On Quality-of-Service and Energy Consumption Tradeoffs in FEC-Encoded Audio Streaming

Z. Zhou, P. K. McKinley and S. M. Sadjadi
Software Engineering and Network Systems Laboratory
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824
{zhouzhin,mckinley,sadjadis}@cse.msu.edu

Abstract

This paper addresses the energy consumption of forward error correction (FEC) protocols as used to improve quality-of-service (QoS) for wireless computing devices. The paper also characterizes the effect on energy consumption and QoS of the power saving mode in 802.11 wireless local area networks (WLANs). Experiments are described in which FEC-encoded audio streams are multicast to mobile computers across a WLAN. Results of these experiments quantify the tradeoffs between improved QoS, due to FEC, and additional energy consumption caused by receiving and decoding redundant packets. Two different approaches to FEC are compared relative to these metrics. The results of this study enable the development of adaptive software mechanisms that attempt to manage these tradeoffs in the presence of highly dynamic wireless environments.

I. Introduction

Over the past few years, the number of mobile computing devices has grown dramatically. This increase is driven primarily by the rapid growth in the use of the Internet and the wide deployment of wireless networks. Unfortunately, advances in rechargeable battery technologies have not kept pace with the development of other hardware components. In mobile computing, energy is a limited resource. Unlike other system resources, such as memory, energy that has already been consumed cannot be "released" and "reallocated." This property motivates both the need to increase energy efficiency (more work per unit of consumed energy) and the need to extend battery lifetime (work longer under a given load).

While wireless communication brings mobility to the user, the network subsystem is also one of the largest consumers of energy in a mobile device. This problem is exacerbated in noisy environments, where error control strategies generate additional network traffic. Traditional error control methods are based on retransmissions of lost packets, while others involve forward error correction (FEC) [1]. FEC introduces redundancy in the data stream in the form of parity packets, enabling recovery of lost packets at the receiver without retransmissions. FEC is particularly well-suited for use with interactive, real-time communication streams, where waiting for retransmissions introduces unacceptable delay and jitter. However, transmitting and receiving parity packets consumes additional energy.

In this paper, we investigate the relationship between quality of service (QoS) and energy consumption characteristics when FEC is used in communication with wireless devices. The work is experimental and focuses on FEC support for interactive audio multicasting to handheld computers and laptops in wireless local area networks (WLANs). In this study, we focus on WLANs that extend wired LANs, that is, they are used in infrastructure mode. One dimension of our ongoing work addresses energy management and QoS in mobile ad hoc networks. Two FEC protocols are investigated, one using block erasure codes and the other using the GSM 06.10 encoding algorithm for cellular telephones.

The main contributions of this study are threefold. First, the study helps to quantify the tradeoff between improved packet delivery rate, due to FEC, and additional energy consumption caused by receiving and decoding redundant packets. Second, we assess the effectiveness of periodically putting the wireless network interface card (WNIC) into sleep mode to save energy while satisfying QoS requirements. Third, we demon-

0-7803-8277-3/04/$20.00 ©2004 IEEE.

strate how these results can be used as a basis for the development of adaptive software mechanisms that "manage" the energy consumption in the presence of highly dynamic environments.

The remainder of this paper is organized as follows. In Sections II and III, respectively, we describe the experimental environment and software configuration used in this study. Section IV describes experiments to evaluate energy consumption characteristics under different FEC configurations. In Section V, we assess the quality of audio communication using various FEC protocols and parameters. Section VI shows how an adaptive software framework can respond dynamically to changes in the environment. Related work is discussed in Section VII, and conclusions are given in Section VIII.

II. Experimental Environment

This study was conducted on a mobile computing testbed that includes various types of devices: laptop computers, iPAQ handheld systems, and Xybernaut Mobile Assistant V wearable computers. These systems communicate via an 11Mbps 802.11b WLAN. The local wireless cell is also connected to a multi-cell WLAN that covers many areas of the Michigan State University Engineering Building and its courtyard. To monitor the wireless traffic and help interpret experimental performance results, we execute the WildPackets Airopeek network analyzer on a laptop in the wireless cell.

The interconnection of the systems is depicted in Figure 1. A live audio stream is multicast from a wired desktop computer to multiple mobile devices via the WLAN. Effectively, the receivers are used as multicast-capable Internet "phones" participating a conferencing application. Most experiments in this paper used iPAQs as receivers. Each iPAQ is a model H3650 or H3870, with a 206 MHz StrongARM processor and 64 MB memory. Each is configured with the Familiar Linux distribution and Blackdown Java, and each system has a dual-slot expansion pack to support a PCMCIA wireless card (Cisco Aironet 350 Series) and a IBM 1.0 GB Microdrive. In some experiments we used laptop computers, each with a 2.0 GHz P4 processor and 1.0 GB memory, running RedHat 9.0 Linux.

A key aspect of the experimental environment involves measurement of energy consumption. Such mechanisms are specific to the particular battery configuration on a given system. For example, the iPAQ main unit and the expansion pack have separate batteries that operate independently, unless the voltage value of the main unit battery becomes lower than that of the external battery. In this situation, the main unit battery will draw power from external battery through an activated internal trickle charge until the voltage value exceeds that of the external battery. However, the external battery will never draw power from the main unit [2].

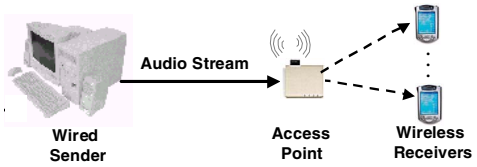


Fig. 1. Physical experimental configuration.

We measure energy consumption using both a hardware method, which is more accurate, as well as a software method, which is the only option in a deployed mobile system that needs to adapt its behavior based on the current state. For the former case, we remove the system batteries and use a power supply (Elenco Model XP-760) to power the system. We use an Agilent 3458A multimeter to measure the current drawn from the power supply. Because the iPAQ main unit and the expansion pack can share power, this configuration supplies power to both the iPAQ main unit and the expansion pack. For software measurements in Linux, we record the drop in battery voltage or capacity provided by the APM (Advanced Power Management) through the /proc file system. Specifically, a program reads from /proc/apm five times per minute, and uses the mean of these samples to represent the voltage or capacity drop in one minute. As noted, this measurement includes only the main unit battery, which can draw power from the expansion pack battery. As we shall see later, however, the expansion pack battery drain much faster than the main battery under communication-intensive scenarios.

III. Software Architecture

This study is part of an ONR-sponsored project called *RAPIDware*, which addresses design of adaptive middleware to support interactive applications in dynamic, heterogeneous environments.

MetaSockets. Our experiments make use of *MetaSockets* [3], which are adaptable communication components that we developed earlier. MetaSockets (short for *metamorphic sockets*) can be used in place of regular Java sockets, providing the same imperative functionality, including methods for sending and receiving data. However, their internal structure and behavior can be adapted at run time in response to changes in their environment.

MetaSockets are implemented in Adaptive Java [4],

an extension to Java that supports run-time modifications to components using computational reflection. Although using a Java-based language (Adaptive Java is source-to-source compiled into Java) introduces some processing overhead, its support for dynamic loading of code is very useful to our investigation of adaptive software. Moreover, even our modest 206 MHz iPAQs can support real-time audio streaming in Java. Figure 2 illustrates the internal architecture of the particular type of MetaSocket used in this study. Packets are passed through a pipeline of Filter components, each of which processes the packets. Example filter services include: auditing traffic and usage patterns, transcoding data streams into lower-bandwidth versions, encrypting and decrypting data, and implementing forward error correction (FEC) to make data streams more resilient to packet loss. Figure 2 shows that the MetaSocket also supports special types of methods to insert and remove filters, as well as retrieve their status. Details of MetaSocket architecture and operation can be found in [3].

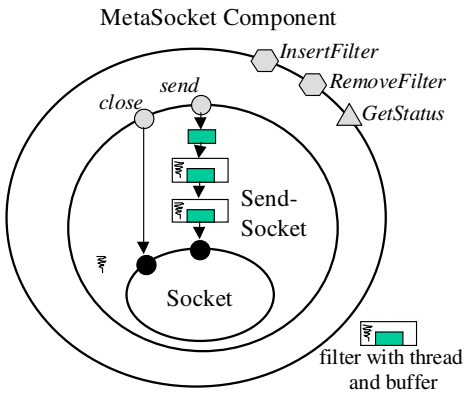


Fig. 2. Structure of a MetaSocket.

Block-Oriented FEC Encoder/Decoder. In this study, we first evaluate the energy consumption characteristics of a particular FEC method based on (n, k) *block erasure codes*, which were popularized by Rizzo [1] and are now used in many wired and wireless distributed systems. Figure 3 depicts the basic operation of these codes. An encoder converts k source packets into n encoded packets, such that any k of the n encoded packets can be used to reconstruct the k source packets [1]. In this paper, we use only *systematic* codes, which means that the first k of the n encoded packets are identical to the k source packets. We refer to the first k packets as *data* packets, and the remaining $n-k$ packets as *parity* packets. Each set of n encoded packets is referred to as a *group*.

The advantage of using block erasure codes for multicasting is that a single parity packet can be used to correct independent single-packet losses among different receivers [1]. We implemented MetaSocket filters for block-oriented FEC encoding and decoding using an open-source Java implementation of Rizzo's C library. In the remainder of the paper, we will refer to the block-oriented FEC simply as "FEC (n, k)."

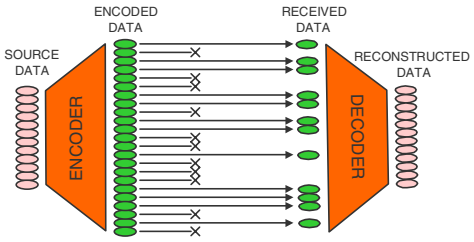
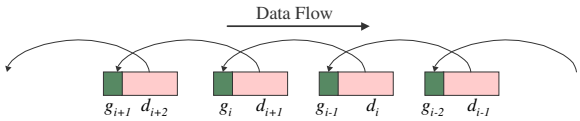


Fig. 3. Operation of block erasure code.

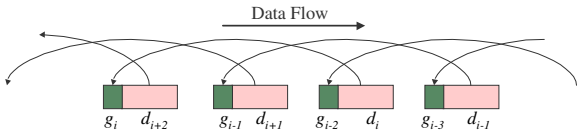
While block-oriented FEC approaches are effective in improving the quality of interactive audio streams on wireless networks [5], the group sizes must be relatively small in order to reduce playback delays. In our studies, we typically use (n, k) values of (6,4) or (8,4). Hence, the overhead in terms of parity packets is relatively high.

GSM-Oriented FEC Encoder/Decoder. An alternative approach with lower delay and lower overhead is *signal processing based FEC (SFEC)* [6], [7], in which a lossy, compressed encoding of each packet i is piggybacked onto one or more subsequent packets. If packet i is lost, but one of the encodings of packet i arrives at the receiver, then at least a lower quality version of the packet can be played to the listener. The parameter θ is the offset between the original packet and its compressed version. Figure 4 shows two different examples, one with $\theta = 1$ and the other with $\theta = 2$. As mentioned, it is also possible to place multiple encodings of the same packet in the subsequent stream, for example, using both $\theta_1 = 1$ and $\theta_2 = 3$.

We use GSM 06.10 encoding for generating the redundant copies of packets. Although GSM is a CPU-intensive coding algorithm [7], the bandwidth overhead is very small. Specifically, the GSM encoding creates only 33 bytes for a PCM-encoded packet containing up to 320 bytes (160 samples in our experiments). We use the Tritonus Java version of the GSM codec, a freeware package available under GNU public license. Unfortunately, this Java version is unable to satisfy real-time audio encoding and decoding requirements on iPAQs with low processing power, so all the GSM-related experiments were conducted on laptop computers. In the remainder of the paper, we will refer to the GSM-oriented FEC simply as "GSM (θ, c)," which means copies of the coded packet p are placed in c successive



(a) GSM encoding with $\theta = 1$



(b) GSM encoding with $\theta = 2$

Fig. 4. Different ways of using GSM encoding on a packet stream.

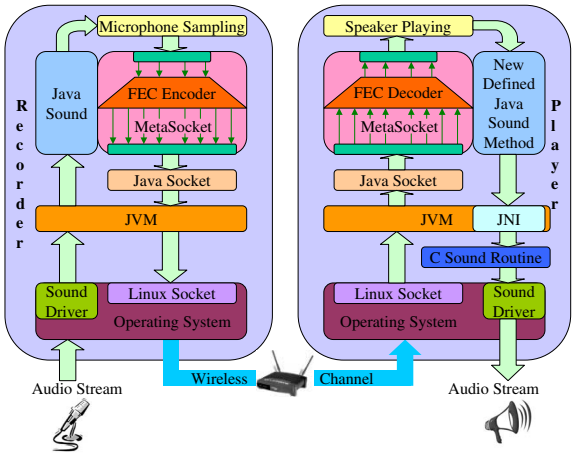


Fig. 5. Software component interaction.

packets, beginning θ packets after p.

Audio Streaming Application. To investigate adaptation in interactive audio communication, we developed an audio streaming application (ASA), depicted in Figure 5. ASA uses MetaSockets instead of regular Java sockets, enabling dynamic insertion and removal of FEC filter pairs, as well as filters to measure and report packet loss characteristics. As shown, the ASA comprises two main parts. On the sending station, typically a desktop computer, the *Recorder* reads live audio data from a system's microphone. The Recorder multicasts this data to the receivers via a MetaSocket. If the MetaSocket is configured to introduce FEC on the data stream, it invokes an FEC encoder and transmits the modified audio stream on the network. On each receiving node, the stream arrives on a MetaSocket, where it is decoded as necessary, and delivered to the *Player* component. When executing on an iPAQ, the Player delivers the stream to the speaker using the Java Native Interface (JNI), necessary due to a known problem with audio in Blackdown Java.

IV. Experiments and Results

We first conducted a set of baseline experiments designed to evaluate the effect of FEC/GSM parameter values on energy consumption. For interactive audio streams, the values of k and θ must be relatively small to limit the playback delay to an acceptable level. For example, in many of our experiments we used 8-bit samples and placed 200 samples, or 25 milliseconds of audio, in each packet. If an FEC $(8, 4)$ code is used and the first data packet of a group is lost, then at least 75 milliseconds additional delay will be introduced between the last packet arrival of the preceding group and playing the (decoded) data packet. Therefore, in most experiments we set $k = 2$ or $k = 4$, although in some cases we used $k = 8$ for comparison purposes.

Packet loss characteristics. How to set parameters n and c depends on *packet loss rate and burst error* characteristics. The 802.11b MAC layer provides neither RTS/CTS signaling nor link-level acknowledgements for multicast frames, as it does for unicast frames. Hence, the loss rate for multicast frames can be considerably higher than that for unicast frames [8]. Most error bursts in WLANs are short. Figure 6 illustrates a typical example of this behavior. We plot the overall distribution of packet burst error length that occurred during three traces of audio packets, as recorded by a receiving computer near the room where our wireless access point is located. The average packet loss rate, across the three traces, was 17%. Also plotted in the figure is the distribution produced by a simulation using a two-state Markov model [9], which is widely used to model losses in wireless networks.

Two characteristics of this plot are important to this study. First, while some large bursts occur, the vast majority are under 4 packets long, and most "burst" errors comprise a single packet loss. Such results are encouraging because they imply that a relatively small amount of FEC information is likely to correct most errors, that is, $n - k$ or c can be small. Second, we note that the simulation is reasonably accurate in modeling the loss distribution. Being able to reproduce environmental conditions is notoriously difficult in wireless networks [10], so simulating losses provides a way to test different protocols and parameter values under the same loss conditions. Therefore, many of the experiments described in this section and in Section V use iPAQs and laptops located near the access point, but with emulated packet losses produced by a two-state Markov model and a specified overall loss rate. The results given in

Section VI, however, were collected under real packet loss conditions.

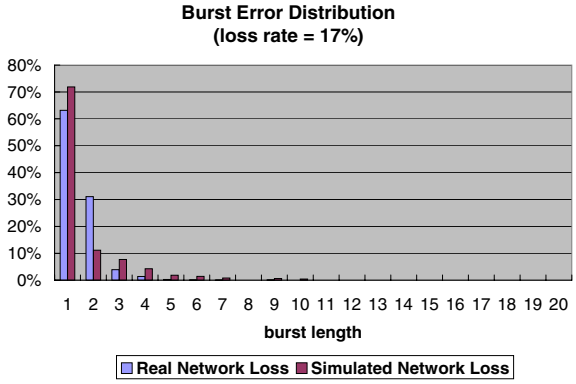
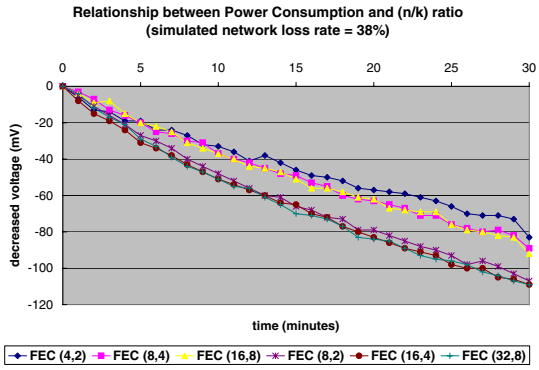


Fig. 6. Burst error distribution (experiments and simulation).

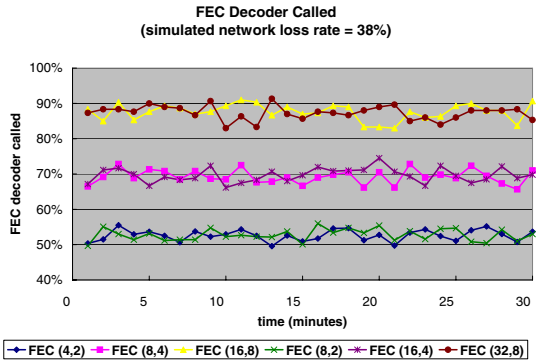
Effect of n, k values. Figure 7(a) shows voltage drop for different (n, k) values under emulated loss conditions, with a mean packet loss rate of 38%. As shown in the figure, the curves are grouped approximately according to the (n/k) ratio: the curves for the $(16, 8)$, $(8, 4)$ and $(4, 2)$ cases, where $n/k = 2$, are relatively close together, as are the curves for the $(32, 8)$, $(16, 4)$ and $(8, 2)$ cases, where $n/k = 4$. This result indicates that the total *number* of incoming (data and parity) packets dominates the energy consumption, at least on the main unit. Other factors, such as how often the FEC decoder is invoked, appear to be less important.

This conclusion is supported by Figure 7(b), which plots the percentage of time that the FEC decoder is invoked for different (n, k) pairs, under the same packet loss conditions. The probability of invoking the decoder depends primarily on the value of k, rather than the (n/k) ratio, and we see that the curves are grouped in that manner. However, despite the fact that FEC decoding is computationally intensive, Figure 7(a) shows that decoding has little effect on the overall behavior of the voltage drop curves, which are linear in the number of incoming packets.

Effect of Power Saving Mode. Adjusting block-oriented FEC parameters, n and k, is not the the only way to manage the energy consumption. The IEEE 802.11 specification also provides a *power saving (PS) mode*, which can be used to switch the WNIC periodically between "sleep" state and "active" state, in order to conserve energy. In the sleep state, power is shut off to most parts of the WNIC, except the timing circuit. For 802.11 WLANs operated in infrastructure mode, the access point (AP) buffers frames destined for hosts in PS mode. All PS hosts are synchronized by the beacon from the AP. Each PS host will wake up to listen to the beacon, which contains a delivery traffic indicator message (DTIM). The DTIM identifies those PS hosts for which buffered frames are waiting to be delivered. Those identified nodes will remain awake until the next beacon. After the AP transmits the DTIM message, it transmits any buffered data. Other researchers [11] have investigated how to exploit the 802.11 PS mode, for example, to support energy efficient routing in mobile ad-hoc networks (MANETs). To our knowledge, however, the interaction between 802.11 PS mode and FEC in streaming audio has not been studied previously.



(a) energy consumption

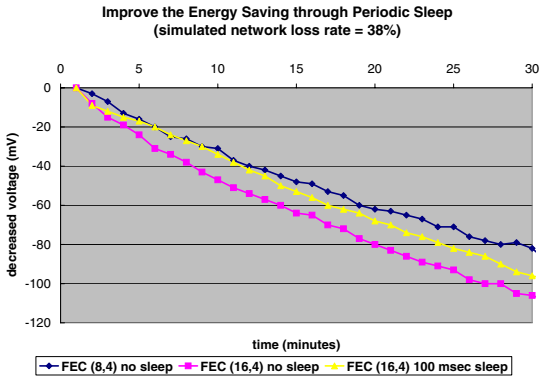


(b) decoder invocations

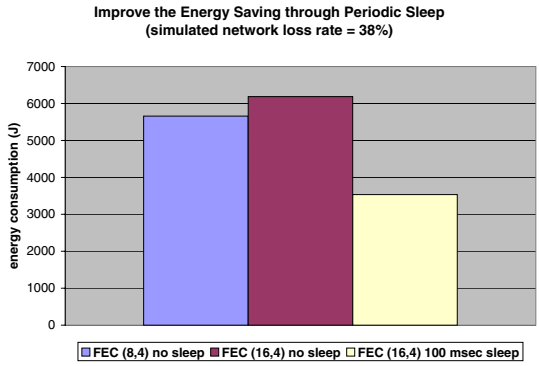
Fig. 7. Baseline experiments.

Figure 8(a) plots the voltage drop over a half-hour experiment, as measured by software, for two different FEC parameters. Use of periodic sleep provides a noticeable, albeit somewhat modest, energy savings on the main unit. However, Figure 8(b), which shows energy consumption as measured by hardware, provides a more complete representation of the situation. Power saving mode combined with a $(16, 4)$ code reduces energy consumption by 42% compared to a $(16, 4)$ code

without PS mode, and by 37% compared to the $(8, 4)$ code. This result indicates that much of the energy being saved is from the battery in the iPAQ expansion pack, rather than from the battery in the main unit. Since the main unit can draw power from the expansion pack, but not vice versa, the expansion pack battery can drain completely before that of the main unit, leaving the iPAQ operational but disconnected from the network. Indeed, the use of PS mode not only reduces energy consumption, but in doing so, also makes practical use of FEC codes with higher n/k ratios. The effect of PS mode on delay is discussed in Section V.



(a) main unit (software measurement)



(b) entire system (hardware measurement)

Fig. 8. Energy savings through periodic sleep.

Effect of GSM coding. From observing the energy consumption characteristics of block-oriented FEC, we conclude that the total number of incoming packets dominates the energy consumption. In contrast, the piggyback method in GSM does not increase the total number of transmitted packets. Therefore, we might expect better energy performance with GSM. This hypothesis is supported by Figure 9, which compares the estimated battery lifetime under various FEC configurations. All the GSM configurations are significantly more energy-efficient than block-oriented FEC.

Of course, reporting energy consumption tells only part of the story. Other factors, such as bandwidth usage, packet delivery rate, and delay, must also be considered in assessing audio streaming communication. Considering the examples illustrated in this section, the use of FEC introduces bandwidth overhead that depends on the values of n, k, θ, and c. Given the same packet size, GSM not only is more energy-efficient, but also consumes much less bandwidth. However, in the next section we will see that these savings have a clear effect on QoS and that packet loss rate is not always the most important factor in determining QoS.

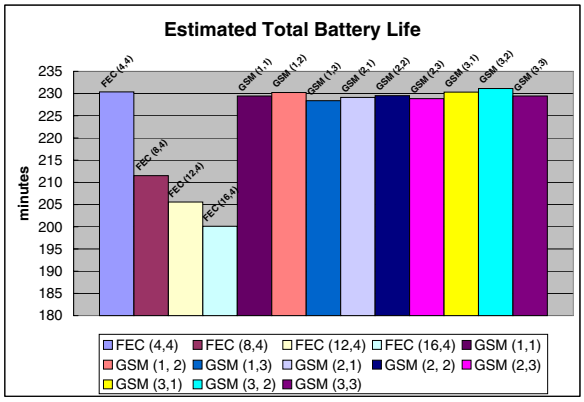


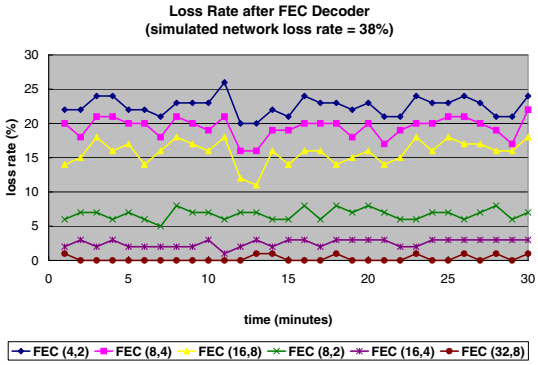
Fig. 9. Energy consumption for FEC and GSM.

V. QoS Assessment

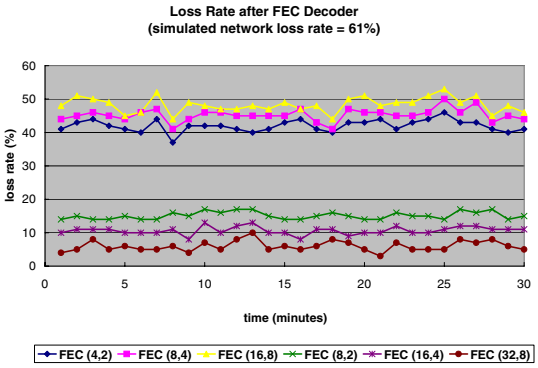
Packet Delivery Rate. Figures 10(a) shows the loss rate as perceived by the receiving application, that is, *after block-oriented FEC decoding*, for different (n, k) settings. The mean network loss rate is 38%. As expected, codes with higher n/k ratios are more effective in correcting losses. Among codes with the same n/k ratio, loss rate decreases as n increases. For example, the $(32, 8)$ code results in a lower packet loss rate than the $(8, 2)$ code, even though the two codes consume approximately the same amount of energy. Both codes do well in correcting single packet losses and short burst errors, but the $(32, 8)$ code can handle any burst error of 24 or fewer packets. However, we need to decrease the packet size to compensate for the jitter introduced by the large value of k.

On the other hand, in some high-loss situations, a smaller value of n can produce better results. For example, let us consider the results in Figure 10(b), where the mean loss rate is very high, 61%. When $n/k = 4$, a larger n value produces a lower loss rate, as in Figure 10(a). However, when $n/k = 2$, a smaller n value

is more effective than a larger one. Effectively, since at least half a group's packets must arrive in order to recover the data, and since errors are bursty, a smaller group size is more likely to achieve this goal. For example, consider four groups using a $(4, 2)$ code, compared to one group of the $(16, 8)$ code. Although the number of packets is the same for each, because the loss rate is 61%, on average they will both lose 10 packets. The $(16, 8)$ code can not recover such a loss, but due to the short burst length, in some cases, the $(4, 2)$ can recover one or more groups, yielding a higher packet reception rate.



(a) simulated packet loss rate = 38%.



(b) simulated packet loss rate = 61%.

Fig. 10. Loss rate after FEC decoding.

Next, let us consider the combination of PS mode and block-oriented FEC. The results presented in Section IV confirmed that a lower n/k ratio consumes less energy than a higher ratio. However, in some situations a low n/k ratio FEC cannot meet QoS expectations due to high loss at the network layer, and a higher n/k ratio FEC is needed. Figure 11 shows that using a $(16, 4)$ code, with and without a periodic sleep of 100 msec, is very effective in reducing losses, compared to an $(8, 4)$ code. Specifically, the loss rate drops from near 20% to only 3%.

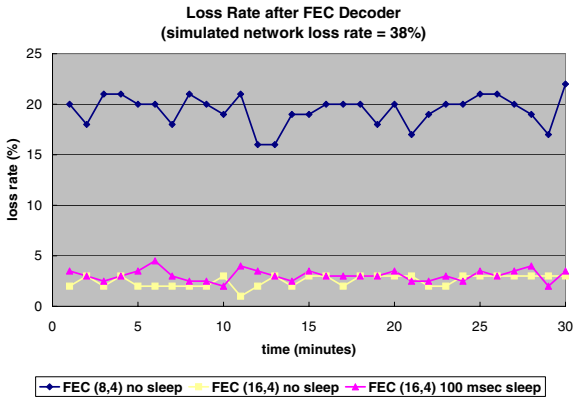


Fig. 11. Effect of sleep mode on loss rate.

We tested the GSM-oriented FEC by setting θ to different values and using 1, 2 and 3 copies of the encoded data. Table I shows that using multiple copies produces a clear advantage in terms of packet delivery rate. However, the loss recovery performance for different GSM parameters highly depends on the actual loss distribution. For example, the loss rates of GSM (1, 1), GSM (2, 1) and GSM (3, 1) are not monotonically decreasing as expected.

Delay. Another factor important to real-time communication is the additional delay introduced into the packet stream. Table II calculates the worst case delay introduced by different FEC codes to wait for the encoded packets. For example, considering FEC (8, 4) and GSM (3, 1), if the first data packet is lost, then the receiver will need to wait for at least 3 packets until the first parity packet or piggybacked packet arrives to recover the loss. In order to satisfy the real-time audio requirement, the delay should not exceed 150 msec [12]. Table II shows that all these codes satisfy this requirement.

Audio Quality. Although packet delivery rate and delay are important objective factors to evaluate the QoS, the most important factor is how the played audio stream sounds to the human ear. Since, the assessment of audio quality by individuals is inherently subjective, we need an objective method. Perceptual Evaluation of Speech Quality (PESQ), defined by ITU-T recommendation P .862, is used to determine voice quality in the telecommunication networks. The PESQ score is mapped to a MOS (Mean Opinion Score) like scale, a single number in the range from -0.5 to 4.5.

Figures 12(a) and 12(b) use PESQ to compared the audio quality of FEC (8, 4) and GSM (3, 3), which achieves the highest packet delivery rate among the block-oriented and GSM-oriented codes respectively.

TABLE I. Loss Rate Comparison of Different FEC Codes

Code	Raw	GSM (1,1)	GSM (1,2)	GSM (1,3)	GSM (2,1)	GSM (2,2)	GSM (2,3)	GSM (3,1)	GSM (3,2)	GSM (3,3)	FEC (4,4)	FEC (6,4)	FEC (8,4)
%	28	11.05	6.28	3.53	13.88	6.8	3.78	10.27	4.8	2.75	28.20	16.29	9.16

TABLE II. Delay Comparison of Different FEC Codes

Code	GSM (1,1)	GSM (1,2)	GSM (1,3)	GSM (2,1)	GSM (2,2)	GSM (2,3)	GSM (3,1)	GSM (3,2)	GSM (3,3)	FEC (4,4)	FEC (6,4)	FEC (8,4)
Delay (msec)	19.95	39.95	59.9	40.83	60.37	80.10	62.55	82.52	102.87	17.79	37.73	52.33

Although GSM (3, 3) has a higher packet delivery rate under different simulated network loss rates, its PESQ score is lower since the recovered packets are generated from the highly compressed, lossy encodings. Considering that PESQ score of 2.0 and above corresponds to acceptable audio quality and Figure 12(c) shows that approximately 20% of the audio data is GSM-quality, we can conclude that GSM-oriented FEC is still suitable for voice communication over wireless networks. However, in situations where a higher quality audio stream is needed, block-oriented FEC may be worth the additional costs in bandwidth and energy.

VI. Toward Dynamic Adaptation

Experimental results such as those presented above can be used to develop rules for dynamic adaptation in mobile computers. Although this aspect of our project is ongoing, we present a sample of the results here. We have conducted a series of experiments in which we used MetaSockets to provide adaptive error control for interactive audio streaming.

In our implementation, two MetaSocket filters, SendNetLossDetector and RecvNetLossDetector, cooperate to monitor the raw loss rate of the wireless channel. Similarly, the SendAppLossDetector and RecvAppLossDetector filters are used to monitor the packet loss rate as observed by the application, which may be lower than the raw packet loss rate due to the use of FEC. At present, a small set of rules is used by a decision maker (DM) component to govern changes in filter configuration. For example, if the loss rate observed by the application rises above a specified threshold, then the DM can decide to insert an FEC filter in the pipeline or modify the (n, k) parameters of an existing FEC filter. On the other hand, if the raw packet loss rate on the channel drops below a lower threshold, then the level of redundancy may be decreased, or the FEC filter may be removed entirely.

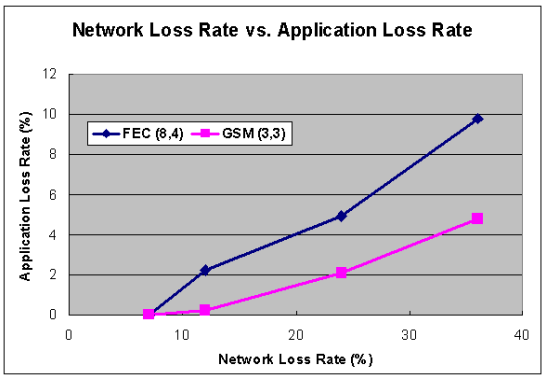
Figure 13(a) shows a trace of an experiment using the ASA described earlier, running in ad hoc mode. A stationary user speaks into a laptop microphone, while another user listens on an iPAQ as he moves among locations in the wireless cell. In this particular test, the iPAQ user remains in a low packet loss area for approximately 30 minutes, moves to a high packet loss area for another 40 minutes, moves back to the low packet loss location for another 30 minutes, then reenters the high packet loss location. He remains there until the iPAQ's external battery drains and the WNIC is disconnected. In this experiment, the upper threshold for the RecvAppLossDetector to generate an *UnAcceptableLossRateEvent* is 20%, and the lower threshold for the RecvNetLossDetector to generate an *AcceptableLossRateEvent* is 5%. As shown in Figure 13(a), the FEC $(4, 2)$ code is effective in reducing the packet loss rate as observed by the application. Figure 13(b) plots the remaining battery capacity as measured during the above experiment and that for a non-adaptive trace. The adaptive version extends the battery lifetime by approximately 27 minutes.

VII. Related Work

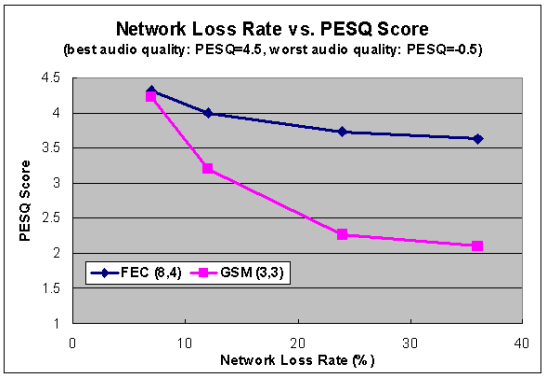
In recent years, numerous research projects have addressed energy consumption issues on mobile devices. A comprehensive treatment is impractical in this paper. Instead, we focus on those contributions that are most related to the work presented here.

Researchers at the Technical University of Berlin [13] investigated the power consumption of an IEEE 802.11 WLAN interface card under different working modes (idle, sleep, receive, transmit) and wireless network conditions. They found that the power consumed by the WNIC is significantly affected by the data rate and packet size. In particular, the energy consumed per bit of data successfully transmitted over the medium decreases as the packet size and data rate increased. The study did not address FEC, however.

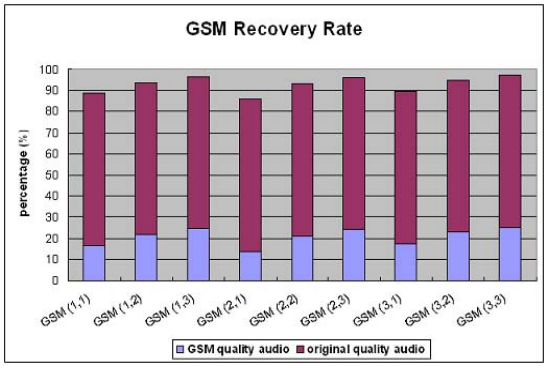
Three main approaches to error control have been used in wireless packet networks [14]: retransmission based (ARQ) [15], pure FEC [16] and hybrid FEC-ARQ [6]. Recent works show that performance gains can be expected from the coupling of the delay-oriented playout adaptation and the error control schemes. Rosenberg et al. [12] investigated the problem of the delay introduced by FEC. They pointed out



(a) loss rate



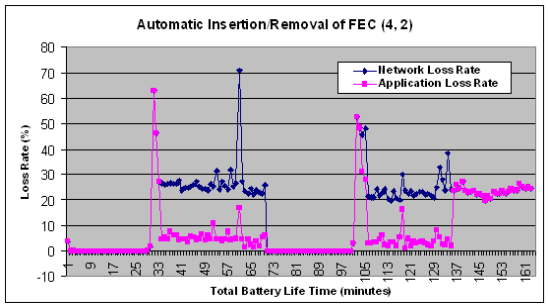
(b) PESQ score



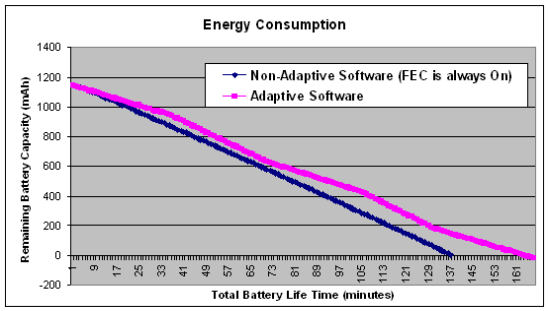
(c) GSM recovery rate

Fig. 12. Audio quality assessment.

that waiting for all the redundant information is inappropriate when network loss rate is low and proposed a number of new playout algorithms to implement playout buffers and absorb delays observed by users. On the other hand, Dempsey et al. [15] proposed S-ARQ and managed to perform timely retransmission of lost packets by controlling the playback time for the first



(a) MetaSocket packet loss behavior with dynamic FEC filter insertion and removal



(b) trace of energy consumption during experiment (software measurement)

Fig. 13. Adaptation between energy and QoS.

packet in each "talkspurt."

Lettieri et al. [17] used theoretical analysis and simulation to compare how different error control strategies (FEC, ARQ, and hybrids) affect energy consumption in wireless networks. The comparison is based on the mean power consumed versus the actual computational load and the delay introduced for different methods. The FEC cost is independent of the channel condition, but in return, FEC can reduce the probability of retransmission. ARQ has good performance when the channel is clear, but as the loss rate increases, ARQ retransmissions adversely affect energy consumption. From the results of their study, the authors argue that the system should be able to select an energy-efficient error control strategy according to QoS requirements, channel quality and packet size. Such studies, as well as our experimental results, support the need to explore adaptive approaches to error control for energy-constrained devices.

Havinga [18] conducted an extensive experimental study of both ARQ and block-based FEC in a WaveLAN network. Havinga found that receiving of parity packets by the WNIC is a major consumer of energy, relative to the encoding/decoding work of the processor. Our study supports this result and additionally consid-

ers the interaction of the 802.11 PS mode with FEC-encoded audio streams.

In addition to manipulating the number of packets transmitted for error control, using compression to reduce the number of bits transmitted also reduces energy consumption. Xu et al. [19] investigated the relationship between data compression and reducing the energy consumption on handheld devices. They proposed an energy model to estimate the energy consumption for compressed downloading of files. Moreover, they developed an adaptive method to fill the CPU idle periods with decompression work. Using this approach, the system can take full advantage of the CPU resource in an energy-efficient way.

VIII. Conclusions

In this paper, we evaluated the energy consumption of forward error correction on wireless devices, where encoded audio streams are multicast to multiple mobile computers. Our results quantify the tradeoff between improved packet delivery rate, due to FEC, and additional energy consumption, delay, bandwidth usage caused by receipt and decoding of redundant packets. We also studied the impact of the 802.11 power saving mode on system energy consumption and compared two different FEC approaches. In future work, we plan to use these studies as a basis for the development of adaptive software mechanisms that attempt to manage these tradeoffs in the presence of highly dynamic environments.

Further Information. A number of related papers and technical reports of the Software Engineering and Network Systems Laboratory can be found at the following URL: `http://www.cse.msu.edu/sens`.

Acknowledgements. The authors would like to thank Zhiyuan Li and Rong Xu at Purdue University for their advice and assistance in establishing the energy management components of our experimental testbed. This work was supported in part by the U.S. Department of the Navy, Office of Naval Research under Grant No. N00014-01-1-0744, and in part by National Science Foundation grants CCR-9912407, EIA-0000433, EIA-0130724, and ITR-0313142.

References

[1] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM Computer Communication Review*, April 1997.

[2] J. M. Vincent, "iPAQ H3100/H3600/H3700 series Pocket PC battery white paper," tech. rep., Compaq Computer Corporation, October 2001.

[3] S. M. Sadjadi, P. K. McKinley, and E. P. Kasten, "Architecture and operation of an adaptable communication substrate," in *Proceedings of the Ninth IEEE International Workshop on Future Trends in Distributed Computing*, (San Juan, Puerto Rico), May 2003.

[4] E. Kasten, P. K. McKinley, S. Sadjadi, and R. Stirewalt, "Separating introspection and intercession in metamorphic distributed systems," in *Proceedings of the IEEE Workshop on Aspect-Oriented Programming for Distributed Computing (with ICDCS'02)*, (Vienna, Austia), July 2002.

[5] P. K. McKinley and S. Gaurav, "Experimental evaluation of forward error correction on multicast audio streams in wireless LANs," in *Proceedings of ACM Multimedia 2000*, (Los Angeles, California), pp. 416–418, November 2000.

[6] M. Podolsky, C. Romer, and S. McCanne, "Simulation of FEC-based error control for packet audio on the Internet," in *Proceedings of IEEE INFOCOM'96*, (San Francisco, California), March 1998.

[7] J.-C. Bolot and A. Vega-Garcia, "Control mechanisms for packet audio in Internet," in *Proceedings of IEEE INFOCOM'96*, (San Francisco, California), pp. 232–239, April 1996.

[8] P. K. McKinley, C. Tang, and A. P. Mani, "A study of adaptive forward error correction for for wireless collaborative computing," *IEEE Transactions on Parallel and Distributed Systems*, September 2002.

[9] E. Elliot, "Estimates of error rates for codes on burst-noise channels," *Bell System Technology Journal*, vol. 42, pp. 1977–1997, September 1963.

[10] D. A. Eckhardt and P. Steenkiste, "A trace-based evaluation of adaptive error correction for a wireless local area network," *Mobile Networks and Applications*, vol. 4, no. 4, pp. 273–287, 1999.

[11] Y.-C. Tseng, C.-S. Hsu, and T.-Y. Hsieh, "Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks," in *Proceedings of the IEEE INFOCOM 2002*, (New York), June 2002.

[12] J. Rosenberg, L. Qiu, and H. Schulzrinne, "Integrating packet FEC into adaptive voice playout buffer algorithms on the Internet," in *Proceedings of IEEE INFOCOM 2000*, pp. 1705–1714, 2000.

[13] B. Burns and J.-P. Ebert, "Power consumption, throughput and packet error measurements of an IEEE 802.11 WLAN interface," tech. rep., Telecommunication Networks Group, Technische University Berlin, August 2001.

[14] D. Xu, B. Li, K. Nahrstedt, and J. W.-S. Liu, "Providing seamless QoS for multimedia multicast in wireless packet networks," in *Proceedings of SPIE Multimedia Systems and Applications*, (Boston, MA, USA), pp. 352–361, 1999.

[15] B. J. Dempsey, J. Liebeherr, and A. C. Weaver, "A new error control scheme for packetized voice over high-speed local area networks," in *In 18th IEEE Local Computer Networks Conference*, (Minneapolis, MN), pp. 91–100, 1993.

[16] M.Luby, L.Vicisano, J.Gemmell, L.Rizzo, M.Handley, and J.Crowcroft, "RFC 3452 Forward Error Correction (FEC) Building Block."

[17] P. Lettieri, C. Fragouli, and M. B. Srivastava, "Low power error control for wireless links," in *Proceedings of ACM/IEEE MobiCom'97*, pp. 139–150, 1997.

[18] P. J. M. Havinga, "Energy efficiency of error correction on wireless systems," in *Proceedings of the IEEE Wireless Communications and Networking Conference*, September 1999.

[19] R. Xu, Z. Li, C. Wang, and P. Ni, "Impact of data compression on energy consumption of wireless-networked handheld devices," in *Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems (ICDCS'03)*, (Providence, Rhode Island), May 2003.