Energy-efficient adaptive wireless network design

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

University of Twente, department of Computer Science, the Netherlands

{havinga, smit, m.bos}@cs.utwente.nl

Abstract

Energy efficiency is an important issue for mobile computers since they must rely on their batteries. 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. Due to the dynamic nature of wireless networks, adaptations are necessary to achieve energy efficiency and an acceptable quality of service.

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.

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.

Keywords: energy efficiency, adaptability, Quality of Service, application interface

1 Introduction

Wireless communications have experienced a rapid growth over the last decade. Portable computers like PDAs and laptops that use wireless communication to interact with the environment rely on their limited battery energy for their operation. Energy consumption is the limiting factor in the amount of functionality that can be placed in these devices. More extensive and continuous use of network services by multimedia applications will only aggravate this problem. 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 such that it minimises 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.

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 [4][18].

Considerations of energy efficiency are fundamentally influenced by the trade-off between energy consumption and achievable Quality of Service (QoS). Adaptations to the dynamic nature of wireless networks are necessary to achieve energy efficiency and acceptable quality of service.

Overview of the paper

With the increasing integration levels, energy consumption has become one of the critical design parameters. While low-power components and subsystems are essential building blocks for portable systems, we have shown in our research [10][11][12] that a system-wide architecture that incorporates the low-power vision into all layers of the system is beneficial because there is a vital relationship between hardware architecture, operating system and applications, where each benefits from the others. Therefore, energy reduction techniques have to be applied in all design levels of the system. 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 paper we address the issue of *energy efficiency* in protocols for wireless networks and the host network interface, and in doing so incorporate all layers of the mobile system. A critical design issue is that the operating system is removed from the critical communication path, and that data traffic is reduced, mainly because unnecessary data copies are removed.

Section 2 introduces the opportunities of energy conservation in the various layers of the wireless network protocol stack. Section 3 gives an overview of the various energy reduction techniques for

wireless communication that have been applied in the MOBY DICK project. In section 4 some conclusions are drawn.

2 Energy-efficient wireless communication

In this section we will discuss some techniques that can be used to reduce the energy consumption that is needed for the wireless communication. This can be accomplished efficiently and effectively only by a careful design of all network layers in the network protocol stack. *Adaptability* of the protocols is a key issue.

- *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 can adapt its modulation techniques and basic error-correction schemes. The bandwidth a radio offers influences its energy consumption. The energy per bit transmitted or received tends to be lower at higher bit rates. For example, the WaveLAN [23] radio operates at 2Mb/s and consumes 1.8 W, or 0.9 µJ/bit. A small FM transceiver (Radiometrix BIM-433) operates at 40 kb/s and consumes 60 mW, or 1.5 µJ/bit. This example shows that the low bit-rate radio is 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 will consume about 30 times more energy than the low bit-rate radio.
- *Medium access layer* – In an energy-efficient MAC protocol [5] the basic objective is to minimise the time the radio needs to be powered. Since the overhead introduced due to state transitions is also significant, (e.g. for WaveLAN, the transmit/receive transition takes about the same time as needed for transport of 58 bytes), minimising the number of transitions also reduces energy consumption. Therefore the data cells from one mobile should be grouped together as much as possible within the QoS restraints of the connection. By scheduling data transfers in bulk, an inactive terminal is allowed to doze and power off the receiver. Consequence is that the network interface must be reactivated at a scheduled time. Unsuccessful actions of the transceiver due to collisions and errors should be avoided, and the protocol should adapt to the dynamic environment.

- *Logical link control layer* – Due to the dynamic nature of wireless networks, *adaptive error control* can give significant gains in effective bandwidth and energy efficiency. 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 slot scheduler in the base-station can also adapt its traffic scheduling to the error conditions of wireless connections of 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 stale packets. Depending on the service class and QoS of a connection, a different flow control can be applied such 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. For such traffic the buffer is probably relatively small, and when the connection is hindered somewhere, stale data will be discarded and fresh data will be used. Flow control would needlessly spend energy for transmitting 'old' images and flow-control messages.
- *Network/transport 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, characteristic for 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 unnecessary increases in energy consumption and deterioration of QoS. 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 [1].
- *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 [8]. 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 [14]. 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 goes 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). Aside from causing performance problems, this also incurs significant energy consumption. To address the performance problem, several user-level communication architectures have been developed that remove the operating system from the critical communication path [2]. The same principles may also be used to increase energy efficiency.

3 Energy-efficient wireless networking in MOBY DICK

The MOBY DICK project develops and defines the architecture of a new generation of handheld computers, the so-called *Mobile Digital Companion*. This section describes the basic principles and mechanisms of the network interface architecture being implemented in the MOBY DICK project, our energy-efficient medium access control for wireless links, called E^2MaC, and how adaptability is included into all layers of the system. The implementation is described briefly.

3.1 MOBY DICK

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 believe *reconfiguration* is of fundamental importance to the success of low-power handheld systems. 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. Current research on several aspects of wireless networks (like error control, frame-length, access scheduling) indicate that adapting to the current condition of the wireless link continuously has a big impact on the energy-efficiency of the system (e.g. [6][9][15][19][20][22][24]). In the MOBY DICK project [21] 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. In MOBY DICK we have a reconfigurable systems-architecture which, in combination with a QoS driven operating system and network protocols, can cope with the inherent dynamics of a mobile environment. The protocol and the architecture are targeted to a system in which quality of service and energy consumption plays a crucial role.

3.2 System overview

The wireless network is composed of several base-stations where each handles a single radio cell covering several mobile stations. We consider an office environment in which the cells are small and have the size of one or several rooms. Small cells save energy as the transmitters can be low powered and they also provide a high aggregate bandwidth since small cells need to be shared with only few mobiles. The backbone of the base-stations is a wired network.

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 QoS manager that is responsible for the traffic in his communication cell is mainly responsible for providing the QoS to all connections in its cell. This manager tries to find a (near) optimal 'schedule' that satisfies the wishes of all applications.

In the protocol the base-station controls access by dividing bandwidth into transmission slots. 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, and synchronise the mobile and the base-station which allows the mobile to power-on precisely when needed [16].

The QoS provisions of ATM with its small fixed sized cells (48 bytes data, 5 bytes control) 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), or even multiple priority levels. However, when adopting ATM in a wireless environment, we need a much more dynamic

approach to resource usage. The small cells of ATM 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 and error control mechanism that can be adapted to the QoS of a connection. 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 layers of the communication protocol are summarised in Figure 1. 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.

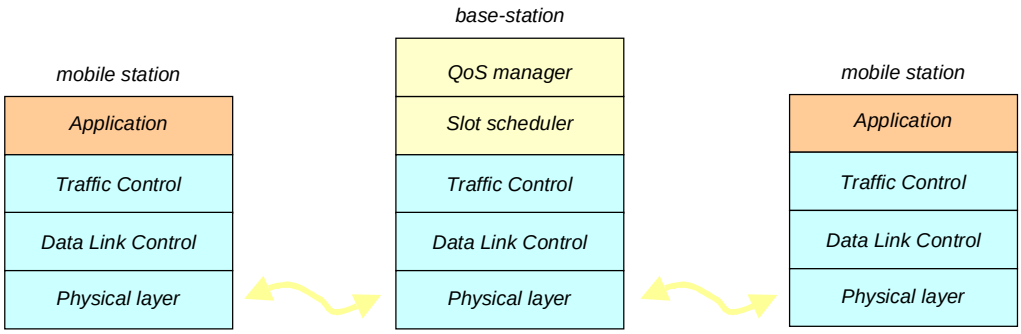

Figure 1: Protocol stack

The lower layers exist in both the mobile and the base station. The *Data link control* manages the data-transfer with the physical layer, 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.

3.3 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.

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 applied 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.

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 allows 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 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. In [18] a bandwidth reallocation algorithm is described that fits well to the QoS model used by the QoS manager.

3.4 Slot scheduler

The *slot scheduler* assigns bandwidth and determines the required error coding for each individual connection. The QoS manager provides the service contracts used. A schedule is broadcast to all mobiles so that they know when they should transmit or receive data. 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. 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 uplink and downlink packets for a mobile are also grouped sequentially (if possible) so that the mobile can power down longer and make minimal transitions between power modes.

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 according to the error rate observed at the receiver. The slot scheduler retrieves this status information via a backward connection and indicates to the network interface which error coding scheme to use. 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 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 [6].

3.5 Energy-efficient and adaptive network interface

The general theme that influences many aspects of the design of the network interface is adaptability and flexibility. This implies that for each connection a different set of parameters concerning scheduling, flow control and error control can be applied.

Figure 2 depicts the basic blocks of the architecture of the Network Interface. The number of connection queues is dynamic and the figure is just an example.

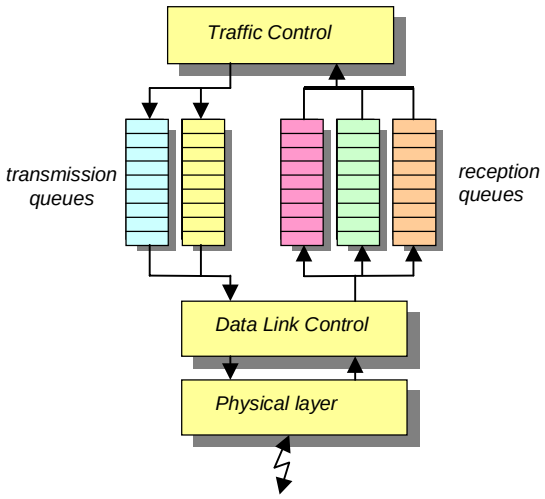

Figure 2: The network interface architecture.

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 manager, 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*. This buffer is only meant to store ATM cells for a short time, just enough to implement an effective error-control mechanism. The buffer is organised in such a way that it has a small queue for each connection. 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 and basic error correction on each ATM cell.

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. 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.

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 [18]. 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 bandwidth accordingly.

Each connection has its own connection queues with *customised flow control*. Flow control is needed to prevent buffer overflow. Depending on the type of data carried by a connection, a replacement policy and size can be set for each connection. Flow control information is transmitted in the header of each data packet.

3.6 Energy-efficient MAC protocol (E^2MaC)

The E^2MaC protocol uses fixed-length frames. The frame is divided in time-slots that can have three basic types: *traffic control*, *registration request*, and *data*. 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 data slots are collision-less.

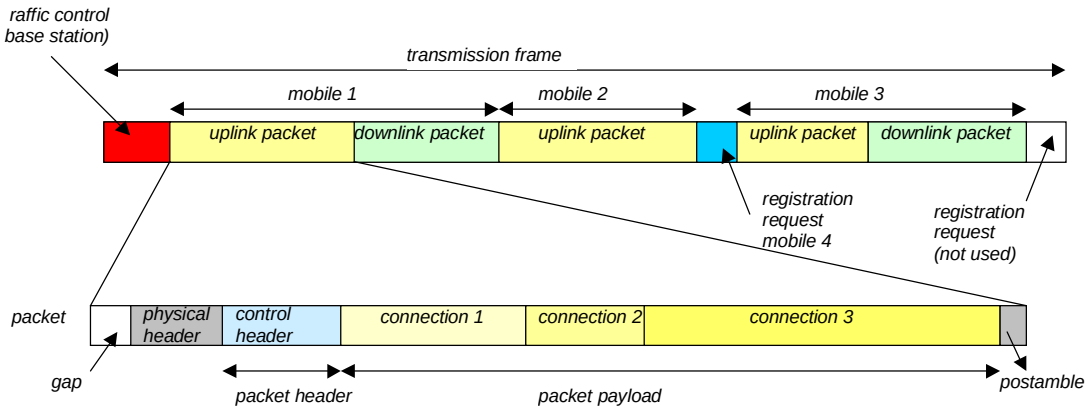

Figure 3: Example of a transmission frame.

To minimise the overhead due to state transitions, the traffic from one mobile is grouped together as much as possible within the QoS restraints. These cells form a packet that is a sequence of ATM cells for one mobile, possibly for multiple connections. Each packet is constituted of a header, followed by the payload consisting of ATM cells. In general the transition-overhead between

transmit and receive modes is much less than the transition overhead between power down modes, thus transmission packets and reception packets for one mobile are placed right after each other in the frame. The header of a packet contains 1) information about the actual length of the data for each connection, 2) the error coding applied, 3) status information, like buffer status and effectiveness of error control on previous data, and 4) a connection control message.

The adaptive error control has a tight feedback loop to allow the transmitter to adapt the coding rate instantaneously (i.e. in general effective on the next frame) to the error rate observed at the receiver. The Traffic Control module in the network interface that performs the error detection and correction keeps track of the rate of corrected and un-correctable number of cells. This status information, together with the status information of the radio, is fed back to the slot scheduler (either in the header of an uplink packet if the status is from a mobile, or via an internal connection if the status is from the base-station). The slot scheduler retrieves this information and can adapt the required error coding and bandwidth. Note that with this technique the energy efficiency is increased, but it cannot guarantee a reliable connection. Higher level protocols or mechanisms in the application are needed to ascertain this if reliability is required.

3.7 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 [17]) 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, it becomes possible to scale both ways. As bandwidth can be used only once, it is better to have one authority that divides it instead of having the possibility that two applications 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.

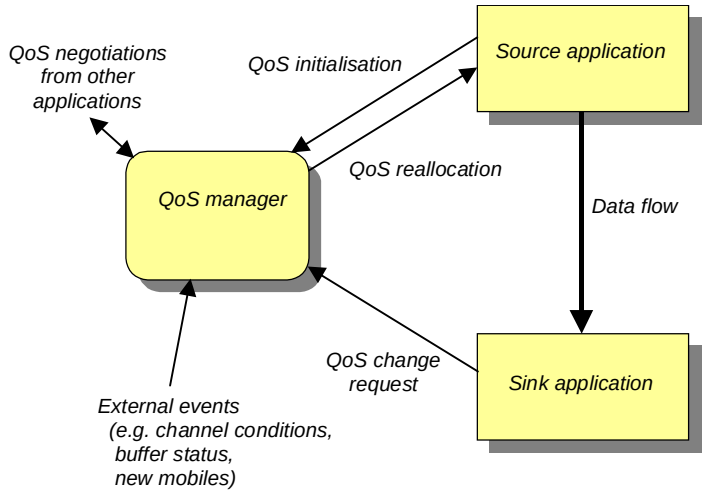

Figure 4: The service model for adaptive applications.

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. In the MOBY DICK Project *Inferno* from Lucent Technologies Bell Labs [7] is used. Communicating programs are multithreaded by nature in Inferno. 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, 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 message from channel
                "parse message";
                usecolor = 0 or 1 depending on contents of message;
        }
}
```

3.8 Implementation

The medium access protocol and data link layers have been analysed. These analysis have shown that the protocol can 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 [10].

Currently we are implementing a test-bed for the network interface that we can use to experiment with the various techniques and mechanisms for e.g. error control and the MAC protocol. Both base-station 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 uses a lightweight protocol with the mobile to transfer the packets. 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 mobile from ‘standard’ protocols.

The testbed is build with off-the-shelf components to allow a short design cycle. Figure 5 shows the architecture of the network interface test-bed.

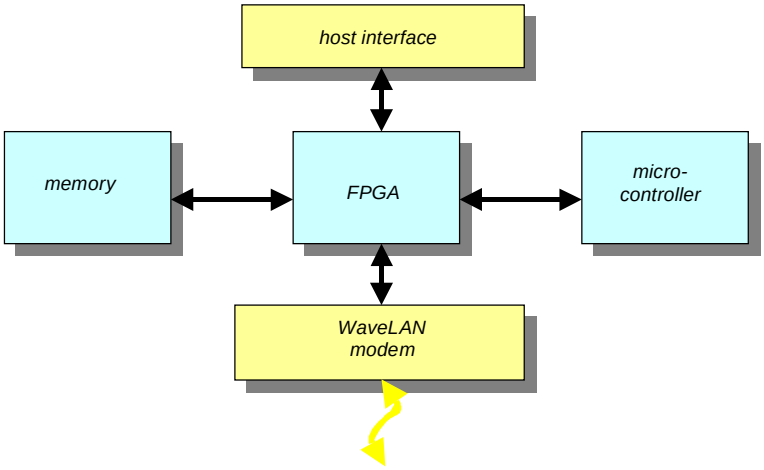

Figure 5: Network interface test-bed architecture.

The four basic components are:

- *memory* (512 kBytes SRAM) that will be used to implement the connection queues,
- 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 3.3). It controls the functions to be performed by the FPGA, controls the radio modem, and performs 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.
- Currently we use a WaveLAN modem as the physical layer. The raw data rate is 2 Mb/s. The modem provides the basic functionality to send and receive frames of data. It does not include a Medium Access Protocol, but provides signalling information like carrier sense and collision detect.

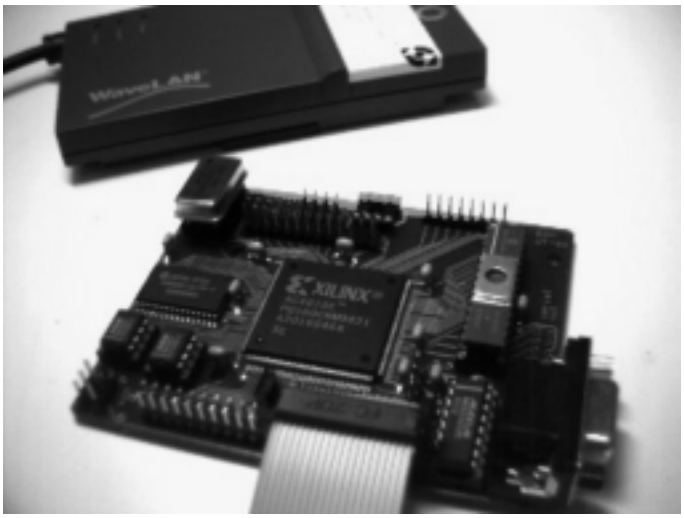

Figure 6: Network interface implementation.

Figure 6 shows a photograph of the network interface implementation. The FPGA controls the data-flow between the radio and the host. 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 the host. It therefore performs the queue setup and administration. This administration is used to setup VCI mapping tables and queue address maps stored in the FPGA. It thereby administrates 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. 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 when to initiate error correction on the received packet. The error correction is being performed in a close interaction 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 corrected packet will be forwarded to its destination. Cells arriving from the host can be protected against errors that might occur on the wireless channel by adding redundant cells. Just

like the error correction mechanism, the FPGA provides the compute intensive functions that the microcontroller can use to build the packet that is protected by the required error correcting code.

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.

3.9 Future design issues

Multiple radios – The mobiles are expected to spend most of their time in sleep modes if we are to attain acceptable battery lifetime. This also means that they can neither transmit nor receive radio transmissions most of the time. As discussed above, 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 detect whether the base-station has some messages waiting.

Another means of detection 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 data. Therefore we are also considering using a very low power receiver for the signalling only. This receiver will 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.

System decomposition – In a flexible system such as the Companion there are many trade-offs. Decomposition of system functions - such as the network protocol stack - and a careful analysis the data flow in the system can reduce the energy consumption considerably.

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 our system 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 contrary 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 functions from one module to another, possibly even to another machine. A complicating factor is that a wireless environment is very dynamic, it is not feasible to search for *the* optimal solution.

Hybrid MAC protocol – The E^2MaC 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. Thus, a hybrid MAC protocol that can operate in two modes and that is optimised for both network types will probably be the most efficient.

4 Conclusions

We have presented an architecture for 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 approach 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.

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, 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] Chockalingam, A., Zorzi, M.: "Energy consumption performance of a class of access protocols for mobile data networks", *VTC'98*, Ottawa, Canada, May 1998.

[7] Dorward S., Pike R., Presotto D., Ritchie D., Trickey H., Winterbottom P.: "Inferno", *Proceedings COMPCON Spring'97*, 42nd IEEE International Computer Conference, 1997.

[8] 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.

[9] 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.

[10] 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.

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

[12] Havinga P.J.M., Smit G.J.M.: “Octopus: embracing the energy efficiency of handheld multimedia computers” , *proceedings ACM/IEEE Mobicom* 1999

[13] Havinga P.J.M.: “Energy efficiency of error correction on wireless systems”, *proceedings IEEE Wireless Communications and Networking Conference (WCNC’99*), 1999.

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

[15] 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.

[16] Mangione-Smith, B.: “Low power communications protocols: paging and beyond”, Low power symposium 1995, http://www.icsl.ucla.edu/~billms/Publications/pagingprotocols.pdf.

[17] RealPlayer, http://www.realplayer.com

[18] 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.

[19] Schuler C.: "Optimization and adaptation of error control algorithms for wireless ATM”, *International Journal of Wireless Information Networks*, Vol. 5, No. 2, April 1998.

[20] Sivalingam, K.M., Srivastava, M.B. Agrawal, P.: “Low power link and access protocols for wireless multimedia networks”, *Proc. IEEE Vehicular Technology Conference*, Phoenix, AZ, pp. 1331-1335, May 1997.

[21] 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.

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

[23] “WaveLAN/PCMCIA network adapter card”, http://www.wavelan.com/support/libpdf/fs-pcm.pdf.

[24] 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.

APPENDIX A. Evaluation of the E^{2}MaC protocol

In this section we provide the results of some evaluation analysis of the E^{2}MaC protocol. More extensive results can be found in [10].

A.1 The access protocol

The chance of a *collision* in the E^{2}MaC 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. Figure 7 shows the energy efficiency characteristics of Slotted Aloha for a various number of mobiles, and for E^{2}MaC for two frame-sizes (i.e. 10 and 50 ATM cells).

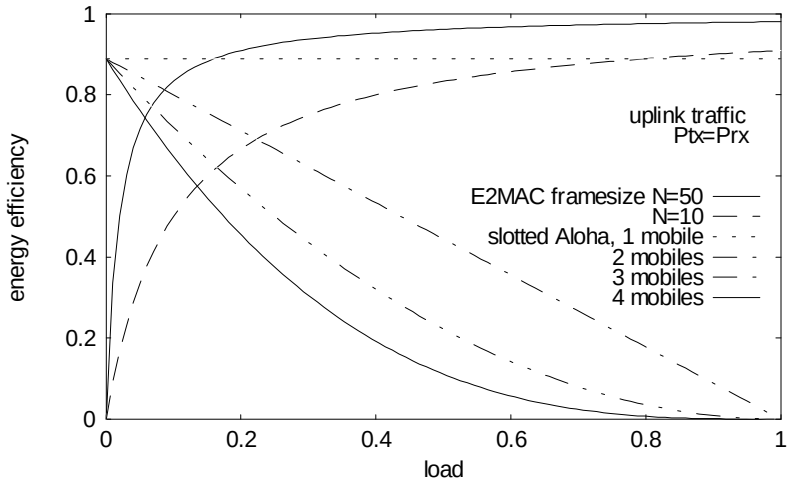

Figure 7: Energy efficiency vs. load for uplink traffic on Slotted Aloha and E^{2}MaC.

First of all we must note that the energy efficiency of the E^{2}MaC protocol is independent of the activity of the users, in contrast with 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), then the energy consumed by the E^{2}MaC protocol is more than with Slotted Aloha that does not have this overhead. 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.

A.2 Adaptive packet size

Figure 8 shows the effect on the energy efficiency with respect to the packet size and the overhead introduced with one transition.

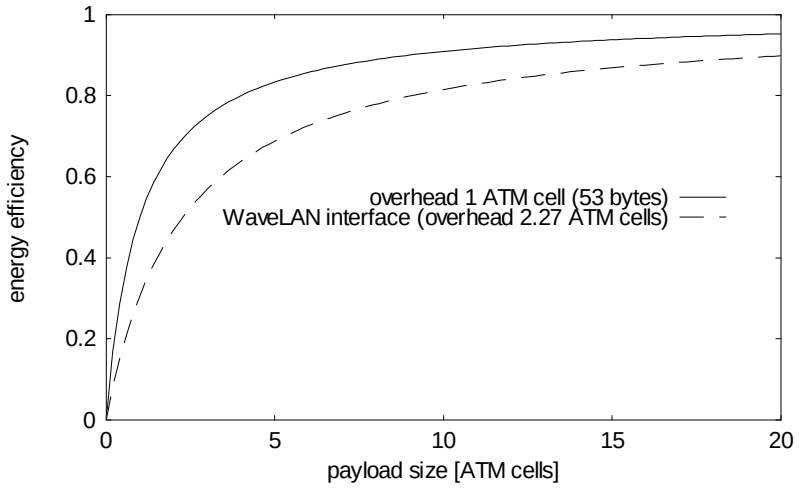

Figure 8: Energy efficiency vs. payload size for various overheads.

The figure shows that the packet size has a big influence on the energy efficiency, which makes a large packet size profitable. However, this is valid for ideal situations in which no errors occur only. 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 space between two transmissions), transmission mode transitions) and overhead in bits transmitted over the air (preamble, MAC control header, postamble). We have plotted the goodput *g* versus packet size *s* for various bit error rates in Figure 9. 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.

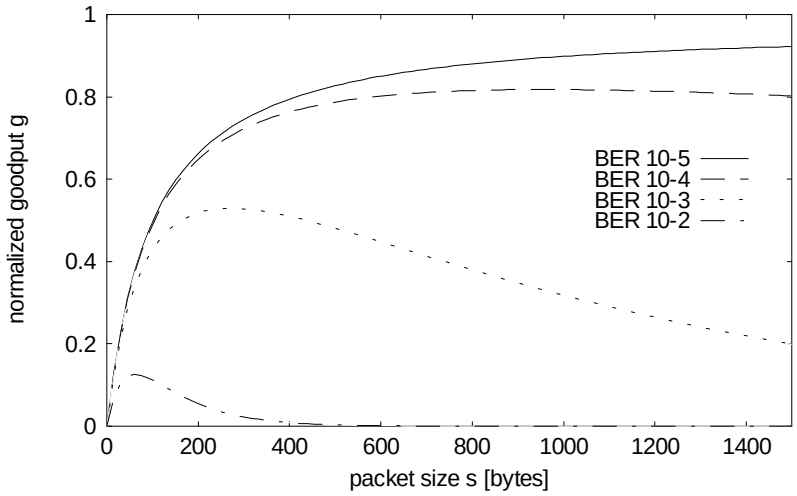

Figure 9: Goodput vs. packet size on WaveLAN for various BER.

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 error control mechanism can be adapted, such that it divides the packet into smaller segments that each has their own error control.

APPENDIX B. Error-correction energy efficiency of a minimal communication system

Since high error rates are inevitable to the wireless environment, *energy-efficient error control* is an important issue for mobile computing systems. We have studied the energy efficiency of two different error-correction mechanisms (i.e. Reed-Solomon coding and EVENODD coding [3]) and have measured the efficiency of an implementation in software. A model is presented in [13] that can be used to determine an energy-efficient error-correction scheme of a minimal system consisting of a general-purpose processor and a wireless interface. This model can be used to *adapt* the error correction parameters, such that the system is energy efficient. We define the *energy efficiency e* as the amount of data processed divided by the energy that is consumed to process that data.

Error-correction mechanisms for wireless communication involve encoding overhead and communication overhead. The *communication overhead* mainly depends on the number of

additional bits that are transmitted. The energy efficiency of transmitting an *(n,k)* redundant code equals:

$$e_{com} = E_{com} \,.\, k \,/\, (n - k) \qquad (1)$$

in which E_{com} is determined by the energy consumption of the wireless interface.

The energy efficiency e_{ec} of Reed-Solomon coding and of EVENODD coding (approximation when n-k>2) for k large is approximately:

$$e_{ec} = E_{ec} \,/\, (n-k) \,.\, k \qquad (2)$$

The factor E_{ec} is determined by the implementation of the coding. We measured E_{ec} on a PentiumPro 133 MHz for EVENODD and Reed-Solomon using n-k=2. In Figure 10 both functions are plotted (with (n-k) constant), and the trade-off is shown clearly. The energy efficiency of error correction increases when the number of source packets k decreases, and the energy efficiency of communication e_{com} shows a greater efficiency when the ratio between the number of source packets k and the number of redundant packets (n-k) is large. The value of k where the communication overhead equals the error correction overhead depends on the implementation of the coding and on the energy efficiency of the communication.

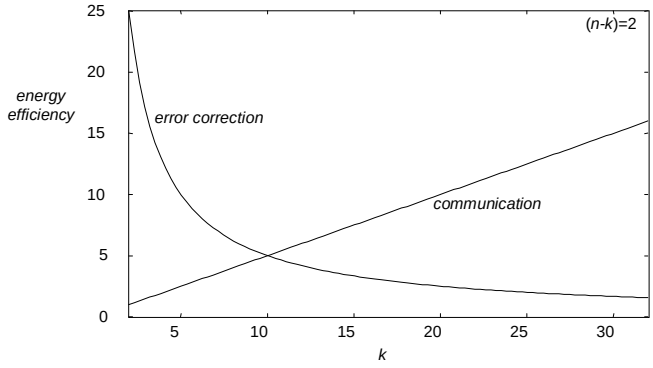

Figure 10: System energy efficiency *e* vs. number of source packets *k*.

When we combine both equations we can determine the energy efficiency of a system composed of a transceiver and a codec as:

$$e_{tot} = \{ (n - k) / E_{com} \,.\, k \;\; + \;\; (n - k) \,.\, k / E_{ec} \}^{-1} \qquad (3)$$

So, once we know the values of E_{com} and E_{ec}, this equation can be used to adapt the parameters of the error correcting code, such that an energy efficient implementation is achieved. As an example we determine the energy efficiency of a system consisting of a WaveLAN PCMCIA card with a PentiumPro and a StrongARM SA-1100. The energy efficiency of the StrongARM is about 70 times better than the PentiumPro. We will compare the energy efficiency using a rating that indicates the amount of energy consumed to process one byte, using:

$$E = 1 / (\text{time to process 1 byte [} \mu s] \text{ . required power [mW] }) \qquad (4)$$

Figure 11 shows the result.

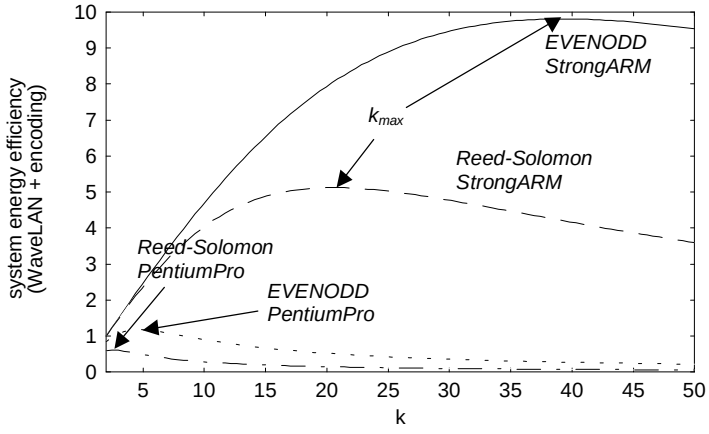

Figure 11: Energy efficiency of error correction on systems with WaveLAN.

The general tendency is that the energy efficiency increases with increasing k (because less communication is needed), up to a certain k_{max} where the encoding cost is becoming the limiting factor. If the efficiency of encoding is high (like the StrongARM implementation of EVENODD), then k_{max} is high also. If the efficiency of encoding is low (like the Reed-Solomon encoding on a PentiumPro), then the communication cost is negligible.

To sustain the bad state probability of the system in a certain environment, k must be chosen small enough. This k is called k_{min}. For n-k=2 the tolerable error rate ε is determined by $\varepsilon<2/(k+2)$, thus $k<2/\varepsilon - 2$. On the other hand, it is of no use to have a larger k than k_{max}, not only because it is less efficient, but also because it has lower error correcting capabilities. Both k_{max} (for energy efficiency) and k_{min} (for error correcting capabilities) thus determine the appropriate choice for k in a system.