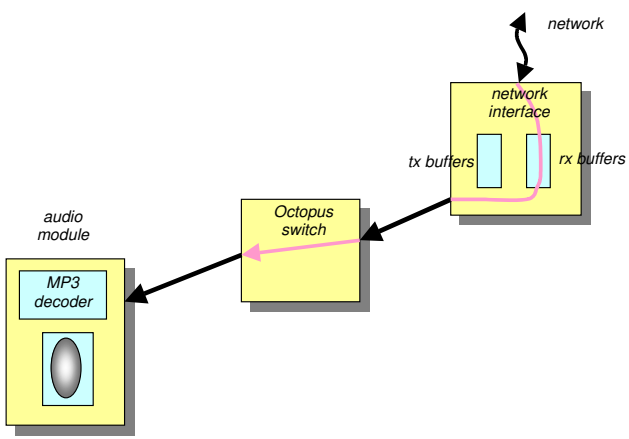


Figure 2: Data-flow through the Mobile Digital Companion.

The following table shows the power consumption *P* of the various parts of concern in this setup.

Table 2: Power dissipation of various parts in the Mobile Digital Companion.

Device	Specification	P active [mW]	P idle [mW]
MPEG decoder	Cirrus Logic EP7209 (sample frequency > 24 kHz) [3]	110	0.01
Network	WaveLAN modem 2.4 GHz [11]	1800 (RX)/1825 (TX)	180
	Buffer (SRAM HM628512) [6]	300	0.01
	Controller FPGA XC4010 (20 MHz)	120	50
switch	Octopus switch 32 Mb/s	150	60

The MPEG decoder is based on the Cirrus Logic EP7209, which is a single chip MPEG layer 2/3 audio decoder. The conversion from ATM to the required bit-stream can be done easily within the Module Interface Controller of the Octopus switch, and the power consumption involved can be neglected. The wireless network is based on the testbed network interface module that comprises basically of the WaveLAN modem, static RAM, and a controller FPGA. All these components are not low power, and can in a production implementation be replaced by dedicated low-power components.

6.1.3 Power dissipation MP3 application

To calculate the power dissipation involved in our MP3 application, we must incorporate the actual duty cycle of which the various parts are operating.

Traditional design

A MP3 audio stream with a sample frequency of 44.1 kHz, 16 bits, has a bit-rate of 128 kb/s. In the traditional architecture both the processor and the wireless interface have to be continuously in an active operating mode to be able to handle this data-stream[1].

The memory will be accessed multiple times for the processing of the communication protocols (we assume 7 times [11]) and also for decoding of the data-stream (one write and one read on an approximately 10 times larger data-size). However, the duty cycle of the memory remains low (active for approximately 10.8%) when we use a 32-bits wide memory bus, with a total access time of 50 ns. The power dissipation for memory access then becomes 216 mW. The bus interface is assumed to be active all the time.

Mobile Digital Companion

In the Mobile Digital Companion architecture, the network interface can be in idle mode for a significant time. The power cannot be turned off completely because the power-on time of the WaveLAN modem is 200 ms. We have assumed that the base-station transmits the data in bursts of 50 ATM cells (which would fill 50% of a typical frame in

[1] At least the inactivity threshold will not be reached, and thus the device will not enter an idle or sleep mode.

E^2MaC with a frame rate of 50 Hz). This requires the modem to receive 7 frames per second. If we would use a larger burst-size, then we would be able to turn off the modem completely from time to time (instead of entering sleep mode) to save more energy. Incorporating the additional overhead (receiving the traffic control slot, and the transitions from changing between receiving and idle modes), the duty cycle of the modem becomes then 8% active, and 92% idle. This implies a power dissipation of 310 mW. The memory of the network interface has a much lower duty cycle (3.2% active, 96.8% idle).

The Octopus switch can handle the required data rate easily and is thus idling most of the time (99.6 % idle, 0.4% active), implying a power dissipation of 60 mW.

The MPEG decoder will be continuously active and thus dissipates 110 mW.

Comparison

Figure 3 shows the power dissipation of the two architectures with the various parts involved in the MP3 application.

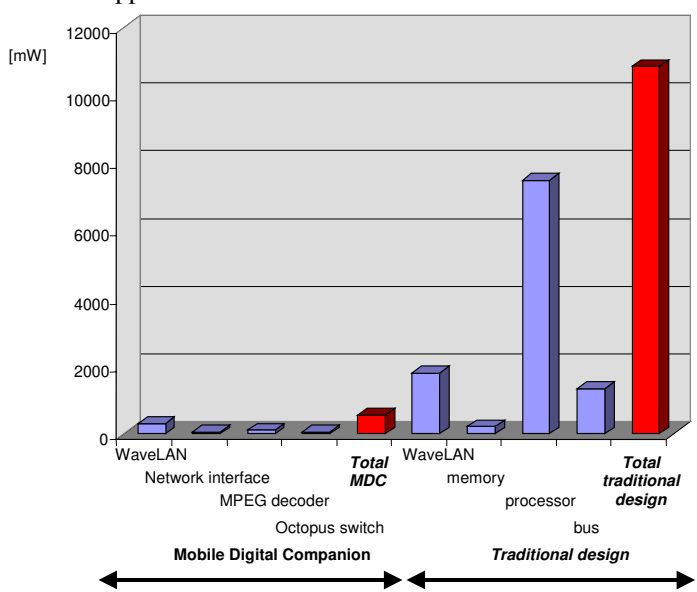


Figure 3: Power dissipation MDC and traditional design with during MP3 decoding.

Clearly shown is that the traditional architecture has a much higher power dissipation than the MDC architecture (i.e. 20.6 times higher).

Power dissipation breakdown

Figure 4 shows the resulting power dissipation of the various parts for the MP3 application in the MDC architecture.

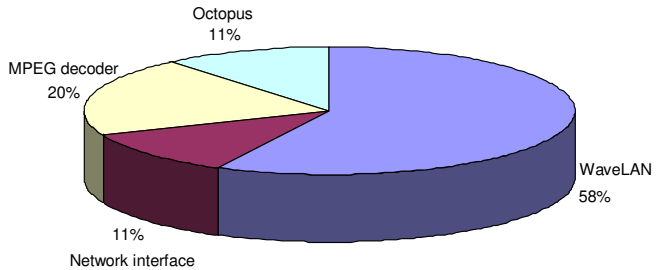


Figure 4: Power dissipation breakdown MDC when decoding MP3.

The energy required for the wireless communication takes a significant part (58% + 11%) of the total energy consumption of the MDC. Note that the WaveLAN modem is not optimised for hand-held devices.

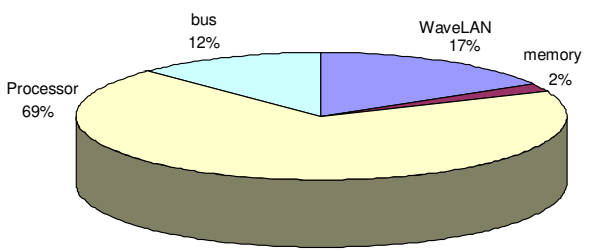


Figure 5: Power dissipation breakdown traditional architecture when decoding MP3.

In the traditional architecture, of which the power dissipation breakdown is shown in Figure 5, the wireless communication even has a much higher power dissipation (i.e. 1800 mW, instead of 430 mW), but still is not the most dominant. Even the power dissipation due to memory access is relatively low. Most of the power dissipation is due to the processor handling the communication protocols (IP, TCP, etc.) and decoding the MP3 data-stream.

6.1.4 Power dissipation when idling

It is expected that the hand-held will be idling for a significant time during the day. Most of the time the system will be waiting for an external event (like an incoming data packet from the wireless interface or the user pressing a button). The power dissipation during these idle periods will thus be an important factor to determine the lifetime of the batteries.

Traditional design

In a traditional design the network module must be powered on all the time because the protocol is usually based on a broadcast mechanism. Measurements on our WaveLAN system indicates that there are many (short) broadcasts (with a frequency of approximately 10 Hz). The power dissipation will thus be 1800 mW because the receiver has to be 'on' all the time. This broadcast mechanism also implies that the processor will be interrupted frequently. We will assume that this processing will take 0.1% of its time, thus $\tau_{active} = 0.1\%$. The resulting power dissipation is then approx. 507 mW.

The bus is not allowed to sleep. The memory will be idling most of the time, which results in a power dissipation of 30 mW.

Mobile Digital Companion

The network protocol is based on E^2MaC which implies that the network interface has to be turned on for a short time to receive the Traffic Control Slot (TCS). The frequency of which the TCS has to be received depends on the application. Here we will assume an interactive application (like waiting for an incoming phone call) that needs to receive the TCS twice per second. The active time is then approximately 0.1%, which implies a power dissipation of 182 mW. Since we are also capable of powering down the whole modem, we can even further reduce the power consumption to approx. 73 mW (incorporating the power-up latency of 200 ms). This once again shows that the power-up latency is an important factor in the power dissipation. In future modern designs this will be an important design issue. The network interface contributes with approximately 50 mW (idling).

The Octopus switch consumes 60 mW when idling. All other parts of the system can be turned off.

Comparison

Figure 6 shows the power dissipation of the two architectures when idling.

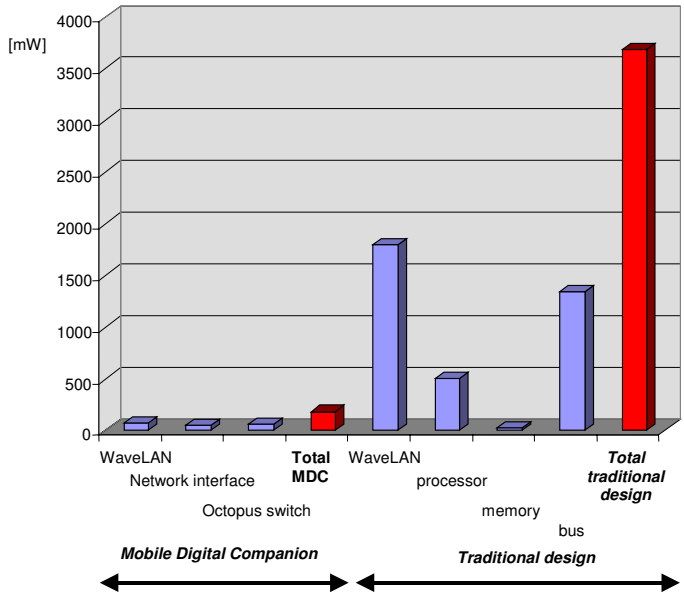


Figure 6: Power dissipation when idling.

Note that the power consumption of the MDC is constituting mainly of idling components. Because in our test-bed we have not used low-power components, the resulting power dissipation is much higher than could be expected when using components that are designed for a low idle power consumption.

6.2 Future research

A thesis is never finished. Although in this thesis we have presented solutions to a number of problems in the field of mobile multimedia computing, many others have remained unsolved or received only minor attention. This section attempts to give a few suggestions for future research.

Having an energy efficient architecture that is capable to handle adaptability and flexibility in a mobile multimedia environment requires more than just a suitable hardware platform. Therefore, 1) more research is required in the operating system architecture that needs to deal with the hardware platform and the adaptability and flexibility of its devices, 2) we need to provide support for reconfigurable computing, and 3) we need to have a model of all resources in the mobile computer so that we can design a proper energy management system.

Research in these items will be continued in the Moby Dick project. The *Chameleon* project [12] will in particular perform research in reconfigurable computing for these systems.

6.2.1 Operating system architecture

Our design of the architecture is geared toward achieving low energy consumption, while achieving good application performance like high throughput and low end-to-end latency. In meeting these requirements we have to place constraints on the operating system and communication protocols. The operating system for the *Mobile Digital Companion* has to deal with the flexibility and adaptability of its devices. Since operating system support is out of the scope of this thesis, we will only mention the main relevant topics. The operating system being used for the experiments is derived from Lucent Technologies Bell Labs' *Inferno* [4] and must adapted in several ways:

- The operating system must be adapted to the hardware architecture: it must be able to manage *connections* between modules rather than DMA transfers, and it must provide a suitable interface to change and manage the functionality of the programmable modules. The operating system has to provide the service such that applications can make effective and efficient use of programmable hardware.
- Applications can exhibit different behaviour depending on its QoS-level and operating conditions (energy resources, current network conditions, etc.). The operating system must define and manage a global energy policy, it must manage the required *adaptation* and, of course, inform applications of changes in their environment. Energy state transitions should never compromise functionality.
- It must provide provisions for *distributed processing* in which part of the application or service will be (possibly even dynamically) reallocated and performed on a remote server when this is more efficient.
- Since it is not feasible to try to adapt the current *communication protocols*, and adapt all current systems and applications, our target will be to support existing and widely used protocols. However, in our connection-centric model in which data streams flow directly from source to sink without interference from a CPU, we believe that is neither feasible nor efficient that all functional modules implement the whole communication protocol stack of several protocols. Furthermore, most existing protocols are not designed to operate over a wireless link, and do not provide any, or not efficient support for mobile computers. The effect of these protocols is that they waste energy and often have a poor performance and a poor quality. Therefore, we should adopt an distributed processing style in which parts of the communication protocol will be handled by a remote server (most probably the base station to which the mobile is connected to at that moment). The mobile will be relieved from the task of handling ‘standard’ protocols and will use an internal dedicated and efficient protocol to communicate with the base-station. The resulting architecture can be viewed as a packet routing system that supports data transfer between modules and the wireless link without microprocessor intervention. In MOBY DICK both base stations and mobiles use the operating system *Inferno*. The

operating system allows application and system functions to be split and migrated easily. The base station will handle the TCP/IP protocol in lieu of the mobiles, and use an internal dedicated protocol with the mobile to transfer the packets [1].

These adaptations are necessary and worthwhile because they lead to significantly lower energy consumption while the performance is high. However, a secondary goal is that when our system will be used with current applications, protocols and operating systems with no modification, then our system should not perform worse than the existing systems. For these applications, and to be able to develop and test new applications, the general-purpose processor can play an important role.

Another important issue related to the operating system is that applications have a clean and simple interface. Only when application programmers are willing to use the possibilities that our system provides, and can easily use the programmers interface, then the system can show its real value.

6.2.2 Reconfigurable computing

Reconfigurable computing systems combine programmable hardware with programmable processors to capitalise on the strengths of hardware and software [8]. The earliest configurable computing machine was proposed, designed, and implemented by Professor Gerald Estrin at UCLA in the early 1960s [5]. Estrin proposed the "Fixed plus variable structure computer", where some fixed hardware was dedicated to an inflexible abstraction of a programmable processor and a flexible component implemented digital logic. Today, the most common devices used for reconfigurable computing are *Field Programmable Gate Arrays* (FPGA). FPGAs present the abstraction of gate arrays, allowing developers to manipulate flip-flops, small amounts of memory, and logic gates.

Currently, many reconfigurable computing systems are based on FPGAs. However, these systems have a number of limitations [7]:

- *Limited functionality* – Not all computations can be implemented efficiently with today's FPGAs: they are well suited to algorithms composed of bit-level operations, but they are ill suited to numeric operations, such as high-precision multiplication or floating point calculations. General-purpose processors (including digital signal processors) use optimised function units that operate in bit-parallel fashion on long data words. Compared with general-purpose processors, FPGAs are inefficient in performing ordinary arithmetic operations.
- *Gate capacity* – Available FPGAs provide an equivalent of 10K to 1000K gates. These devices are often large enough to experiment with the basic strategies, but limit the scope of the designs. Future FPGAs will be much larger and will have much broader application, including highly complex communications and signal-processing algorithms.
- *Configuration speed* – Most existing FPGAs use relatively slow paths for device configuration, and few have the ability to reconfigure only selective parts of the device. The configuration speed determines the characteristics of the computation

model: it should change frequently enough to take advantage of programmability, but slowly enough to mask hardware configuration time.

- *Memory structures and interface* – FPGAs currently provide little on-chip memory for storage of intermediate results in computation; thus many reconfigurable computing applications require large external memories. The transfer of data to and from the FPGA increases energy consumption and may slow down the computations.

Although less energy efficient than application specific integrated circuits, reconfigurable devices, such as field programmable gate and function arrays can be used for implementing customised circuits. These technologies allow circuits to be created and removed on demand: instead of including all the customised circuits in a system, only those components required for a particular algorithmic stage need to be present. When finished they can be replaced by other customised hardware following a hardware reconfiguration at run time. This capability is especially relevant for wireless networks, where operating conditions are often unpredictable and protocol standards may vary from one network to another.

The *Mobile Digital Companion* has a hierarchical-granularity architecture. The programmability granularity of the modules of the *Companion* is coarse, and the modules themselves can be programmed more fine-grained. Most of the modules of the Companion include programmable hardware that can be used to implement the various circuits. Currently there is a large gap between the hardware devices and the application. Research that is needed in this area includes design languages, development methods, as well as compile-time and runtime environments and operating system support for such systems.

6.2.3 Modelling energy management

Applications that users run on a mobile need several functional resources of the system, such as processor, memory, wireless network interface, compression/decompression logic etc. In the Companion's architecture we assume that such modules are programmable and can adapt to the demands of the applications and to the state of the environment, e.g. available bandwidth, bit error rate, available energy, etc. In general these modules are not independent and choices for the setting of one module may influence other modules. For example: when video has to be transmitted it can be compressed, which reduces the required bandwidth on the wireless network. However, more compression requires not only more processing power, it also needs better error-control. All these functional modules often have contradictionary effects on the resources needed, and a trade-off has to be made to find an optimal solution. Not only the parameters can be changed, it might as well be profitable to migrate complete function from one module to another, possibly even to another machine. A complicating factor is that a wireless environment is very dynamic, so it is not feasible to search for *the* optimal solution.

Adaptability and flexibility are two recurring items when we mention energy efficiency and performance on mobile multimedia computers. The architecture of the *Mobile*

Digital Companion has many ways in which adaptation can be applied. This leads to a key problem of *policy optimisation*, which must be the central issue in any energy management system. The policy is the algorithm that decides what measures have to be taken to minimise the energy consumption. Traditional power management schemes only decides how and when to activate or shut down system resources to minimise the energy consumption, depending on usage patterns and performance constraints.

Currently several system developers and vendors are pursuing a long-term, wide scope strategy (ACPI and OnNow) to greatly simplify the task of large and complex power-managed systems. However, both ACPI and OnNow assume a CPU and operating system centric system, where the activities of the system are managed by a single entity. Furthermore, ACPI and OnNow are developed to support the implementation of power managed computer systems, and are too detailed to effectively support design exploration [2].

In the early phases of the design of any part of the system, either hardware or software, the designer needs to experiment with alternative designs. However, energy efficiency is not only a one-time problem that needs to be solved during the design phase. When the system is operational, frequent adaptations to the system are required to obtain an energy efficient system that can fulfil the QoS requirements imposed. Finding the energy management policy that minimises energy consumption without compromising performance beyond acceptable levels is already a complex problem. If the resources are also flexible, and can adapt their functionality, this problem becomes even bigger.

To be able to make valid decomposition that satisfies many requirements and provides an efficient solution, more research is needed to 1) provide support for early system level architectural exploration of energy-managed systems, and 2) to provide a model that can be used to manage adaptable entities at various levels of the architecture.

6.3 Conclusion

In this thesis we considered the problem of designing an architecture for a mobile multimedia computer. The requirement of portability of hand-held multimedia computers and portable devices places severe restrictions on size and energy consumption. In its most abstract form, a mobile computer system has two sources of energy drain during operation: communication and computation. Broadly speaking, minimising energy consumption is a task that will require minimising the contributions of communication and computation, making the appropriate trade-offs between the two.

Even though battery technology is improving continuously and processors and displays are rapidly improving in terms of power consumption, battery life and battery weight are issues that will have a marked influence on how hand-held computers can be used. These devices often require real-time processing capabilities, and thus demand high throughput. The increasing levels of performance and integration that is required will be accompanied by increasing levels of energy consumption. Without a significant energy reduction techniques and energy saving architectures, battery life constraints will limit

the capabilities of these machines. More extensive and continuous use of network services will only aggravate this problem since communication consumes relatively much energy.

As the mobiles must remain usable in a wide variety of environments, they must be flexible enough to accommodate a variety of multimedia services and communication capabilities and adapt to various operating conditions in an (energy) efficient way.

We have shown that it is not sufficient to simply continue advancing our chip architectures and technologies as just more of the same: building microprocessors and devices that are simply more complicated versions of the kind built today. We use the technology and the abundant logic gates to built an architecture that is capable of processing multimedia applications that operate in the dynamic mobile environment. Key issues in this are *energy efficiency* and *Quality of Service*.

Main principles – We have found that there are two main principles that can be used when designing mobile multimedia systems.

1. *System-wide layer integration/co-operation*. Co-operation or integration of the various layers significantly improves energy efficiency of the system because it reduces waste and data streams retain a high locality of reference.

   The art of low-power design used to be a narrow speciality in analog-circuit design. As the issue of energy efficiency becomes even more pervasive, the battle to use the bare minimum of energy will be fought on multiple fronts: semiconductor technology, circuit design, design automation tools, system architecture, operating system, and application design. We have shown that there is a vital relationship between hardware architecture, operating system architecture and applications architecture, where each benefits from the others. In our architecture we have applied several supplementary energy-reduction techniques on all levels of the system. Achieving high energy efficiency requires first of all *the elimination of the waste* that typically dominates the energy consumption in general-purpose processors. The second main principle used is to have a *high locality of reference*. The philosophy is that all operations that are required on the data should be done at the place where it the most efficient, thereby also minimising the transport of data through the system.

2. *Use a Quality of Service framework*. We have demonstrated in our research and in particular in the design of a system architecture, a switching network, and the wireless network design, that Quality of Service is not only important to provide an adequate level of service for a user, but can also be used as a tool to achieve an energy-efficient system. Users and applications request a certain QoS level. The system then operates in such a way that it will try to satisfy these requirements, but never gives more quality than required and necessary. Adaptability is the basic mechanism to achieve this.

Of particular importance to the system architecture is the interconnection structure that connects the application domain specific modules. The system architecture of the *Mobile Digital Companion* is connection centric, which means that the media type of the traffic drives the data flow in the system using *connections*. In our infrastructure all

connections are identified with a connection identifier which is used to identify the type of data, and the module destination address. This identifier provides the mechanism to support lightweight protocols that provide data-specific transport services that are associated with a certain QoS. This approach not only eliminates the need to transfer a large number of address bits per access, it also gives the system the possibility to control the QoS of a task down to the communication infrastructure.

The wireless network is another important aspect of a mobile multimedia system. We have shown that energy-awareness must be applied in almost all layers of the network protocol stack. To achieve maximal performance and energy efficiency, *adaptability* is important, as wireless networks are dynamic in nature. Furthermore, if the application layer is provided with feedback on the communication, advantage can be taken from the differences in data streams over the wireless link. To allow this, feedback is needed from many layers: the physical layer provides information on link quality, the medium access layer on effectiveness of its error correction, and the data link layer on buffer usage and error control.

Although our testbed currently consists of various small printed circuit boards containing Field Programmable Gate Arrays (FPGAs), microcontrollers, and memory, the complexity of these designs is low. This low complexity will make it possible to transfer the architecture to a (large) custom IC.

The lessons learned from the design of this architecture serve as a first step towards a system-level design of an energy-efficient mobile multimedia computer.

References

[1] Balakrishnan H., Padmanabhan V.N., Seshan S., Katz R.H.: A comparison of mechanisms for improving TCP performance over wireless links", *ACM SIGCOMM'96*, Stanford, August 1996.

[2] Benini L., De Micheli G.: "Dynamic Power Management, design techniques and CAD tools", *Kluwer Academic Publishers*, ISBN 0-7923-8086-X, 1998.

[3] Cirrus Logic EP7209, Ultra-low-power audio decoder system-on-chip, http://www.cirrus.com.

[4] Dorward S., Pike R., Presotto D., Ritchie D., Trickey H., Winterbottom P.: "Inferno", *Proceedings COMPCON Spring'97*, 42nd IEEE International Computer Conference, 1997, URL: http://www.lucent.com/inferno.

[5] Estrin G. "Organization of Computer Systems: The Fixed-plus Variable Structure Computer", *Proceedings of the Western Joint Computer Conference*, pp. 33-40, 1960.

[6] Hitachi, "HM628512 Series, 524288-word ´ 8-bit High Speed CMOS Static RAM", 1995.

[7] Mangione-Smith W.H., et al.: "Seeking solutions in configurable computing", *IEEE Computer*, pp. 38-43, December 1997.

[8] Mangione-Smith W.H., Hutchings B.L.: "Configurable computing: the road ahead", *1997 reconfigurable architectures workshop*, 1997.

[9] "16 MEG x 32 SDRAM DIMM", http://www.micron.com. Micron Technology Inc., 1999.

[10] Mobile Pentium II, http://www.intel.com/mobile/pentiumII.

[11] PLX technology: "PCI9060, PCI Bus master interface chip for adapters and embedded systems", datasheet, 1995, http://www.plxtech.com/download/9060/datasheets/9060-12.pdf.

[12] Gerard J.M. Smit, Martinus Bos, Paul J.M. Havinga, Sape J. Mullender, Jaap Smit: "Chameleon - reconfigurability in hand-held multimedia computers", *proceedings First International Symposium on Handheld and Ubiquitous Computing*, HUC'99, September 1999.

[13] Steenkiste P.: "Design, implementation and evaluation of a single-copy protocol stack", *Software – practice and experience*, January 1998.

[14] WaveMODEM 2.4 GHz Data Manual, Release 2, AT&T 1995.

Appendix A

Energy efficiency of error correction for wireless communication

Since high error rates are inevitable in the wireless environment, energy efficient error control is an important issue for mobile computing systems. When error-correction mechanisms are implemented on general-purpose processors the power consumption that is required to perform the error-correction mechanism can be significant. To illustrate this we have studied two different error correction mechanisms with different characteristics and capabilities, i.e. EVENODD and Reed-Solomon[1].

A.1 Introduction

Traditionally, network protocols have been designed around the conventional metrics of throughput and latency. However, a proper design of these protocols offers many opportunities for optimising the design metric that is more relevant to battery operated devices: the amount of energy consumed per useful user level bit transmitted across the wireless link. Since high error rates are inevitable to the wireless environment, *energy-efficient error-control* is an important issue for mobile computing systems. This includes energy spent in the physical radio transmission process, as well as energy spent in computation, such as signal processing and error control at the transmitter and the receiver.

In communication systems forward error correcting (FEC) codes are used to protect packets of data that are transmitted over some network. Error-control mechanisms traditionally trade off complexity and buffering requirements for throughput and delay [4][11][12].

Error correcting codes are generally applied at several layers in the communication protocol stack. Error control at the lower layers (physical, data link layer) is often implemented with dedicated hardware and embedded software. We will concentrate on error correction mechanisms for the higher protocol layers, which are mostly

[1] Major parts of this chapter have been presented at the *IEEE Wireless Communications and Networking Conference 1999* [7].

implemented in software. Dedicated hardware may be used to implement (parts) of the error control (as described in Chapter 5), though in a system, which requires the flexibility to alter error-control schemes on a stream by stream basis, a software solution may be preferable.

At these layers the error correction mechanism operates on relative large blocks. Generally, block codes such as Bose, Chaudhuri and Hockuenghem (BCH) and Reed-Solomon codes require a decoder capable of performing arithmetic operations in finite fields [14]. A comparison between application-specific integrated circuit (ASIC), FPGA, and digital signal processing (DSP) implementations of the decoder shows that the performance of FPGA-based designs lean more toward that of ASICs, but retain flexibility more like DSPs [3][6]. Unfortunately, good VLSI designs for codes using BCH or Reed-Solomon codes do not map well to FPGAs [1]. A code that does not require finite-field arithmetic, is the *EVENODD* code [2]. The EVENODD code was originally designed for a system of redundant disks (RAID).

When error-correction mechanisms are implemented on general-purpose processors the power consumption that is required to perform the error-correction mechanism can be significant. We have studied a software implementation of the EVENODD error correcting mechanism, and compared it with an implementation of the Reed-Solomon mechanism.

The total energy consumption per useful bit will depend both on the energy of transmission and the energy of redundancy computation. We will show that the computational cost associated with FEC cannot be ignored, constituting a significant portion of the overall energy cost. Furthermore, the trend has been toward smaller communication cells, e.g. with the size of an office room, thus requiring lower transmit power. The ratio of computational to transmit power under these circumstances is therefore only likely to increase.

A.1.1 The encoding packet model

The basis for most currently designed wireless systems is packet switching, which manages data transfer in blocks (*packets*) that contain multiple symbols (or bits). The size of a packet is in principal not related to the actual amount of data transmitted over the channel in a MAC frame. Errors are assumed to be detected by some detection technique (e.g. by using Cyclic Redundancy Check (CRC) data), and the whole packet will be discarded. The residual channel characteristic after the physical and link layer processing is then based on *erasures*, i.e. missing packets in a stream [17]. Figure 1 shows a graphical representation of the error correction mechanism. The sender collects a number of *source data packets* in a buffer. When the buffer is full, the data is encoded, and the encoded data is transmitted. The receiver is able to reconstruct the original data from a subset of the encoded data, and so can allow the erasure of some packets.

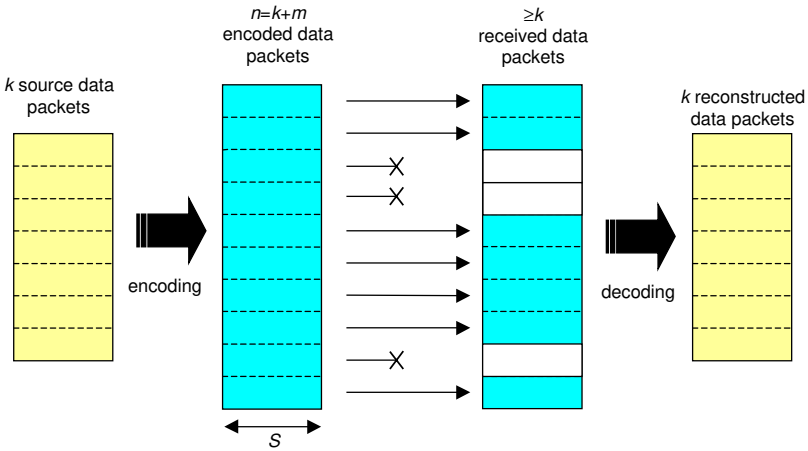


Figure 1: Graphical representation of error correction.

We will denote the number of source packets as k, the packet size as S, the number of redundant packets m, and the number of encoded packets as n. Such a code is called an (n,k) code and allows the receiver to recover from m $(=n-k)$ losses in a group of n encoded packets. This structure can be seen as an (S) x $(k + m)$ array in which the columns represent a packet of length S, the first k columns represent the source data packets, and the last m columns represent the redundant packets. All packets together build up one *frame*. Figure 2 gives a graphical representation of this scheme.

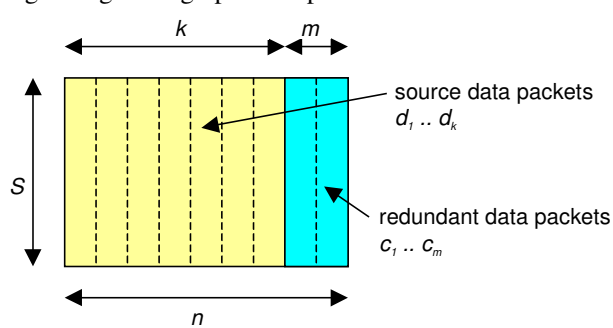


Figure 2: Representation of an encoded frame.

A general technique for tolerating m simultaneous failures with m redundant packets is a technique based on *Reed-Solomon* coding [14]. This technique requires computation over finite fields and results in a complex implementation. An alternative might be a scheme like EVENODD that only requires simple exclusive-OR operations and that it is able to tolerate two erasures. In the following section we will give an overview of the EVENODD and Reed-Solomon coding and determine the energy efficiency of these mechanisms.

We define the *energy efficiency e* as the amount of data processed divided by the energy that is consumed to process that data:

$$e = \frac{\textit{Amount of data}}{\textit{Energy consumed to process the data}} \qquad [J^{-1}] \qquad (1)$$

One can view this as the inverse of the cost (in terms of energy) of calculating the redundancy to be transmitted over a channel.

A.1.2 Reed-Solomon coding

The Reed-Solomon coding scheme is an (n,k) code. There are three main aspects involved with the Reed-Solomon algorithm: the use of the Vandermonde matrix to calculate the redundant packets with simple matrix arithmetic, the use of Gaussian elimination to recover from failures, and the use of Galois fields to perform arithmetic [16].

A major concern is that the domain and range of our computations are binary words of a fixed length w. Since practical algebra implementation does not use infinite precision real numbers, we must perform addition and multiplication over a *finite field* of more than $k + n$ elements. Fields with $q = p^w$ elements, with p prime and $w > 1$ are called extension fields or Galois Fields denoted as $GF(p^w)$. Operations on extension fields are simple in the case $p = 2$. The elements of $GF(2^w)$ are integers from zero to $2^w - 1$. Addition and subtraction of $GF(2^w)$ are simple exclusive-OR operations. Multiplication and division are more complex and require two mapping tables, each of length 2^w. These two tables map an integer to its logarithm and its inverse logarithm in the Galois field. A table for the multiplication can be used as well if the number of field elements is not too large. Note that a multiplication in $GF(2^8)$ already requires a 64 kB lookup table! However, a Reed-Solomon coding implementation that is not parameterised (i.e. n and k are fixed) can be implemented much more efficient [9].

Energy efficiency of the Reed-Solomon algorithm

The *encoding overhead* depends on the number of source packets k, on the number redundant packets m $(= n - k)$ and on the size S of a packet. The encoder requires k source data packets to produce each encoded packet, and thus the encoding overhead to process k source packets is $O((n-k).k.S)$. Therefore, an approximation of the energy efficiency of encoding e_{rse} equals:

$$e_{rse} = E_{rse} \frac{k}{(n-k)\,k} = E_{rse} \frac{1}{m} \qquad (2)$$

in which E_{rse} is the efficiency for encoding with Reed-Solomon. The value is determined by the implementation and is dependent on the packet size S.

The *decoding overhead* is more complicated as it involves two parts: the Gaussian elimination, and the reconstruction. This requires a matrix inversion to be performed

once, and then a matrix multiplication for each reconstructed packet which is maximal m. Although the matrix inversion requires $O\ (k.(n-k)^2)$ operations per k packets, the cost of matrix inversion becomes negligible for reasonable sized packets (matrix inversion is not required for a non-parameterised implementation with $m \leq 2$ like EVENODD). In the experiments this will be shown clearly. The matrix multiplication requires $O(k)$ operations for each reconstructed data item, or a total of $O((n-k).k.S)$ operations per block of k packets. So, if we assume the number of reconstructed packet to be equal to $(n-k)$ then an approximation of the energy efficiency of decoding e_{rsd} equals:

$$e_{rsd} = E_{rsd} \frac{k}{(n-k)\,k} = E_{rsd} \frac{1}{m} . \qquad (3)$$

in which E_{rsd} is the efficiency for decoding with Reed-Solomon. The value is determined by the implementation and is dependent on the packet size S.

A.1.3 EVENODD coding

The EVENODD coding scheme is an $(k+2,k)$ code. It was originally meant for tolerating two failures in RAID architectures, but we will show that it is also suitable in communication systems. The basic scheme requires the number of source packets k to be a prime number. If we want to use a non-prime number for k, then we can take the next prime following the required k, and assume the extra imaginary packets to contain zeros. The packet size S is for simplicity restricted to contain $(k - 1)$ symbols. This restriction is not hard too because a symbol can be of any size, and also because we can introduce imaginary symbols to fill the columns to the required size.

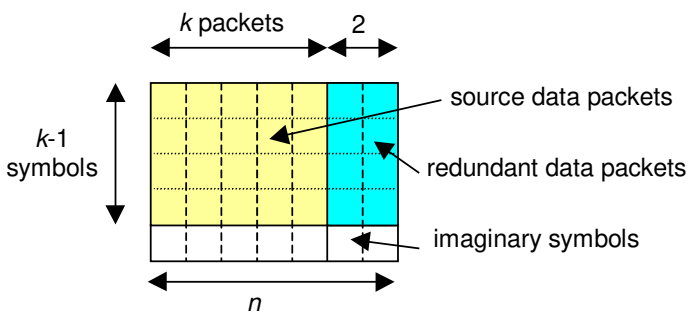


Figure 3: Basic EVENODD frame.

So, the packet model can be considered as an $(k - 1)$ x $(k + 2)$ array, k a prime number, such that the symbol d_{ij}, $0 \leq i \leq (k - 2)$, $0 \leq j \leq (k - 1)$, is the i-th symbol (row) in the j-th packet (column). The last two packets (k and $k + 1$) are the redundant packets. One imaginary row (k-1) is added containing zeros.

Data encoding

There are two types of redundancy: *horizontal* redundancy and *diagonal* redundancy. The redundant value of each is stored in a redundant packet. The value of the horizontal redundancy (stored in packet k) is the exclusive-OR of packets 0, 1, …, k –1. This is thus exactly the same as with simple parity encoding. Packet (k+1) carries a diagonal redundancy. This is calculated using the exclusive-OR of the diagonals of the matrix and P. P is calculated via the exclusive-OR of a special diagonal. So, for example the first redundant symbol in redundant packet k +1, denoted as $d_{0,k+1}$, is calculated with: $d_{0,k+1} = P \oplus d_{0,0} \oplus d_{k-1,1} \oplus d_{k-2,2} \oplus \ldots \oplus d_{1,k-1}$. Since the source packet matrix is an (k – 1) x (k) matrix, one diagonal is not calculated. This diagonal (formed by the indices (k-1,0), (k-2,1), (k-3,2), …, (0,k-1)) is used to determine the value of P.

	0	*1*	*2*	*3*	*k-1*	*k*	*k+1*
0	1	0	1	1	0	1	0
1	0	1	1	0	0	0	0
2	1	1	0	0	0	0	1
k-2	0	1	0	1	1	1	0

Figure 4: EVENODD coding example for k=5.

An example of an encoded frame with symbols of one bit is shown in Figure 4. Notice that (without the imaginary row k-1)

$P = d_{3,1} \oplus d_{2,2} \oplus d_{1,3} \oplus d_{0,4}$. , and e.g. $d_{2,6}$ is obtained as follows: $d_{2,6} = P \oplus d_{2,0} \oplus d_{1,1} \oplus d_{0,2} \oplus d_{3,4}$.

Data recovery

Data encoded with the EVENODD scheme is able to recover maximal two packet erasures. This equivalents to Reed Solomon encoding with n-k=2. Reconstruction when only one packet is erased (and assuming it is not a redundant packet) is simple as the missing packet can be retrieved using the exclusive-OR of the packets. When two packets i and j , $0 \le i < j \le k + 1$, are erased, then the decoding scheme is more complicated, but still requires only exclusive–OR operations. Note that recovering is also possible for finer grained erasures: i.e. not all erasures need to be in the same two packets. Depending on the topology of the symbol erasures up to $2(k - 1)$ *symbol erasures* can be restored [8]. A similar effect can be reached with Reed-Solomon coding when interleaving is used.

Energy efficiency of EVENODD coding

Encoding for the horizontal parity packet requires k exclusive-OR operations to be performed on each data symbol with size s. The diagonal parity requires $k + 1$ exclusive-OR operations, including the calculation of P. This makes the total encoding complexity

$O(2k+1)$. The amount of data encoded is $(k-1).s.k$. Using the packet size $S=(k-1).s$ we get $S.k$. Therefore, the energy efficiency e_{eoe} is:

$$e_{eoe} = E_{eoe} \frac{k}{2k+1} \qquad (4)$$

in which E_{eoe} is the implementation dependent efficiency for encoding EVENODD. The value of E_{eoe} is determined by the specific implementation in either hardware or software and is dependent on the packet size S.

The *decoding overhead* is dependent on the number of erased packets, which packets are erased (i.e. whether redundant packets are involved or not), the number of source packets k, and on the size s of a symbol. We will only deal with the complexity of the erasure of two data packets as this is the most most complex. This case has three main steps. First, calculating the diagonal parity P requires $O(2 . S)$ exclusive-OR operations. Then two syndromes are calculated, requiring $O(S . (k-1))$ and $O(S . k)$ XOR operations. Finally the reconstruction takes another $O(2 . S)$ XOR operations. This makes the total decoding complexity $O((2k + 3) S)$, which are basically all XOR operations. When E_{eod} is the efficiency for decoding EVENODD then the energy efficiency equals:

$$e_{eod} = E_{eod} \frac{k}{2k+3} \qquad (5)$$

The value of E_{eod} is determined by the specific implementation in either hardware or software and is dependent on the packet size S.

A.2 Implementation and results

A.2.1 Software implementation

In the next sections we show the results of a software implementation on a general-purpose processor of both error correction mechanisms. Such an implementation is the most flexible solution and can adapt its algorithm and its parameters very quickly to changing environments. Adaptive error correction has shown to be much better than a single code scheme in terms of utilised bandwidth and in terms of a profit function which combines the bandwidth utilised and the deadline miss rate [5].

We are aware of the fact that a software implementation is not the most energy-efficient solution, and might not provide enough performance. However, there exist many applications and systems that do not need high performance and cannot use the capabilities and advantages of dedicated hardware. For example, a notebook computer that lacks such dedicated hardware can also benefit from an energy-efficient solution, even if it is not the most optimal implementation.

A software implementation has a number of specific advantages compared to a hardware solution:

- The use of a microprocessor allows *very rapid adaptations* to varying error conditions (burst size, frequency) and required QoS from applications. The adaptation to perform can be applying another error-control scheme, or adapting some parameters of the error-control scheme.
- A software implementation allows us to experiment with a large set of error-control schemes, and experience in 'real life' how applications behave. When we have a good feeling of the behaviour of the schemes, then we could compose a subset of error-control schemes that is suitable to be implemented in hardware, either in an FPGA, a DSP, or a custom chip.
- The error control can easily and efficiently be embedded in various layers of the communication protocol where the data is buffered anyway. With a good engineered and well-integrated error correction mechanism little extra overhead is expected.
- A standard processor also allows the use of relatively large memories, and thus allows for much larger block lengths than standard custom chips (that typically allow a block length of up to 255 bytes [13]). In a wireless office environment burst errors of 1 to 100 ms can be expected. To handle these large erasures at relatively high speed (say 2 Mb/s), a large block size is needed.

We will assume that there is a linear relation between the energy consumed by the algorithm and the amount of time needed for the processor to do its calculations. Although this assumption introduces some inaccuracy (for example because this does not incorporate possible energy management mechanisms of the processor, the energy consumption of the memories being used for buffering, and the energy consumption of other parts of the computer system that also needs to be active), this still gives a good indication of reality [9]. Both implementations are written in *C* and are portable across many platforms. The implementation of the Reed-Solomon coding is flexible, it can be used for arbitrary n and k. The measurements were performed on a Toshiba 220CS notebook that has a Pentium Pro 133 processor and runs Windows 95. The results can only be used as a reference, since the actual performance depends on items like memory speed, cache size, quality of the compiler, operating system, etc. The code is written straightforward, and uses the most obvious optimisations only (like the use of a multiply lookup table for Reed-Solomon coding). Handcrafted code that makes good use of the specific features of the processor (like registers and the use of special instructions) might achieve significant speedups.

Using the timing measurements we can calculate the costs χ (in μs/byte) to encode and decode one byte. The efficiency E can then be defined as $1/\chi$, which is independent from the actual power consumption of the processor on which the coding was performed. The energy efficiency e incorporates the power consumption P of the processor, thus $e = 1 / \chi P$ (in Joule^{-1}).

A.2.2 EVENODD coding implementation

The data model that we have used in our implementation of the EVENODD coding resembles the basic model in which an imaginary 0-row is added to simplify the implementation [1]. So, we use a (k) x (k + 2) array, k a prime number, of symbols with size s. Each column represents a packet. The last two columns (k and k + 1) are the redundant columns. The symbols can be of any size, but normally are a multiple of a byte. Note that the values of the efficiency E_{eoe} and E_{eod} are independent from the power consumption of the processor.

Figure 5 and Figure 6 show the characteristics of the efficiency E_{eoe} and E_{eod} of Equation (4) and Equation (5) versus the number of source packets k, for various values of the symbol size s.

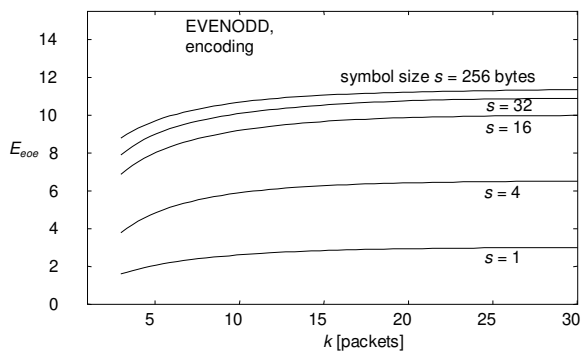


Figure 5: Efficiency E_{eoe} vs. k as a function of symbol size s.

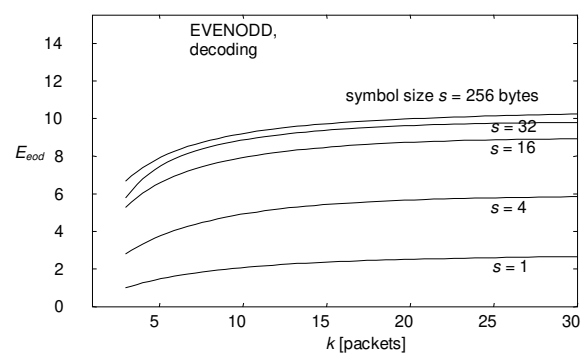


Figure 6: Efficiency E_{eod} vs. k as a function of symbol size s.

The effect of the implementation overhead gets less when the packet size over which the code has to work is enlarged, because it will be amortised over more data. A better performance is also reached due to the effect of caching, since the same data is used several times. In the constant region the encoding time on the Toshiba 220 CS is approximately 88 µs/kByte and the decoding time approximately 96 µs/kByte.

A.2.3 Reed-Solomon coding implementation

Our code of the Reed-Solomon coding is based on an implementation from Rizzo, Karn and others [18]. The data model that we have used in our measurements is an (*S*) x *(k + (n-k)*) array of symbols with size *s*. Each column represents a packet with size *S*. The last *n-k* columns are the redundant columns. The code supports $GF(2^w)$, for any *w* in the range of 2..16. In the measurements we have used $w = 8$. This gives the maximum efficiency because most operations can be executed using lookup tables [17]. So, the symbol size *s* in our measurements will be one byte. We chose *S* to be multiples of ATM cells sizes (53 bytes). We have used a lookup table for the multiply operations.

Figure 7 shows the characteristics of the efficiency E_{rse} and E_{rsd} (of Equation (2) and (3)) versus *k*, for various values of *S*. The energy efficiency of encoding is hardly influenced by the packet size or the number of source packets; therefore only one graph is shown in the figure for encoding. Encoding is already stable for small values of *k* and for all packet sizes.

Decoding is more influenced by the packet size. This is mainly caused by the cost of matrix inversion which cost $O(k \cdot l^2)$, where *l* is the number of packets which must be recovered (which we assume to be equal to *n-k*). The influence is small for packet sizes greater or equal to 8 ATM cells only.

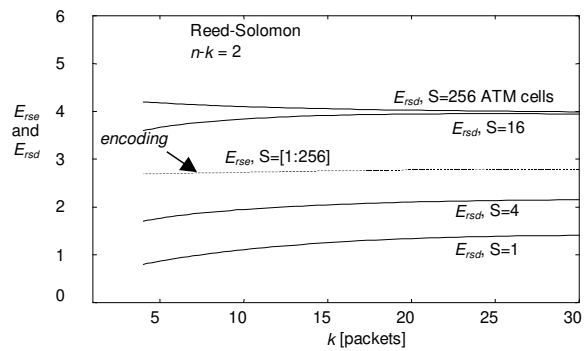


Figure 7: Efficiency vs. number of source packets as a function of packet sizes.

In the constant region the encoding time is approximately 365 µs/kByte and the decoding time is approximately 257 µs/kByte.

A.2.4 Comparison

We can compare the implementations of the EVENODD and the Reed-Solomon mechanisms for *n-k*=2. Both implementations reach a constant performance with constant overhead at small values of *k* and for small data sizes. We calculated the energy efficiency *e* using the power consumption of the Pentium Pro 133 processor, which is approximately 14 W [15]. To summarise we have the following results:

mechanism	speed [µs/kB]	E	e [µJ^{-1}]	Minimal k	Minimal data size
EVENODD encoding	88	11.6	0.83	5	symbol: 32 bytes
EVENODD decoding	96	10.6	0.76	5	symbol: 32 bytes
Reed-Solomon encoding	365	2.8	0.20	5	packet: 1 ATM cells
Reed-Solomon decoding	257	3.9	0.28	5	packet: 16 ATM cells

Table 1: Characteristics of error correcting codes for *n-k*=2.

The efficiency of the implementation of the mechanisms will in general be a bit better when larger k and/or symbol sizes are used. The used implementation of the Reed-Solomon encoding is about four times as inefficient as EVENODD. Decoding is more than two times as inefficient. The minimal size of a data item (either a symbol or a packet) depends on the choice of the packet size for EVENODD. When the packet size is chosen to be one column (just like our Reed-Solomon implementation), then the minimum size of a packet for EVENODD equals $32(k\text{-}1)$. The minimum packet size for EVENODD is thus smaller than for Reed-Solomon for approximately $k<26$. E.g. when $k=7$, then the minimal packet size for EVENODD equals just more than 4 ATM cells, which is much less than the minimum of 16 cells for Reed-Solomon coding. Note that a non-parameterised implementation of the Reed-Solomon code can be more efficient and has less initial overhead [9].

A.2.5 A minimal communication system

Error correction mechanisms for wireless communication involve computational overhead and communication overhead at both the transmitter and the receiver side. This is overhead in time, but also overhead in energy consumption. In our context we mainly focus on the *energy overhead*. The overhead is composed out of two elements, the encoding overhead and the communication overhead.

In the previous sections we have investigated the *computational* energy efficiency of two error correction mechanisms. We will now consider the *energy efficiency of a system* in which also the energy consumption of the communication interface is incorporated. We will only consider the data that is actually transmitted, and not incorporate additional costs involved with the wireless interface like turning ‘on’ and ‘off’ the transceiver, sending extra control data, etc. These matters are dealt with in e.g. the medium access control layer. A more precise analysis would require these costs to be incorporated as well. However, these costs are dependent on the underlying protocols and operating system, and the energy savings capabilities of the system. So, to have a clean comparison we will only use in our analysis the energy needed for the actual data transfer.

The *communication overhead* mainly depends on the number of additional bits that are transmitted. The number of redundant bytes equals the number of redundant packets *m* multiplied by the packet size *S,* and thus the total communication overhead of *k* source packets is O ($m\ S$). So the communication energy efficiency of transmitting an *(n,k)* redundant code equals:

$$e_{com} = E_{com} \,.\, k / m \qquad (6)$$

in which E_{com} is the energy-efficiency factor that is determined by the energy consumption of the wireless interface.

As an example we will determine the energy efficiency of a small system consisting of a WaveLAN PCMCIA card as wireless communication device and a Pentium Pro 133 MHz as general-purpose processor (the same processor as used in our experiments). We will compare the efficiency using a rating ρ that indicates the amount of energy consumed to process one byte, using:

$$\rho = \text{time to process 1 byte [s] . required power [W]} \qquad (7)$$

The WaveLAN 2.4 GHz modem interface consumes approx. 1800 mW when transmitting [19]. The data transfer-rate is 2Mb/s. One byte takes thus 4 µs to process, which results in an energy consumption of ρ=7.2 µJ/byte. The Pentium Pro 133 processor takes 14 W [15]. As an example we will now compare this with the energy consumption to encode data with EVENODD. The time needed to encode 1 kB of data using the EVENODD mechanism is 88 µs, so one byte takes 88/1024 µs, resulting in an energy consumption of ρ=1.2 µJ/byte. The energy-efficiency ratio between encoding with EVENODD and communication thus equals 7.2 / 1.2 = 6.0.

This shows that when the power consumption of the wireless interface is relatively high, then it is worthwhile to use adaptive error control that tries to minimise the amount of data transmitted over the wireless channel. However, if the energy consumption of the wireless interface is relatively low compared to the power consumption required to implement the error control, then it might be more effective to utilise an error coding that is optimised for worst case conditions and does not need the control-loop.

A.3 Conclusion

When error-correction mechanisms are used for wireless systems, a major design criterion should be the energy efficiency of a mechanism. *Adaptable error correction*, that adapts its parameters and scheme according to the error-rate and required QoS, can be used to trade-off between performance and cost, including the required energy consumption.

We have shown that the power consumption that is required to perform the error-correction mechanism can be significant. To illustrate this we have studied two implementations of different error correction mechanisms with different characteristics

and capabilities, i.e. EVENODD and Reed-Solomon. We have identified that the choice of whether to apply adaptive error control depends on the energy efficiency of the error control, and on the energy consumption of the wireless interface. Adaptive error control is effective only when the energy consumption of error control to process a certain amount of data is lower than the energy consumption required to transmit this amount over the wireless channel. When the energy consumption of error control and wireless communication are almost equal, then the extra complexity and the required feedback from the receiver to the transmitter must be carefully incorporated in the decision to use adaptive error control.

The implementations of these mechanisms on a general-purpose processor show that they already reach constant performance and constant energy efficiency for small values of k and for small data sizes. The Reed-Solomon code is attractive because it is the most general technique capable of tolerating n-k simultaneous failures. The code rate can be defined fine-grained. However, this flexibility makes the encoding about four times less energy efficient than EVENODD (due to the introduced complexity and the requirement of computations in the finite field). The EVENODD mechanism on the other hand is rather course-grained. It can sustain two packet erasures, or, more generally, it allows the reconstruction of up to 2(k-1) erased *symbols* (and not only packet erasures as with Reed-Solomon without interleaving). This ability gives EVENODD an inherent adaptability, since it can efficiently tolerate variable burst error rates using the same code rate. This increases its flexibility and gives a further reduction in energy consumption for decoding when the burst-error size is smaller than a whole packet.

References

[1] Ahlquist G.C., Rice M., Nelson B.: “Error control coding in software radios: an FPGA approach”, *IEEE Personal Communications,* August 1999, pp. 35-39, 1999.

[2] Blaum M., et al.: “EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures”, *IEEE Transactions on computers*, Vol. 44, No 2, pp. 192-201, February 1995.

[3] Bowers H., Zhang H.: “Comparison of Reed-Solomon codec implementations”, *Technical rep. UC Berkeley*, http://infopad.eecs.berkeley.edu/~hui/cs252/rs.html.

[4] Cho, Y.J., Un, C.K.: “Performance analysis of ARQ error controls under Markovian block error pattern”, *IEEE Transactions on Communications*., Vol. COM-42, pp. 2051-2061, Feb-Apr. 1994.

[5] Elaoud, M, Ramanathan, P.: “Adaptive Use of Error-Correcting Codes for Real-time Communication in Wireless Networks”, *proceedings IEEE Infocom’98*, pp. 548-555, March 1998.

[6] Goslin G.R.: “Implement DSP functions in FPGAs to reduce cost and boost performance”, EDN magazine, 1996, http://www.ednmag.com/reg/1996/101096/21df_05.htm.

[7] Havinga P.J.M.: “Energy efficiency of error correction on wireless systems”, *proceedings IEEE Wireless Communications and Networking Conference (WCNC’99)*, September 1999

[8] Havinga, P.J.M.: "Energy efficiency of error correcting mechanisms for wireless communication", *CTIT Technical Reports 1998*, TR-CTIT 98-19, September 1998, the Netherlands.

[9] Krol Th., personal communication.

[10] Lettieri P., Schurgers C., Srivastava M.B.: "Adaptive link layer strategies for energy efficient wireless networking", ACM WINET, 1999.

[11] Lin, S., Costello, D.J., Miller, M.: "Automatic-repeat-request error-control schemes", *IEEE Comm. Magazine*, v.22, n.12, pp. 5-17, Dec 1984.

[12] Liu, H., El Zarki, M.: "Delay bounded type-II hybrid ARQ for video transmission over wireless networks", *proceedings Conference on Information Sciences and Systems*, Princeton, March 1996.

[13] "L64711/12/13/14 Reed-Solomon Encoders/decoders", *LSI logic*, http://www.lsilogic.com/products/unit5_7d.html.

[14] MacWilliams, F.J., Sloane, N.J.A.: "The theory of error-correcting codes", *North-Holland Publicing Company*, Amsterdam, 1977.

[15] "Pentium Pro processors, Product overview", http://developer.intel.com/design/pro.

[16] Plank, J.S.: "A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems", *Software, practice & experience*, 27(9), Sept 1997, pp. 995-1012.

[17] Rizzo, L.: "Effective Erasure Codes for Reliable Computer Communication Protocols", *ACM Computer Communication Review*, Vol. 27- 2, pp. 24-36, April 97.

[18] Rizzo, L., sources for an erasure code based on Reed-Solomon coding with Vandermonde matrices. Available at http://www.iet.unipi.it/~luigi/vdm.tgz.

[19] "WaveLAN/PCMCIA network adapter card", http://www.wavelan.com/support/libpdf/fs-pcm.pdf.

Biography

Paul Havinga was born in Groningen, the Netherlands, on January 1st 1962. He obtained his Atheneum-diploma from the Florens Radewijnsz College in Raalte in 1980. He graduated in 1985 at the Enschede University of Professional Education in Computer Science. Subsequently, he started to work as project-assistant at the faculty of Computer Science at the University of Twente in 1985.

He worked there at various projects: on a multiprocessor system *Tumult*, on a central ATM network switch for multimedia applications *Rattlesnake*, and during the last four years in the *MOBY DICK* project on mobile computing. Main emphasis in the latter project has been on the design of an energy-efficient architecture for handheld multimedia systems. Although the subject of these projects seems at first appear to be quite different, there are many similarities. Many techniques and mechanisms that are used in networks for multiprocessor systems, are also suitable for switching elements for distributed systems, and even for communication inside a small handheld computer.

Currently he is involved with the *Chameleon* project that deals with reconfigurable computing for handheld multimedia systems. This project is a spin-off from the *MOBY DICK* project and the research presented in this thesis.

Publications

[1] Havinga P.J.M., Smit G.J.M.: "Design techniques for low power systems", *Journal of Systems Architecture*, Vol. 46, Issue 1, 2000.

[2] Havinga P.J.M., Smit G.J.M., Bos M.: "Energy efficient wireless ATM design", to appear in *ACM/Baltzer Journal on Mobile Networks and Applications (MONET), Special issue on Wireless Mobile ATM technologies,* Vol. 5, No 2., 2000.

[3] Havinga P.J.M., Smit G.J.M.: "Octopus – an energy-efficient architecture for wireless multimedia systems", *ProRISC workshop on Circuits, Systems and Signal Processing, ProRISC'99*, pp. 185-192, November 1999.

[4] G.J.M. Smit, M. Bos, P. J.M. Havinga, J. Smit: "Reconfigurable Mobile Multimedia Systems", *ProRISC workshop on Circuits, Systems and Signal Processing, ProRISC'99*, pp. 431-436, November 1999.

[5] M. Bos, P.J.M. Havinga, G.J.M. Smit: "Single shared memory space architecture for less power", *ProRISC workshop on Circuits, Systems and Signal Processing, ProRISC'99*, pp. 43-48, November 1999.

[6] Gerard J.M. Smit, Martinus Bos, Paul J.M. Havinga, Sape J. Mullender, Jaap Smit: "Chameleon - reconfigurability in hand-held multimedia computers", *proceedings First International Symposium on Handheld and Ubiquitous Computing*, HUC'99, September 1999.

[7] Havinga P.J.M.: "Energy efficiency of error correction on wireless systems", *proceedings IEEE Wireless Communications and Networking Conference (WCNC'99)*, September 1999.

[8] Havinga P.J.M., Smit G.J.M.: "Octopus: embracing the energy efficiency of handheld multimedia computers" , *proceedings fifth annual ACM/IEEE international conference on mobile computing and networking (Mobicom'99),* pp.77-87, August 1999.

[9] Havinga P.J.M., Smit G.J.M., Bos M.: "Energy efficient wireless ATM design", *proceedings second IEEE international workshop on wireless mobile ATM implementations (wmATM'99)*, pp. 11-22, June 1999.

[10] Smit G.J.M., Havinga P.J.M., van Opzeeland M., Poortinga R.: "Implementation of a wireless ATM transceiver using reconfigurable logic", *proceedings wmATM'99*, June 2-4, 1999.

[11] P.J.M. Havinga, G.J.M. Smit: "E^{2}MaC: an energy efficient MAC protocol for multimedia traffic", *Moby Dick technical report*, 1998, http://www.cs.utwente.nl/~havinga/papers/e2mac.ps.

[12] Havinga P.J.M., Smit G.J.M.: "The Pocket Companion's architecture", *Euromicro summer school on mobile computing '98*, ISBN 951-38-4576-1, Oulu, pp. 25-34, August 1998.

[13] G.J.M. Smit, P.J.M. Havinga, S. Mullender, A. Helme, G. Hartvigsen, T.Fallmyr, T. Stabell-Kulo, A.Bartoli, L.Rizzo, M. Avvenuti: "An overview of the Moby Dick project", *1st Euromicro summer school on mobile computing*, ISBN 951-38-4576-1, pp. 159-168, Oulu, August 1998.

[14] P.J.M. Havinga, "Energy efficiency of error correcting mechanisms for wireless communication", *CTIT Technical Reports 1998*, TR-CTIT 98-19, September 1998.

[15] S. Mullender, G.J.M. Smit, P.J.M. Havinga, A. Helme, G. Hartvigsen, T.Fallmyr, T. Stabell-Kulo, A.Bartoli, L.Rizzo, M. Avvenuti: "The MobyDick architecture", *CTIT technical report*, TR-CTIT 98-18, September 1998.

[16] Smit J., Stekelenburg M., Klaassen C.E., Mullender S., Smit G., Havinga P.J.M.: "Low cost & fast turnaround: reconfigurable graph-based execution units", *proceedings 7th BELSIGN workshop*, Enschede, the Netherlands, May 7-8, 1998.

[17] Havinga, P.J.M., Smit, G.J.M.: "Minimizing energy consumption for wireless computers in Moby Dick", *proceedings IEEE International Conference on Personal Wireless Communication ICPWC'97*, ISBN 0-7803-4298-4, pp. 306-311, December 1997.

[18] George R.J. Linnenbank, Paul J.M. Havinga: "An Event-Driven Wireless MAC Protocol Simulator", *Proceedings IEEE International Conference on Personal Wireless Communications* (ICPWC'97), pp. 110-114, December 1997.

[19] G.J.M. Smit, P.J.M. Havinga: "A survey of energy saving techniques for mobile computers", *Moby Dick technical report*, 1997, http://www.cs.utwente.nl/~havinga/papers/energy.ps.

[20] P.J.M. Havinga, G.J.M. Smit: "The system architecture of the Pocket Companion", *Moby Dick technical report*, 1997.

[21] Havinga P.J.M., Smit G.J.M.: "Low power system design techniques for mobile computers", *CTIT technical report series 97-32*, ISSN 1381-3625, Enschede, the Netherlands, 1997.

[22] G.R.J. Linnenbank, P. Venkataram, P.J.M. Havinga, S.J. Mullender, G.J.M. Smit "A request-TDMA multiple-access scheme for wireless multimedia networks", *Mobile Multimedia Communications*, (D. Goodman, D. Raychaudhuri, Eds.), pp. 173-180, 1997.

[23] Havinga P.J.M., Smit G.J.M.: "Minimizing energy consumption for handheld computers in Moby Dick", *Proceedings of the 23rd Euromicro Conference 97*, pp. 196-201, September 1997, published in *Journal of Systems Architecture*, ISSN 1383-7621/0165-6074, September 1997.

[24] 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*, ISBN 0-8186-8129-2, pp. 231-238, September 1997.

[25] G.R.J. Linnenbank, P. Venkataram, P.J.M. Havinga, S.J. Mullender, G.J.M. Smit: "A request-TDMA multiple-access scheme for wireless multimedia networks", *Proceedings MoMuC-3*, 1996.

[26] P.J.M. Havinga, G.J.M. Smit, A. Helme: "Survey of electronic payment methods and systems", *Proceedings Euromedia 96*, pp. 180-187, London, December 1996.

[27] P.J.M. Havinga, G.J.M. Smit, A. Helme: “Survey of electronic payment methods and systems”, *Memoranda Informatica* 96-15, University of Twente, 1996.

[28] G.J.M. Smit, P.J.M. Havinga: “Rattlesnake – a multimedia ATM switching system”, *Memoranda Informatica* 96-16, University of Twente, 1996.

[29] G.J.M. Smit, P.J.M. Havinga, F.M. Dillema, P.G.A. Sijben: “Audio source location for a digital TV-Director”, *Proceedings Euromedia 96*, pp. 103-111, London, December 1996.

[30] Havinga P.J.M., Smit G.J.M.: “Rattlesnake – a single chip high-performance ATM switch”, *proceedings International conference on multimedia networking (MmNet’95*), ISBN 0--81--867090—8, pp. 208-217, Aizu, Japan, September 26-29, 1995.

[31] G.R.J. Linnenbank, P.J.M. Havinga, S.J. Mullender, G.J.M. Smit: “Request-TDMA: A Multiple-Access Protocol for Wireless Multimedia Networks”, *Proceedings IEEE 3rd Symposium on Communications and Vehicular Technology in the Benelux*, ISBN 9--06--144992--8, pp. 20--27, 1995.

[32] P.J.M. Havinga, W.H. Tibboel, G.J.M. Smit: “Virtual lines; A deadlock free and real-time routing mechanism for ATM networks”, *Information sciences*, ISSN 0020--0255, 851--3, 1995.

[33] G.J.M. Smit, P.J.M. Havinga: “Multicast and Broadcast in the Rattlesnake ATM Switch”, *Proceedings Intl. Conference on Multimedia and Networking*, ISBN 0--81--867090--8, pp 218--226, 1995.

[34] G.J.M. Smit, P.J.M. Havinga: “A Switch Architecture for Real-Time Multimedia Communications”, *Euromicro Workshop on Parallel and Distributed Processing*, ISBN 0--81--865370--1, pp. 438-444, Malaga, Spain, 1994.

[35] G.J.M. Smit, P.J.M. Havinga: “A Switch Architecture for Real-Time Multimedia Communications”, *Pegasus Paper 94-1,* January 1994.

[36] G.J.M. Smit, P.J.M. Havinga, W.H. Tibboel: “Virtual lines: a routing mechanism for switch networks”, *8th. Intl. Symposium on Computer and Information Sciences*, Antalya, Turkey, 1993.

[37] G.J.M. Smit, P.J.M. Havinga, W.H. Tibboel,: “Virtual lines; A deadlock free and real-time routing mechanism for ATM networks”, *Pegasus Paper 93-5,* November 1993.

[38] G.J.M. Smit, P.J.M. Havinga: “Virtual lines: a dead-lock free and real-time routing mechanism for ATM Networks”, *4th Intl. Workshop on Network and Operating Systems Support for Digital Audio and Video*, Lancaster, United Kingdom, pp. 83-86, 1993.

[39] G.J.M. Smit, P.J.M. Havinga: “Performance Analysis of Routing Algorithms for the Rattlesnake Network”, *ACM/IEEE Intl. workshop on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS’93)*, San Diego, ISBN 1--56--555018--8, January 1993.

[40] G.J.M. Smit, P.J.M. Havinga, M.J.P. Smit: “Rattlesnake: a Network for Real-time Multimedia Communication”, *Proceedings. IEEE Multimedia ’92*, Monterey, California, pp. 89-100, April 1992.

[41] P.J.M. Havinga, G.J.M. Smit,: “The architecture of Rattlesnake: a Real-time Multimedia Network”, *Proc.3rd IEEE Workshop on Network and Operating System Support for Digital Audio and Video*, San Diego, CA, 1992.

[42] G.J.M. Smit, P.J.M. Havinga, P.G. Jansen: “On the design of a dynamic reconfigurable network switch”, *Microprocessing and microprogramming*, 4 pp, 1992.

[43] G.J.M. Smit, P.J.M. Havinga, M.J.P. Smit: “Rattlesnake: a network for real-time Multimedia Communication”, *Computer communication review*, 3-22, ISSN 0146--4833, 1992.

[44] G.J.M. Smit, P.J.M. Havinga, P.G. Jansen: “A Programmable communication architecture based on Kautz graphs”, *Proc.12th IFIP World Computer Congress*, pp. 578-584, September 1992.

[45] G.J.M. Smit, P.J.M. Havinga, P.G. Jansen: “An algorithm for generating node disjoint routes in Kautz digraphs”, *5th IEEE parallel processing symposium*, ISBN 0--81--869167--0, pp. 102-107, Anaheim, May 1991.

[46] G.J.M. Smit, P.J.M. Havinga, P.G. Jansen: “Generating node disjoint routes in Kautz digraphs”, *CSN'91* Congres SION, pp. 514-527, November 1991.

[47] G.J.M. Smit, P.J.M. Havinga, P.G. Jansen: “A programmable network switch for Kautz networks”, *Proceedings parallel computing 91*, London, September 1991.

[48] G.J.M. Smit, P.J.M. Havinga, P.G. Jansen, F.de Boer, E. Molenkamp: “On hardware for generating routes in Kautz graphs”, *Proceedings Euromicro '91*, pp. 593-600, Vienna, August 1991.

[49] P.J.M. Havinga, G.J.M. Smit, Jansen P.G.: “The interprocessor communication architecture of Tumult-64”, *Proceedings of the fifth International symposium on computer and information Sciences*, pp.461-470, November 1990.

[50] G.J.M. Smit, P.J.M. Havinga, P.G. Jansen: “Communication issues for parallel systems”, *Memoranda Informatica* 90-54, University of Twente, 1990.

[51] G.J.M. Smit, P.J.M. Havinga, P.G. Jansen: “An algorithm for generating node disjoint routes in Kautz digraphs”, *Memoranda Informatica* 90-43, University of Twente, 1990.

[52] P.J.M. Havinga, A.M. Vink: “RSIM: a simulation program for the Tumult-64 network”, *Memoranda Informatica* 89-37, University of Twente, 1989.

[53] P.J.M. Havinga, G.J.M. Smit: “Performance of the Tumult-64 multiprocessor network”, *Memoranda Informatica* 89-38, University of Twente, 1989.

[54] P.J.M. Havinga, G.J.M. Smit, P.G. Jansen: “The interprocessor communication architecture of Tumult-64”, *Memoranda Informatica* 89-39, University of Twente, 1989.

[55] P.J.M. Havinga, G.J.M. Smit, H.C. van der Bij: “Hardware support for a real-time deadline scheduler”, *Proceedings VMEbus in industry*, pp. 333-340, November 1989.

[56] G.J.M. Smit, P.J.M. Havinga: “TUMULT-64, a VMEbus compatible multi-processor system”, *VMEbus in research*,.pp.413-419, October 1988.

[57] G.J.M. Smit, P.J.M. Havinga: “TUMULT a VMEbus compatible multi-processor system”, *VMEbus applications seminar*, pp. 27-39, November 1987.

[58] P.J.M. Havinga, G.J.M. Smit, P.G. Jansen: “Metastability and its consequences for the Tumult-interface”, *Memoranda Informatica* 87-11, University of Twente, 1987.

[59] G.J.M. Smit, P.J.M. Havinga: “Implementation of TUMULT-15 network interface for the VME-bus”, *Memoranda Informatica* 86-11, University of Twente, 1986.

[60] G.J.M. Smit, P.J.M. Havinga: “TUMULT-VME interface board, hardware description”, *Memoranda Informatica* 86-14, University of Twente, 1986.

[61] P.J.M. Havinga: “Implementation of Tumult network interface with an EPLD 5C121”, *Memoranda Informatica* 86-17, University of Twente, 1986.