Energy Aware Distributed Speech Recognition for Wireless Mobile Devices

Brian Delaney[1], Tajana Simunic, Nikil Jayant[1]
Mobile and Media Systems Laboratory
HP Laboratories Palo Alto
HPL-2004-106
June 17, 2004*

low-power, distributed speech recognition, wireless

The use of a voice-user interface for mobile wireless devices has been an area of interest for some time. However, these devices are generally limited by computation, memory, and battery energy, so performing high quality speech recognition on an embedded device is a difficult challenge. In this paper, we investigate the energy consumption of distributed speech recognition (DSR) on the HP Labs SmartBadge IV embedded system and propose optimizations at both the application and network layers that reduce the overall energy budget for this application while still maintaining adequate quality of service for the end-user. We consider energy consumption in both computation and communication. We present software optimization techniques that reduce the energy consumption of the speech signal processing algorithm by 83%. In addition, we estimate the energy consumption of client-side automatic speech recognition without the use of the network. We present a range of results such that the upper bound may match the results of server-based DSR and the lower bound offers reduced functionality (i.e. smaller vocabulary and/or lower accuracy) but with decreased energy usage. In our analysis of DSR, we consider both 802.11b and Bluetooth wireless networks. Given the relatively high bit rates these standards provide with respect to DSR traffic, we investigate the use of synchronous bursty transmission of the data to maximize the amount of time spent in a low-power or off state. The energy savings can be significant even with small, imperceptible delays. With 802.11b, we can reduce the energy consumption of the wireless interface by around 80% with modest application delays of just under half a second. We include the effects of a Rayleigh fading channel in our analysis and investigate the result of bit errors on both energy consumption and DSR accuracy. We have shown that DSR can reduce the required systemwide energy consumption for a speech recognition task by over 95% compared to a software based client-side speech recognition system. These savings include the software optimizations of the DSR front-end as well as the savings from the decreased duty cycle of the wireless interface. We have identified the lower bounds on channel SNR for the various network traffic types and have shown where it becomes advantageous or necessary to perform speech recognition on the embedded device.

* Internal Accession Date Only
[1]Multimedia Communications Laboratory, Georgia Institute of Technology, Atlanta, GA, USA

Energy Aware Distributed Speech Recognition for Wireless Mobile Devices

Brian Delaney, Tajana Simunic, Nikil Jayant

***Abstract*— The use of a voice-user interface for mobile wireless devices has been an area of interest for some time. However, these devices are generally limited by computation, memory, and battery energy, so performing high quality speech recognition on an embedded device is a difficult challenge. In this paper, we investigate the energy consumption of distributed speech recognition (DSR) on the HP Labs SmartBadge IV embedded system and propose optimizations at both the application and network layers that reduce the overall energy budget for this application while still maintaining adequate quality of service for the end-user. We consider energy consumption in both computation and communication. We present software optimization techniques that reduce the energy consumption of the speech signal processing algorithm by 83%. In addition, we estimate the energy consumption of client-side automatic speech recognition without the use of the network. We present a range of results such that the upper bound may match the results of server-based DSR and the lower bound offers reduced functionality (i.e. smaller vocabulary and/or lower accuracy) but with decreased energy usage. In our analysis of DSR, we consider both 802.11b and Bluetooth wireless networks. Given the relatively high bit rates these standards provide with respect to DSR traffic, we investigate the use of synchronous bursty transmission of the data to maximize the amount of time spent in a low-power or off state. The energy savings can be significant even with small, imperceptible delays. With 802.11b, we can reduce the energy consumption of the wireless interface by around 80% with modest application delays of just under half a second. We include the effects of a Rayleigh fading channel in our analysis and investigate the result of bit errors on both energy consumption and DSR accuracy. We have shown that DSR can reduce the required systemwide energy consumption for a speech recognition task by over 95% compared to a software based client-side speech recognition system. These savings include the software optimizations of the DSR front-end as well as the savings from the decreased duty cycle of the wireless interface. We have identified the lower bounds on channel SNR for the various network traffic types and have shown where it becomes advantageous or necessary to perform speech recognition on the embedded device.**

I. Introduction

The demand for tetherless access to data is driving the industry toward smaller but more capable wireless devices. The applications include high-quality wireless web browsing, multimedia e-mail and messaging services, digital music playback, as well as personal data management applications, such as calendar and contact databases. These pocket-sized devices have small screens and tiny keypads, so appropriate use of speech recognition technology can allow users to interact with the system in a natural manner. However, these devices are limited in computation, memory, and battery energy. Complex speech recognition tasks are difficult to perform on the device due to these resource limitations. A typical speech recognition system consists of a signal processing front-end or feature extraction step, followed by a search across acoustic and language models for the most likely sentence hypothesis. The signal processing front-end is a small portion of the overall computation and storage required. The acoustic and language models typically use on the order of tens of megabytes each of storage with significant computation required for large vocabulary search. Therefore, distributing the speech recognition across the network is an attractive alternative for these mobile wireless devices. In the absence of a network connection, some limited speech recognition may be performed on the device.

In distributed speech recognition (DSR), the speech features, typically mel-frequency cepstral coefficients (MFCC), are calculated at the client and sent over the wireless network to a server. Figure 1 shows a block diagram of this system. By only sending the speech data required for machine recognition, we can obtain better accuracy at lower bit rates than traditional human perception-based speech coders. This technique has been well-studied in the literature and is more of a client-server approach than a true distributed computation application. The back-end speech recognition search including HMM state output evaluation and Viterbi search is performed at the server. A true distribution of the workload across many wireless nodes of equal processing capability would likely cause too much wireless

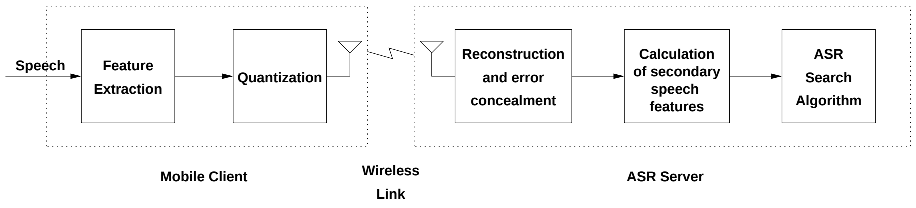


Fig. 1. A distributed speech recognition system.

traffic overhead as HMM output probabilities and Viterbi search best path scores would have to be shared across nodes for many thousands of HMM states with high numerical precision every 10 milliseconds. Therefore, we consider only the client-server approach in distributed speech recognition. In order to minimize the bit rate, the MFCCs are first compressed using some quantization scheme. The result is a three-step process on the mobile client involving computation, quantization, and communication. The resulting text from the speech recognition process can be sent back to the mobile client or handled at the server depending on the nature of the application.

While many cellular phones currently have voice dialing capability, more sophisticated speech recognition tasks require computation capability beyond what these devices currently provide. Applications of speech recognition for embedded devices may include e-mail dictation, web browsing, and scheduling and contact management applications. Small wireless headset microphones such as those commonly used for hands-free cellular communication can be used to send commands to a speech recognition server for command and control within the automobile or other applications requiring hands-free interaction.

One challenge in designing a speech recognition system for a mobile device is minimizing the total energy consumption used in the task. The use of CPU, memory, and the wireless network can cause considerable battery drain if used indiscriminately. In this work, we examine the energy usage in a distributed speech recognition system with respect to the quality-of-service metrics pertinent to this application. We consider both communication- and computation-related energy drain and propose techniques to minimize energy usage in both areas while maintaining a useful level of service for the end-user. Finally, we compare the energy consumption of both client-side speech recognition and DSR using two different network interfaces.

The embedded system used in the experiments is the SmartBadge IV embedded system developed at the Mobile and Media Systems Lab at HP Labs [1]. The SmartBadge contains a 206 MHz StrongARM-1110 processor, StrongARM-1111 co-processor, Flash, SRAM, PCMCIA interface, and various sensor inputs such as audio, temperature, and accelerometers. It runs the Linux operating system. The SmartBadge has speech/audio driven I/O, so speech recognition can provide some level of user interaction through a voice-user interface. It supports a variety of different networking hardware options including Bluetooth and 802.11b wireless interfaces. The StrongARM platform is still used in many high-end PDAs in the market today, such as the HP iPAQ H3800. Table I shows the total average power dissipation of the iPAQ with both 802.11b and Bluetooth transmitting data as well as without the network. The SmartBadge IV uses the same memory and CPU as this version of the iPAQ, but it offers a wider range of hardware based power measurements as well as software simulation tools, therefore it is a better choice to investigate the issues discussed in this paper. Newer PDAs based on the XScale processor have a similar architecture to the StrongARM, and we expect similar results with these processors.

TABLE I
ENGERY CONSUMPTION OF THE HP IPAQ.

Operation	Power Dissipation (mW)
iPAQ (no wireless)	929
iPAQ (802.11b, Tx)	1929
iPAQ (Bluetooth, Tx)	1109

In Section II, we discuss some related work. Section III includes a discussion on the energy consumption of a signal processing front-end as well as an estimation of the energy consumption of client-side ASR. In Section IV, we discuss the energy used in communication for both 802.11b and Bluetooth. Finally, we present a summary in Section V and conclusions in Section VI.

II. Related Work

Earlier work on distributed speech recogntion considered the effects of communication over cellular networks. A method for increased robustness against both impulsive noise and loss over GSM networks was presented in [2]. The effects of using coded speech in ASR was presented in [3]. Low bit rate speech coders, such as those used in cellular telelphony, exhibited significant reductions in ASR accuracy. In an attempt to alleviate the effects of low bit rate speech coders, cepstral coefficients were calculated directly from the wireless bitstream in [4]. While this offered some improvement, a fundamental limitation is that traditional speech coding techniques are aimed at human and not machine listeners. The spectral distortion introduced by speech coding is designed to have minimum impact on human listeners, but speech recognizers rely solely on this spectral information. The result is that currently deployed low-bit rate speech coding techniques are not suitable for high-quality ASR applications.

More recent work on DSR can be grouped into two main areas, those that attempt to design ASR-friendly speech coders, such as the work done in [5], and those that assume to communicate only with a speech recognition system. We consider the latter, where only the spectral information needs to be included, which can result in better performance with lower bit-rates. Additionally, this communication can occur over less expensive or shorter range links such as 802.11b or Bluetooth. Previous work this area has been mainly focused on techniques for the quantization of speech parameters and robustness to loss or errors from wireless transmission. Vector quantization is the dominant compression technique with bit rates in the low kbps range. In [6], a two-stage vector quantizer was used to achieve a fixed rate of 4.0 kbps with little loss of recognition accuracy. A scalable quantization scheme was developed for bit rates ranging from less than 1 kbps to around 3 kbps in [7]. A wider range of quantization schemes was investigated in [8] with the best performance coming from an intra-frame product code vector quantizer. By exploiting the correlation between successive frames of speech, an inter-frame vector quantizer can achieve greater recognition accuracy with lower bit rates as shown in [9]. The ETSI Aurora DSR standard includes a simple intra-frame vector quantizer with some error detection, concealment, and framing techniques in [10]. A low-power DSP solution that uses less than 1mW of power and conforms to the ETSI standard is presented in [11]. However, given that a wireless interface can consume more than half the total power on an embedded device, efficient use of the radio in DSR is an important consideration. This work considers the application of DSR traffic to both Bluetooth and 802.11b networks.

The wireless network power optimization problem has been addressed at different abstraction layers, starting from the semiconductor device level to the system and application level. Energy efficient channel coding and traffic shaping to exploit battery lifetime of portable devices were proposed in [12]. A physical layer aware scheduling algorithm aimed at efficient management of sleep modes in sensor network nodes is illustrated in [13]. Energy efficiency can be improved at the data link layer by performing adaptive packet length and error control [14]. At the protocol level, there have been attempts to improve the efficiency of the standard 802.11b, and proposals for new protocols [15]–[17]. Packet scheduling strategies also can be used to reduce the energy consumption of transmit power. In [18], authors propose the E^2WFQ scheduling policies based on Dynamic Modulation Scaling. A small price in packet latency is traded for the reduced energy consumption. A server-driven scheduling methodology aimed at reducing power consumption for streaming MPEG4 video was introduced in [19]. Savings of as much as 50% in WLAN power consumption, relative to just using 802.11b power management, were reported.

Traditional system-level power management techniques are divided into those aimed at shutting down components and policies that dynamically scale down processing voltage and frequency [20], [21]. Energy-performance tradeoffs based on application needs have been recently addressed [22]. Several authors exploit the energy-QoS tradeoff [23]–[26]. A different approach is to perform transcoding and traffic smoothing at the server side by exploiting estimation of energy budget at the clients [27]. A new communication system, consisting of a server, clients and proxies, that reduces the energy consumption of 802.11b compliant portable devices by exploiting a secondary low-power channel is presented in [28]. Since multimedia applications are often most demanding of system resources, a few researchers studied the cooperation between such applications and the OS to save energy [29]–[32].

III. Modeling the Energy Used in Computation

The computation of speech features is a small portion of the overall speech recognition task in both computation and memory usage. Client-side ASR require more computation and memory bandwidth due to the back-end search algorithm. Table II shows the average cycle count to process one frame of speech in the Sphinx-III large vocabulary speech recognition system. The results were obtained on a 1.4 GHz Pentium 4 workstation. The total processing for the front-end is less than one percent of the overall computation, with the majority of time being spent in the hidden Markov modeling step. Porting a full speech recognition sytem to a mobile device requires more optimization than a simple conversion to fixed-point arithmetic. It involves optimization at many levels, from search space reduction to fast arithmetic kernels and techniques to reduce memory bandwidth. For these reasons, we concentrate our software optimization on the signal processing front-end only, and estimate the full client-side ASR energy usage by using some published results [33].

TABLE II
CYCLE COUNTS FOR THE FRONT-END, GAUSSIAN EVALUATION, AND VITERBI SEARCH PORTIONS OF SPEECH RECOGNITION.

Module	Avg. Cycles/Frame	% of total
Front-End	7.22×10^4	0.4%
Hidden Markov Model	1.21×10^7	32.63%
Viterbi Search	5.88×10^6	66.97%

Client-side ASR may be necessary to maintain interactivity with the mobile device when the network is not present. We compare these results with DSR under various channel conditions, error correction methods, and packet sizes to show the benefits of DSR from an energy consumption perspective. Through the use of algorithmic and architectural optimization in software, we can reduce the energy consumption of the signal processing front-end by 83%. These savings can be enhanced by the use of runtime dynamic voltage scaling (DVS) techniques.

A. Signal Processing Front-End

The acoustic observations generated by the signal processing front-end or "feature extraction" step are mel-frequency cepstral coefficients. Mel-frequency cepstral coefficients are calculated using the real cepstrum, defined as the inverse Fourier transform of the log spectrum:

$$c_s(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log |S(\omega)| \, e^{j\omega n} d\omega \tag{1}$$

where $S(\omega)$ is the spectrum of the speech signal. The most common set of features consists of 13 mel-frequency cepstral coefficients computed every 10ms. Secondary features, consisting of first and second time derivatives of the cepstrum, also are used, but they can be calculated easily at the server. A more in depth discussion of the theory and properties of the cepstrum can be found in [34].

In practice, the calculation of MFCCs requires filtering and windowing operations, a magnitude FFT calculation, a filter-bank operation, a logarithm operation, and a discrete cosine transform. Implementing the front-end feature extraction for a distributed speech recognition system on an embedded platform requires not only speed, but also power optimization, since the battery life-time in such devices is very limited. This work discusses both the source-code and the run-time optimizations.

The source code optimizations can be grouped into two categories. The first category, architectural optimizations, aims to reduce power consumption while increasing speed by using optimization methods targeted to a particular processor or platform (e.g. an embedded system with no floating-point hardware). Ideally, many of these optimizations should be done by a compiler. However, currently available compilers for most embedded systems do not have these optimizations built-in. In addition, measurements presented in [35] show that the improvements that can be gained using standard compiler optimizations are marginal compared to writing energy efficient source code. The second category of source code optimizations is more general and involves changes in the algorithmic implementation of the source code with the goal of faster performance with less power consumption.

The final optimization presented in this work, dynamic voltage scaling (DVS), is the most general since it can be applied at run-time without any changes to the source code. Dynamic voltage scaling algorithms reduce energy consumption by changing processor speed and voltage at run-time depending on the needs of the applications running. The maximum power savings obtained with DVS are proportional to the savings in frequency and to the square of voltage.

1) Architectural Optimization: Signal processing algorithms, such as the calculation of the mel-frequency cepstrum, are generally mathematically intensive, therefore a significant amount of effort was spent in optimizing the arithmetic. In addition, simple C code optimizations were employed to help the compiler generate more efficient code [36].

Profiling of the source code on a StrongARM simulator revealed that over 90% of the time was spent in floating-point emulation. The StrongARM has no on-chip floating-point processor, so all floating-point operations must be emulated in software. Simply changing from double to single precision floats improved the performance considerably. However, profiling showed that 80% of the time was still being spent in floating point emulation. Any further gains require fixed-point arithmetic.

Fixed-point arithmetic uses scaled integers to perform basic math functions using the existing integer hardware. The scaling factor (or location of the decimal point) is fixed at design time and is designated by Qn, where n is the number of bits to the right of the decimal. The basic rules of arithmetic still hold; adding two numbers requires that the decimal points must line up. Multiplying two numbers in Qn format yields a number in $Q2n$ format.

Implementing a pre-emphasis filter and Hamming window using fixed-point arithmetic is straight-forward. Fixed-point FFTs are well studied and have often been implemented on digital signal processor chips. After passing the input frame through the FFT, the mel filter bank must be applied. The filter bank amplitudes are calculated using the squared magnitude. This presents some challenges since this squared number multiplied by the filter coefficients, $H_i[k]$, can easily overflow the 32-bit registers. A 64-bit result can be obtained from the StrongARM multiplier using assembly language, but overflow can be avoided simply by rewriting the filter bank equation to use just the magnitude:

$$Y[i] = \sum_{k=0}^{N/2} \left(|X[k]| \sqrt{H_i[k]} \right)^2 \qquad (2)$$

This avoids overflow since $H_i[k] \ll 1$, therefore the result of each multiplication is small. The coefficients, $\sqrt{H_i[k]}$, are stored in a lookup table.

The one drawback to this method is that computing the magnitude spectrum requires a square root operation. Fast integer square root algorithms exist, but they must be used on each output from the FFT, which is costly. Fortunately, the magnitude can be estimated as a linear combination of the real and imaginary parts using the following equation [37]:

$$|x| \approx \alpha \max(|\Re\{x\}|, |\Im\{x\}|) + \beta \min(|\Re\{x\}|, |\Im\{x\}|) \qquad (3)$$

where α and β are chosen to minimize a particular kind of error, and $\Re\{x\}$ and $\Im\{x\}$ represent the real and imaginary parts of the complex number x. This formula rotates a complex phasor to between 0 and $\pi/4$ radians and then takes a linear combination of the real and imaginary parts. The values of α and β are chosen to have an easy fixed-point representation that minimizes the mean error.

Computing the first 13 coefficients of the DCT is relatively easy to do in fixed-point arithmetic, but taking the natural logarithm is a more difficult task. However, there is an interesting algorithm to estimate $\log_2(x)$ using simple bit manipulation, which is faster than other methods of calculating the logarithm. This algorithm, described in [38], is very low in complexity and gives an approximate fixed-point result. The $\ln(x)$ can be determined by multiplying by a constant as follows:

$$\ln(x) = \log_2(x)\ln(2) \qquad (4)$$

One final adjustment must be made when x is itself a fixed-point number in Qn format, which is just a scaled integer:

$$\ln\left(\frac{x}{2^n}\right) = [\log_2(x) - \log_2(2^n)]\ln(2) \qquad (5)$$

$$\ln\left(\frac{x}{2^n}\right) = [\log_2(x) - n]\ln(2) \qquad (6)$$

Equation (6) is the expression used to calculate the natural log of a fixed-point number. Using precision of Q3, this estimate of the logarithm has a maximum error of around 0.152 and an average error of around 0.0866.

2) Algorithmic Optimization: Profiling of the original source code under a StrongARM simulator revealed that most of the execution time was spent in the computation of the DFT (which is implemented as an FFT). Since speech is a real-valued signal, an N-point complex FFT can be reduced to an $N/2$-point real FFT [39]. Some further processing of the output is required to get the desired result, but this overhead is minimal compared to the reduction in computation. Additional savings can be obtained when the trigonometric functions used in the computation of the FFT are pre-computed and stored in a lookup table, thus eliminating multiple function calls in the FFT loop.

3) Dynamic Voltage Scaling: Once the code is optimized for both power consumption and speed, further savings are possible by changing the processing frequency and voltage at run-time. In this work, we investigate the savings possible with DVS for the front-end of a speech recognizer running on Smartbadge IV hardware. The StrongARM processor on Smartbadge IV can be configured at run-time by a simple write to a hardware register to execute at one of eleven different frequencies. Note that the number of frequencies is predefined by the design of the StrongARM processor. We measured the transition time between two different frequency settings at 150 microseconds. Since typical processing time for the front-end is much longer than the transition time, it is possible to change the CPU frequency without perceivable overhead. For each frequency, there is a minimum voltage the SA-1110 needs in order to run correctly, but with lower energy consumption. The easiest way to determine the lowest possible frequency and voltage for such stand alone application is to run it at all possible frequency settings, with voltage set to minimum allowed, and observe if the code still runs in real time. In our case, we obtained real time performance at all possible frequency and voltage settings.

4) Software Optimization Results: Three main criteria are considered in order to evaluate the effectiveness of a particular optimization: performance (in terms of processor cycle count), energy consumption, and accuracy or word error rate (WER). Simulation results for processing one frame (25ms) of speech on the Smartbadge IV architecture running at 202.4 MHz are shown in Figure 2. The x-axis shows the source code in various stages of optimization. The "baseline" source code contains no software optimizations. The "optimized float" code contains the set of optimizations described in section III-A.2 as well as some of the C source optimizations described in [36]. Double precision floating-point numbers were changed to single precision 32-bit floats in the "32-bit float" version of the code. Finally, the "fixed-point" implementation contains all of the source code optimizations described in this paper. For each version of the code, we report the performance (in CPU cycles) and the total battery energy consumed (in μWhrs). The simulation results are computed by the cycle-accurate energy simulator, and include processor core and level 1 cache energy, interconnect and pin energy, energy used by the memory, losses from the DC/DC converter, and battery inefficiency [35]. The reduction in energy consumption is not as dramatic as the performance improvement for the fixed-point version due to an increase in memory references per unit time. In fixed-point code, basic math operations are reduced to a few cycles as opposed to long iterations of floating-point emulation which do not require as many memory references. However, we have still achieved a reduction in the total battery energy required to process one frame of speech data by 83.5%.

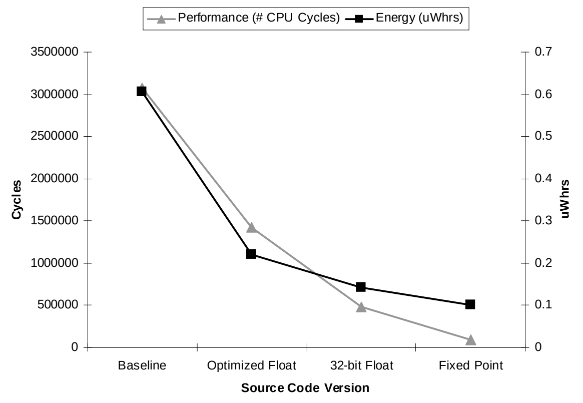


Fig. 2. Performance and energy consumption per frame of speech.

To verify that the approximations used in this software optimization do not introduce significant noise to the output speech parameters, we tested the front-end with a digit recognition task. The results are shown in Table III. A continuous digit speech recognizer was trained using the TIDIGITS database of 8,623 utterances from both male and female speakers. The original floating point front-end was used to generate mel-frequency cepstral coefficients for the training set. No secondary features (first and second time derivatives of the mel-frequency cepstrum) were used in the training or test phases. The trained speech models were then used to recognize speech from the TIDIGITS test set of 8,700 utterances. The WER was calculated using the various front-end implementations and is shown in Table III. There is no loss in accuracy among the three floating-point implementations, but the fixed-point implementation uses some approximate algorithms which can create a slight mismatch between the training and test data. We were able to eliminate the slight 0.1% increase in WER by using the fixed-point front-end during the training phase. In addition, Table III shows a minimal increase in lookup table size and code size, so the memory requirements for the fixed-point optimized code are about the same. Another performance metric reported in Table III is how long it took for each code implementation to process 1 second of speech at the processor clock speed of 202.4 MHz (Time column). The fixed-point version runs 34 times faster than the baseline system.

TABLE III
TIDIGITS TEST SET RESULTS.

	Code size (Bytes)	Lookup table (Bytes)	Time (sec)	WER %
Baseline	29704	N/A	1.510	4.2%
Optimized Float	31960	88120	0.699	4.2%
32-bit Float	31272	88120	0.235	4.2%
Fixed-Point	33124	88136	0.043	4.3%

Because the fixed-point code runs much faster than real-time at 202.4MHz, it is possible to get further reductions in power usage by using DVS as discussed in section III-A.3. The results from this experiment are shown in Table IV. These power measurements are performed on the Smartbadge IV system running the eCos embedded operating system and using the WaveLAN card to transmit the uncompressed cepstral parameters. The P_{sys} measurement is taken from the main power supply output. At 59 MHz the algorithm still runs in real-time, and the system uses 34.7% less power than at 206 MHz. Combining the DVS results with the source code optimizations, we calculate the overall reduction in power consumption to be 89.2%.

5) Vector Quantization: Finally, we include the fixed-point vector quantization code in our profiling and consider different bit rates and quantization levels. Although some differing techniques have been proposed, the most

TABLE IV

MEASURED POWER CONSUMPTION WITH DVS.

Frequency (MHz)	Voltage (V)	P_{sys} (mW)
59	0.78	1721
74	0.94	1807
89	1.09	1901
103	1.21	2029
118	1.33	2114
132	1.42	2234
147	1.51	2320
162	1.57	2432
176	1.63	2508
191	1.67	2568
206	1.69	2636

common technique for compressing MFCCs is some form of vector quantization. In vector quantization, we train a set of codebooks against some speech data. These codebooks contain a set of centroids representing the clusters that occur in the training data. We simply transmit the centroid index for each codebook. Smaller codebooks will result in a noisy representation of the original signal, and speech recognition accuracy will degrade.

For our system, we used an intra-frame product code vector quantization scheme presented in [8]. We used the existing bit allocation in [8] to train a set of codebooks using a K-means training algorithm with bit rates ranging from 1.2 kbps to 2.0 kbps. We have added an additional bit allocation that is similar to the ETSI standard that will operate at 4.2kbps [10]. The actual bit rate needed for a speech recognition task depends on many factors such as acoustic and speaker conditions as well as the vocabulary size and complexity of the acoustic models used. In [8], the range of bit rates was evaluated for a small vocabulary task under ideal acoustic conditions. We can expect the word error rate (WER) to increase under less ideal conditions (i.e. larger vocabulary, more acoustic background noise, etc.) Table V shows the resulting bit rates and word error rates from [8] on the rows labelled VQ-XX, where XX is the number of bits per 10 ms speech frame. The WER for full bandwidth speech at 16 kHz and 16 bits per sample (256 kbps) was 6.55%.

Source code to perform the quantization of the MFCC data was written in fixed point for the StrongARM processor and profiled using the energy consumption simulator. The total energy consumption required to calculate MFFCs for one frame of speech including vector quantization at 4.2 kbps is approximately 380 μJoules. Even at the highest bit rate, the vector quantization is only 12% of the total energy budget. This suggests that speeding up the quantization process by using smaller codebooks would produce minimal reductions in energy consumption and would have a much greater impact on speech recognition accuracy.

TABLE V

WORD ERROR RATE FOR SEVERAL BIT RATES [8].

Description	Bit rate (kbps)	WER (%)
VQ-12	1.2	16.79
VQ-14	1.4	11.71
VQ-16	1.6	9.3
VQ-18	1.8	8.1
VQ-19	1.9	6.99
VQ-20	2.0	6.63
VQ-42	4.2	$\approx$ 6.55
16kHz	256	6.55

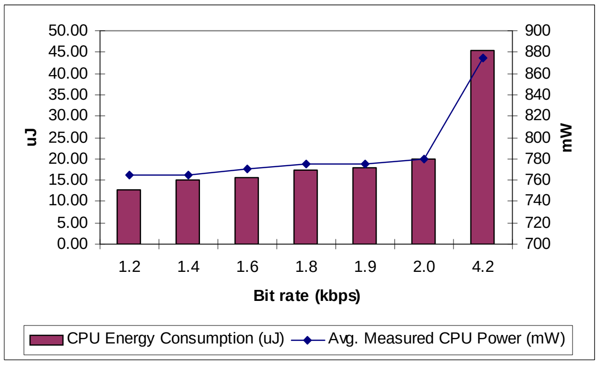


Fig. 3. Computational energy usage and measured average power for different quantization bit allocation schemes.

Figure 3 shows a comparison of energy consumption for various vector quantization bit allocation schemes. The bars represent the total energy consumption per frame of speech for the quantization step, and the line represents the measured CPU power dissipation at each bit rate. The measured values closely match the results from the energy consumption simulator. Those with the smaller bit rates (i.e. 1.2 kbps to 1.6 kbps) offer the poorest speech recognition performance and do not save very much battery energy when compared with the overall computation. There is approximately a 14% increase in CPU power consumption but a greater than 50% reduction in WER between the highest and lowest bit rates. Therefore, we advocate the use of higher and more robust bit rates since the reduction in energy consumption is minimal.

B. Client-Side ASR

In the absence of a network connection it may be necessary to perform ASR on the mobile device. Speaker-dependent speech recognition engines have been optimized for the StrongARM or other mobile processors by many industry players, but it has been shown that they use most available resources and may run several times slower than real-time for many tasks. Power measurements for an embedded dictation ASR system running on a StrongARM based processor are given in [33]. The ASR system ran just over 2.5 times real-time, and the processor was almost never idle during the task.

For the purposes of this work, it is sufficient to describe the energy requirements for local ASR as the product of the average power dissipation of the processor and memory under load and the time required to perform the speech recognition task. For the Smartbadge IV, we have measured the average CPU and memory power dissipation as P_{cpu} = 694 mW and P_{mem} = 1115 mW when under load. Given the real-time factor R for the speech recognition task, we can estimate the energy consumption to recognize one frame of speech as:

$$E_{local} = (P_{cpu} + P_{mem}) \times R \times \frac{1}{100} \tag{7}$$

Therefore, for a speech recognition task that runs $R = 2.5$ times slower than real-time, we can expect to use approximately 45 mJ of battery energy to process one frame of speech. Compare this with just under 0.4 mJ for the front-end only, and we have a difference in computation energy of several orders of magnitude for client side ASR vs. the distributed system. By using smaller vocabularies and simpler acoustic and language modeling techniques, it should be possible to lower the total run-time and energy consumption at the cost of reduced performance. A reduced ASR task running in real-time on a SmartBadge IV would use approximately 18 mJ of energy per frame of speech, but the tradeoff is reduced utility for the end-user.

IV. Modeling the Energy Used in Communication

The wireless network can use significant amounts of energy on a mobile device. Measurements on the SmartBadge IV hardware show that an 802.11b interface card can use up to 45% of the total power budget. Reducing the energy consumption is an important consideration and has been well studied. Section II outlines some of the techniques. We consider both 802.11b and Bluetooth wireless networks in our analysis. We assume single hop communication with a speech recognition server connected to a wired network via a wireless access point. Multi-hop communication has limited utility for this application as it is a client-server scenario over limited range wireless links. Distributing the computation across a set of equally constrained mobile devices is not considered here.

Given the relatively low bit rates used in DSR, both of these networks will operate well below their maxium throughput range. In this situation, more energy saving opportunities will develop from exploiting moderate increases in application latency by transmitting more data less often. This allows the network interface to either be powered down or placed into a low-power state in between transmissions. Other wireless networks with throughput in the low kbps range, such as many cellular telephony networks, may require other techniques, such as better compression, to minimize energy consumption. However, we do not consider such wireless networks here.

In order to estimate the power consumption for wireless transmission, we directly measured the average current into the network interface. These measurements were performed under ideal conditions with no competing mobile hosts or excessive interference. Using these measurements as a baseline, we are able to tailor a simple energy consumption model to investigate the effects of increased application latency. By buffering compressed speech features, we maximize the amount of time spent in the low-power or off state. We introduce a power on/off scheduling algorithm for the 802.11b device that exploits this increased latency. Given the medium access control (MAC) scheme for both 802.11b and Bluetooth, we can incorporate the effects of channel errors into the energy model. We use these results to investigate which techniques should be used to maintain a minimum quality of service for the speech recognition task with respect to channel conditions.

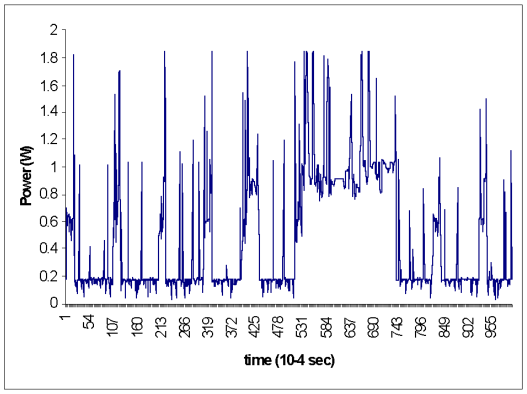
A. 802.11b Wireless Networks

The 802.11b interface operates at a maximum bit rate of 11 Mbps with a maximum range of 100 meters. The MAC protocol is based on a carrier sense multiple access/collision avoidance scheme, which includes a binary exponential backoff system to avoid collision. It uses an automatic repeat request (ARQ) system with CRC error detection to maintain data integrity. We used a PCMCIA 802.11b interface card and measured the average current going into the interface to get the power dissipation.

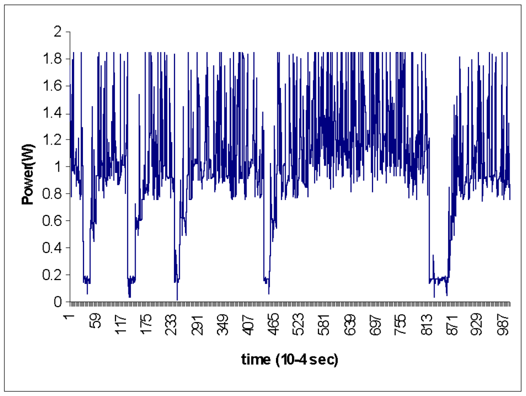
Our measurements indicate that there is only a difference of a few mW in power consumption between the highest and lowest bit rates. This is expected since the bit rates are low, and the transmit times are very short. Also, the use of UDP/IP protocol stacks and 802.11b MAC layer protocols both add significant overhead for small packet sizes. The 11 Mbps WLAN interface is underutilized with this type of low bit rate traffic. However, we can obtain some improvement in power consumption by increasing the number of frames per packet.This increases the total delay of the system, but less battery energy is used since the various networking overhead is amortized across a larger packet size. However, due to the relatively high data rates provided by 802.11b, the WLAN interface spends most of its time waiting for the next packet to transmit. The 802.11b PM mode can provide some savings in energy consumption but this does not hold under heavy broadcast traffic conditions [19], defined as a higher than average amount of broadcast packets. We present an on/off scheduling algorithm to reduce the total energy consumption of the 802.11b device under these conditions. While operating in the 802.11b power management mode, a WLAN card goes into an idle state. Every 100ms it wakes up and receives a traffic indication map, which is used to indicate when the base station will be transmitting data to this particular mobile host. With heavy broadcast traffic, the WLAN interface will rarely be in the idle state and it will consume power as if it were in the always-on mode. This is because the time required to analyze the broadcast packets is larger than the sleep interveral. This increase in power consumption will happen even if there are no applications running on the mobile host. Figure 4 shows the power consumption of the WLAN card in the 802.11b power management mode in both heavy and light traffic conditions. Notice that in the bottom graph, under heavy traffic, the card is unable to transition to the low-power idle state very often. The average power approaches the always-on mode.

Since the energy consumption of PM mode on 802.11b networks breaks down in heavy traffic conditions, we consider an alternate algorithm. If we are only interested in transmitting speech recognition related traffic and not any other broadcast traffic, we can simply power off the WLAN card until we have buffered enough data to transmit. However, powering the card on and off has an energy related cost that needs to be accounted for.

Figure 5 shows the timing of this scheduling algorithm. The period, T, is determined by the number of speech frames sent in one packet. The transmission is synchronous such that every T seconds we will send that amount of compressed speech features and stay in the off state for the remainder of the time. With larger values of T we can hope to amortize the cost of turning the WLAN card on and off at the expense of longer delay. Assuming that a speech recognizer server is able to process speech at or near real-time, the user will experience delay near the value of T. For an interactive application the total delay seen by the user begins when the user stops speaking and ends when some action



(a) light traffic



(b) heavy traffic

Fig. 4. WLAN power consumption in 802.11b PM mode in light and heavy traffic conditions.

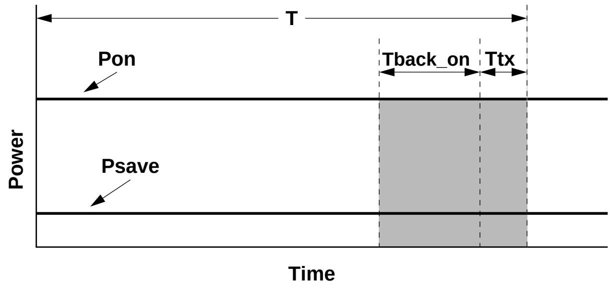


Fig. 5. The timing of the 802.11b scheduling algorithm.

is taken by the mobile device. A server that is able to process speech faster than real-time will be able to reduce this delay but not eliminate it completely. The amount of tolerable delay depends on the application. For user interface applications, such as web browsing, a calendar application, or a voice-driven MP3 player, it is important to reduce the delay to maintain interactivity. Delays of around one second may hardly be noticed by the user, whereas delays of around three seconds or more may hinder interactivity. For a dictation application, such as e-mail, this delay is less important. In this case, the use simply dictates a response, and corrections or edits can occur after the speech-to-text process is complete.

Assuming the average power for the always on WLAN mode is P_{on}, the total energy required to transmit T seconds of speech frames can be estimated as:

$$E_{on} = P_{on} \times T \quad (8)$$

Similarly, the total amount of energy required to transmit in power management mode is:

$$E_{save} = P_{save} \times T \quad (9)$$

where both P_{save} and P_{on} are the measured average power values at the particular bit rates and number of speech frames per packet. These data values were measured directly off the WLAN hardware.

Using the proposed scheduling algorithm, the WLAN card will be on only during the shaded region in figure 5. The value, T_{back_on}, is the amount of time required to turn the WLAN card back on, during which time it uses power as if it were transmitting. The value T_{tx} is the total amount of time required to transmit the data, which is typically much smaller than T_{back_on} for the low bit rates required for speech traffic. The energy required to transmit under the proposed scheduling algorithm is:

$$E_{sched} = P_{on} \times (T_{back\_on} + T_{tx}) \quad (10)$$

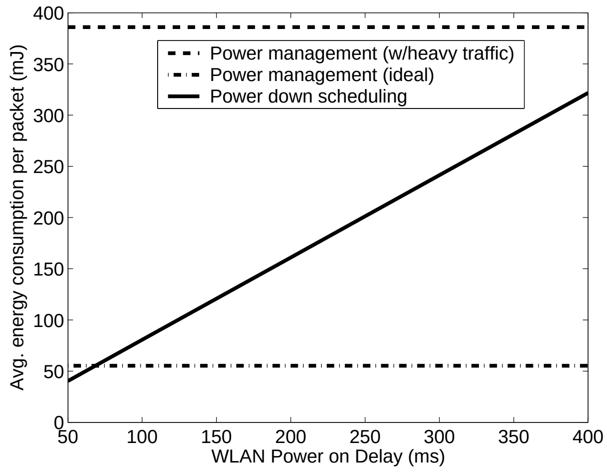


Fig. 6. WaveLAN power on delay vs. energy consumption per packet.

The two interesting parameters to consider are the power on time (T_{back_on}) and the number of speech frames transmitted at once, which dictates the total period T. Figure 6 shows the power on delay on the x-axis and estimated energy consumption on the y-axis. We fixed the value of T to 0.48 seconds, or 48 frames of speech data. The PM mode configuration in light traffic almost always outperforms the proposed scheduling algorithm except for very small values of T_{back_on}. (Typical values may range from 100ms to 300ms.) However, in heavy traffic conditions, the PM mode approaches the always on power consumption (shown by the top line in the plot), so the scheduling algorithm can give better performance under these conditions. With T_{back_on} at 100ms, the total energy consumption per packet is approximately 75 mJ for the scheduling algorithm and approximately 390 mJ for PM mode in heavy traffic conditions (from figure 6). This is a reduction in energy consumption by about 80%. However, this only holds true for heavy broadcast traffic conditions, so the mobile device will have to monitor the broadcast traffic and decide between the standard 802.11b PM mode or the scheduling algorithm.

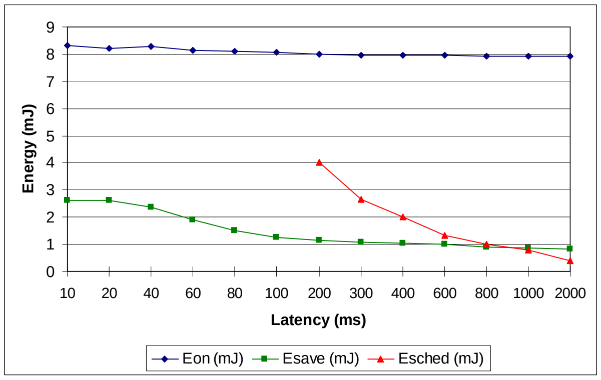


Fig. 7. Average energy consumption per 10ms speech frame vs. DSR latency for various 802.11b power save schemes. (WLAN power on delay is fixed at 100ms.)

Finally, we consider increased delay or latency, T, in Figure 7. with T_{back_on} fixed at 100ms. In this plot, the energy cost was determined using measured values of power consumption. The energy cost has been normalized to show the average energy required to transmit one frame of speech data. As the total number of frames approaches 80 ($T = 800ms$), we can see that the scheduling algorithm (E_{sched}) will be able to outperform the PM mode configuration (E_{save}) regardless of traffic conditions. This will result in less than one second of delay for a user interface application with speech recognition. Shorter power on (T_{back_on}) times can help move this crossover point to shorter delays. Longer delays of two seconds or more can further reduce energy consumption and are good candidates for applications requiring lower interactivity such as dictation.

Since the 802.11b MAC protocol uses an automatic-repeat-request (ARQ) protocol with CRC error detection to maintain data integrity, the energy consumption will be a function of channel SNR. After the reception of a good packet, an ACK is sent across a robust control channel. For a given bit error rate and packet length, the probability of a packet error in the absence of any error correction coding techniques is:

$$P_r = 1 - (1 - BER)^L \tag{11}$$

where L is the packet length, and BER is the bit error probability for the current channel conditions. For our analysis, we used the BER probability for 256-QAM modulation in a Rayleigh fading channel to approximate the 802.11b CCK modulation. A Rayleigh fading channel models the effect of time-varying multipath fading of the received signal by accounting for constructive and destructive interference of the scattered carrier signal. The Rayleigh fading channel asssumption is widely used in wireless communications literature as a more realistic alternative to an additive white Gaussian noise channel [40]. The BER expression for the 256-QAM modulation is:

$$BER = \frac{2^{k-1}}{2^k - 1} \sum_{m=1}^{M-1} \frac{(-1)^{m+1}\binom{M-1}{m}}{1 + m + 2m\bar{\gamma_b}} \tag{12}$$

where $\bar{\gamma_b}$ is the SNR per bit, $M = 8$, and $k = 4$.

Given the probability of retransmission (P_r), the expected number of retransmissions (T_r) is given by [41]:

$$T_r = \frac{1}{1 - P_r} \tag{13}$$

Using these equations, an energy model can be constructed that incorporates the energy used in the MAC overhead as well as the energy required for repeated retransmissions, assuming the average SNR remains the same. Such an energy model is presented in [42] and is summarized here:

$$\begin{aligned} E_{tx}(BER, L) = E_{aq} + T_{ack} \times P_{rx} + \\ (E_{aq} + T_{tx} \times P_{tx}) \times \tfrac{1}{(1-BER)^L} \end{aligned} \tag{14}$$

where E_{aq} is the average energy required to acquire the channel, T_{ack} is the time required to receive the ACK packet, and P_{rx} is the receive power for the robust control channel. Given this energy model, we can incorporate it into our scheduling algorithm model in (10) as follows:

$$E_{sched} = E_{tx}(BER, L) + P_{on} \times T_{back\_on} \tag{15}$$

We use this expression in Section V to quantify the energy consumption of 802.11b vs. channel SNR. In particular, we show how larger packet sizes and lack of error correction techniques force 802.11b to operate in higher channel SNR. However, techniques such as packet fragmentation and error correction can be used to extend the lower SNR range of 802.11b.

B. Bluetooth Personal Area Network

The Bluetooth personal area network provides a maximum bit rate of 1 Mbps, and a variety of different packet types are available to support different traffic requirements [43]. It supports a range that is considerably less than 802.11b, on the order of 10 meters. Bluetooth supports both data and voice traffic packets as well as a hybrid packet containing both voice and data. Media access is handled via a time-division duplex (TDD) scheme where each time slot lasts 625 μseconds. Voice packets are given priority over data packets in scheduling. In this work, we consider only pure voice or pure data packets. Data packets are available in both high rate and medium rate packets. These are DHn or DMn packets for both high and medium data rate respectively, where n depicts the number of TDD slots the packet occupies: 1, 3, or 5. High rate packets use a stop-and-wait automatic-repeat-request (ARQ) protocol with CRC error detection within the packet. Medium rate packets use a 2/3 rate (15,10) shortened Hamming code in addition to the ARQ protocol. Voice packets, due to their time-sensitive nature, do not use an ARQ protocol. Voice packets are available in HV1, HV2, or HV3 types, where the number denotes the amount of error correction rather than slot length. All voice packets occupy one TDD slot with varying data payloads. HV3 packets use no error correction. HV2 packets use the (15,10) Hamming code, and HV1 packets use a 1/3 rate repetition code. Given the soft time deadlines with speech data intended for a machine listener, we can easily use either data packets or voice packets without consideration of packet jitter or delay characteristics.

First we develop a simple model for the energy consumption of a single Bluetooth voice or data packet. We then consider the use of Bluetooth power saving modes to reduce the energy consumption during the idle time, similar to the 802.11b scheduling algorithm. Finally, we investigate the implications of bit errors on both voice and data packets.

Based on the packet types and various error correction overhead, we can construct a simple energy model for Bluetooth packet transmissions. For voice packets, the total energy used is the power used in transmission multiplied by the time required to transmit.

$$E_{HVn} = P_{tx} \times T_{tx} = P_{tx} \times 625\mu s \tag{16}$$

where P_{tx} is the measured power consumption in the transmit state, and T_{tx} is the total time required to transmit (625 μs for HVn packets). Because of the error correction overhead, we need to transmit three times as many HV1 packets as HV3 packets for the same amount of user data.

For data packets, the energy consumption is dependent on the size of the data packet being transmitted. Data packets occupy either 1, 3, or 5 TDD slots. An estimate of the total energy required to transmit a data packet (Dxn) is:

$$E_{Dxn} = (P_{tx} \times 625\mu s \times n) \quad (17)$$

where n is the slot length of the packet, either 1, 3, or 5.

Using power measurements of a USB Bluetooth device attached to the SmartBadge IV, we are able to estimate the energy usage for our system. Figure 8 shows the energy required to transmit one frame of speech data at various DSR compression rates over a Bluetooth link. We consider the use of both high speed and medium speed data packets. We assume an error-free channel with no retransmissions. We can see in figure 8 that there is a higher energy cost for medium rate packets due to the FEC overhead. However, these packets will be a better choice for lower SNR conditions. Energy consumption approximately doubles between the 1.2 kbps and 4.2 kbps bit rates. However, these estimates do not consider idle time between packets which will consume energy as well.

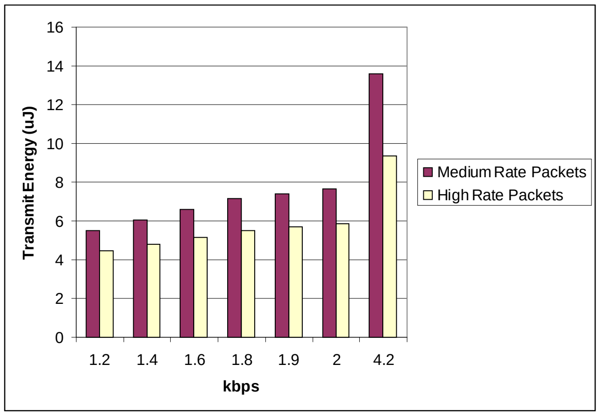


Fig. 8. Energy used to transmit one frame of speech with varying compression rates for Bluetooth radio.

We can incorporate the Bluetooth power saving modes into our model to account for the idle time in between packets. A node within a Bluetooth piconet can operate in a variety of different power management modes [43]. In the default *active* mode, the slave node listens to every master-slave slot to see if the packet is addressed to it. In the *sniff* mode, the node only listens to slots at specified intervals. In *hold* mode, the node goes into a low-power state until some specified interval, after which it powers up to transmit. In *park* mode the Bluetooth node temporarily gives up its membership to the piconet to join a list of parked nodes. The node's only activity in parked mode is to periodically listen for synchronization and broadcast packets.

A Bluetooth node in park mode will wake up upon activity to transmit some data and then enter the park mode when finished. The energy consumption of this scenario is as follows:

$$E = P_{tx} \times T_{tx} + E_{transition} + P_{park} \times T_{park} \quad (18)$$

where $E_{transition}$ is the total energy used to transition to/from the various operating states, and P_{park} and T_{park} are the power dissapation and times in the park mode respectively. The time spent in the deep sleep state is a function of the overall latency of the system and the amount of data being transmitted. We measure 0.18 watts in the transmit mode, and 0.077 watts in the park mode. Transition times to and from the park state are on the order a several milliseconds each.

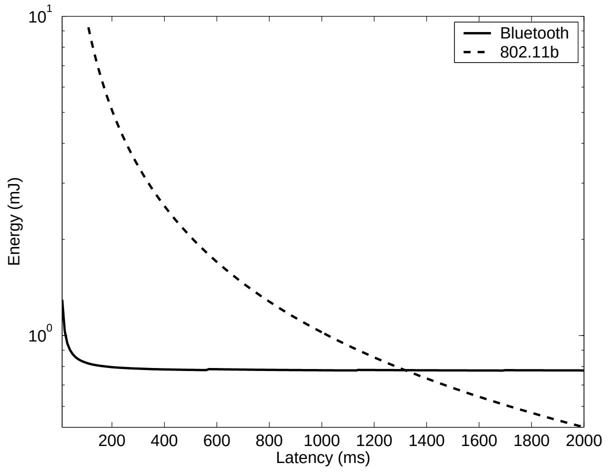


Fig. 9. Energy per frame of speech vs. DSR latency for a Bluetooth and 802.11b.

By varying the amount of data transmitted at once, we can increase the amount of time spent in the park state. Figure 9 shows the tradeoff between speech recognition latency and energy consumption per frame of speech for both 802.11b and Bluetooth. Once again, we assume a perfect channel with no bit errors. For smaller values of T, Bluetooth can offer better performance than 802.11b, but as T approaches 1.3 seconds, 802.11b will use less energy. This is because the 100 ms startup cost of 802.11b is amortized across a larger number of frames, while the Bluetooth node remains in the park state and still consumes power. Powering off a Bluetooth node between packet transmissions is not an option since the paging/inquiry actions required to join a piconet can easily take in excess of 10 seconds. However, since the transition time from park to active and back is small, we see an initial drop in energy consumption with respect to increased latency in Bluetooth. However, with increased delay the energy spent in park mode becomes the dominant factor.

Next, we investigate how the presence of bit errors on the wireless channel will affect both the energy consumption and, in the case of voice packets, speech recognition accuracy. We use this data to identify which types of packets can be used effectively in various channel conditions. The main difference between the two types of packets is that voice packets rely only on FEC and no ARQ, while data packets can use *both* FEC and ARQ.

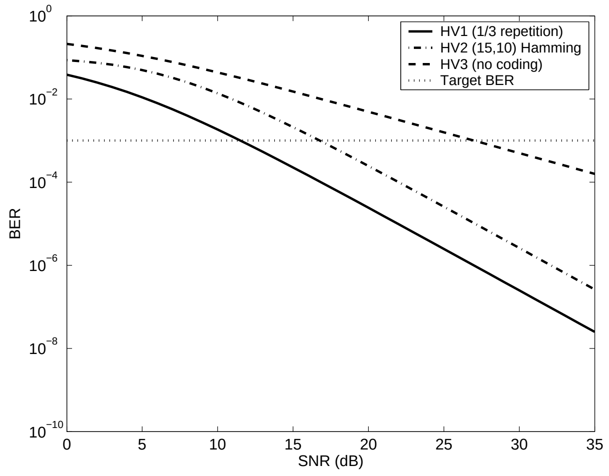


Fig. 10. The error correction performance of Bluetooth voice packet types.

The energy consumption of Bluetooth voice packets is independent of channel conditions. Therefore, we can estimate the energy consumption using (17) and (18). The main difference in energy consumption per frame of speech will come from the reduced user payload due to FEC bits. Figure 10 shows the error correction performance for various types of Bluetooth voice packets. We define a maximum bit error rate that is necessary to maintain a usable level of accuracy for DSR in Section IV-C.

In the presence of bit errors, data packets will continue to be retransmitted until they are received correctly or a timeout occurs. For the purposes of this analysis, we assume BFSK modulation with coherent detection under a Rayleigh fading channel. We also assume that the average SNR remains constant throughout the transmission. The BER expression used is as follows:

$$BER = \frac{1}{2}\left(1 - \sqrt{\frac{\bar{\gamma_b}}{2 + \bar{\gamma_b}}}\right) \tag{19}$$

where $\bar{\gamma_b}$ is the average SNR per bit. The total energy used is also a function of the probability of a packet retransmission. The expression is based on the probability of packet synchronization failure, header failure, payload error, and both synchronization and header failure in the ACK packet. Each of these items is a function of the bit error rate (BER), which is, in turn, a function of the channel signal to noise ratio (SNR). An expression for this probability (P_r) is derived in [44], but for length reasons the expression will not be shown here. If we ignore the overhead for receiving an ACK packet, an estimate of the energy consumption for Bluetooth data packets in the presence of bit errors is:

$$E_{Dxn} = P_{tx} \times 625\mu s \times n \times \frac{1}{1 - P_r} \tag{20}$$

By dividing the energy by the number of frames in a packet, which varies with packet length and coding technique, we can get the energy required to send one frame of speech.

C. The Effect of Bit Errors on DSR

The presence of bit errors in the speech feature stream can cause an significant decrease in accuracy. It is essential that bit errors be detected and concealed when possible [9]. However, it becomes more difficult to conceal errors when lost packets consist of many frames of speech. The correlation between neighboring speech frames decreases with increasing lag. Therefore, for long packets containing half a second or more of speech data, we cannot tolerate much, if any, packet loss. However, with Bluetooth voice packets, the data is delivered even if in error. Using the various error protection schemes in Figure 10, we should be able to indentify a minimum BER after coding that is sufficient for DSR.

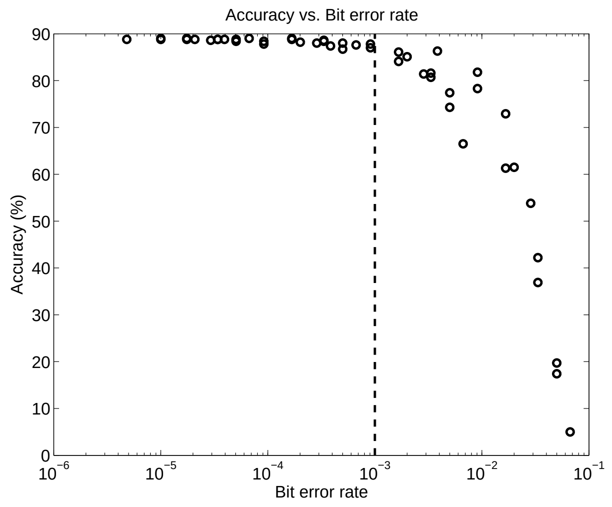


Fig. 11. Bit error rate vs. speech recognition accuracy using the ETSI DSR standard and a 5,000 word speech recognition task.

The ETSI standard uses CRC error detection on consecutive frame pairs to determine if there is a bit error [10]. Errors in the quantized speech vectors are concealed simply by repeating previous or subsequent speech vectors to fill in the gap. Using this error concealment scheme, we simulated bursty error channel via a two-state Gilbert-Elliot [45] channel model for a variety of different channel conditions. In Figure 11, we calculated the average bit error probability for each Gilbert-Elliot channel and plotted the BER vs. accuracy on a 5,000-word vocabulary Wall Street Journal speech recognition task. The accuracy of the system without bit errors was 88.8%. We can see that the system starts to lose accuracy significantly after an average bit error probability of 10^{-3}. Therefore, we use 10^{-3} as our target bit error rate.

V. Summary of DSR Tradeoffs

By using the client-side ASR energy model and the DSR energy model for both Bluetooth and 802.11b wireless networks, we can examine the energy tradeoffs with respect to channel quality, delay, and speech recognition accuracy. Higher bit rates have small increases in system level energy consumption due to the overhead of the power saving algorithms on the wireless device. This tradeoff is shown in Table VI. For the remainder of this analysis, we consider transmission at the highest available bit rate, which offers the best WER. In Figure 12, we plot the energy consumption per frame of speech for client-side ASR and DSR under both 802.11b and Bluetooth wireless networks with respect to channel quality. For DSR, we include the both the communication and computation (feature extraction/quantization) energy costs. For 802.11b, we consider the energy consumption of the power on/off scheduling algorithm with a latency of 240ms, 480ms, and 2 seconds and unlimited ARQ retransmissions. For the Bluetooth interface we show the energy consumption for both medium and high rate data packets as well as the three types of voice packets with latency of 480ms. To the right of the Y-axis we have the approximate energy savings over client-side ASR operating 2.5 times slower than real-time. We can expect a scaled down speech recognition task (i.e. simpler acoustic and language models or smaller vocabulary) running at real-time to give 60% energy savings. However, this will come at a cost of reduced functionality for the user, perhaps going from a dictation system to a command and control system with smaller vocabulary. We have not quantified the cost of reduced utility for the user in this work. However, for the various DSR scenarios in Figure 12 we assume little to no reduction in quality for the end-user by maintaining sufficient data integrity through source coding techniques and/or ARQ retransmissions. Table VII shows the percentages of computation and communication energy for a few different configurations as well as the expected battery lifetime with a 1400mAh/3.6V lithium-ion cell. The 802.11b interface with long delays gives the lowest overal energy consumption and an almost even division between energy spent in computation and communication. DSR with Bluetooth uses a higher percentage of communication energy, and this amount does not decrease significantly with increased delay due to the overhead of the park mode. Expected battery lifetimes exceed that of typical cellular telephones as we do not require real-time communication. Even modest delays of less than 0.5s can yield significant battery lifetime with constant streaming of DSR data.

TABLE VI
Total energy consumption for both computation and communication vs. bit rate for Bluetooth and 802.11b. ($T = 0.48s$).

		Computation + Communication	
Bit rate (kbps)	WER (%)	Bluetooth (mJ)	802.11b (mJ)
1.2	16.79	1.1279	2.4661
1.4	11.71	1.1315	2.4688
1.6	9.3	1.1323	2.4698
1.8	8.1	1.1338	2.4717
1.9	6.99	1.1358	2.4719
2.0	6.63	1.1380	2.4749
4.2	6.55	1.1701	2.5044

In a good channel with high SNR, Bluetooth allows systemwide energy savings of over 95% compared with full client-side ASR. DH5 packets offer the lowest overhead and best energy savings, while DM1 packets offer the most robust operation down to around 10 dB with some minimal energy cost. The ARQ retransmission protocol causes rapid increases in energy consumption after some SNR threshold is reached. It is possible to operate in lower SNR through packet fragmentation, which will lower the probability of a packet being received in error. This is evident in Figure 12 by comparing DH1 and DH5 data packets. The longer packet length in DH5 packets causes a sharp increase in retransmits and energy consumption at around 25 dB, whereas DH1 packets can operate down 15 dB before the number of retransmits becomes excessive. In addition, FEC bits can be used to lower the probability of a packet retransmit. The Hamming code in DM1 and DM5 packets allows operation down to around 10 and 16 dB respectively. Finally, Bluetooth voice packets have energy consumption that is independent of SNR since no ARQ protocol is used. Uncoded HV3 packets have the lowest overhead, and therefore the lowest energy consumption per frame

TABLE VII
SUMMARY OF ENERGY CONSUMPTION FOR ASR AND DSR WITH HIGH CHANNEL SNR.

Type	Computation (%)	Communication (%)	Total per Speech Frame (mJ)	Battery Lifetime (h)
DSR w/Bluetooth (T=0.48s)	32%	68%	1.17	43.1
DSR w/802.11b (T=0.48s)	15%	85%	2.5	20.2
DSR w/802.11b (T=2s)	42%	58%	0.92	54.8
Local ASR (R=2.5)	100%	0%	45	1.12

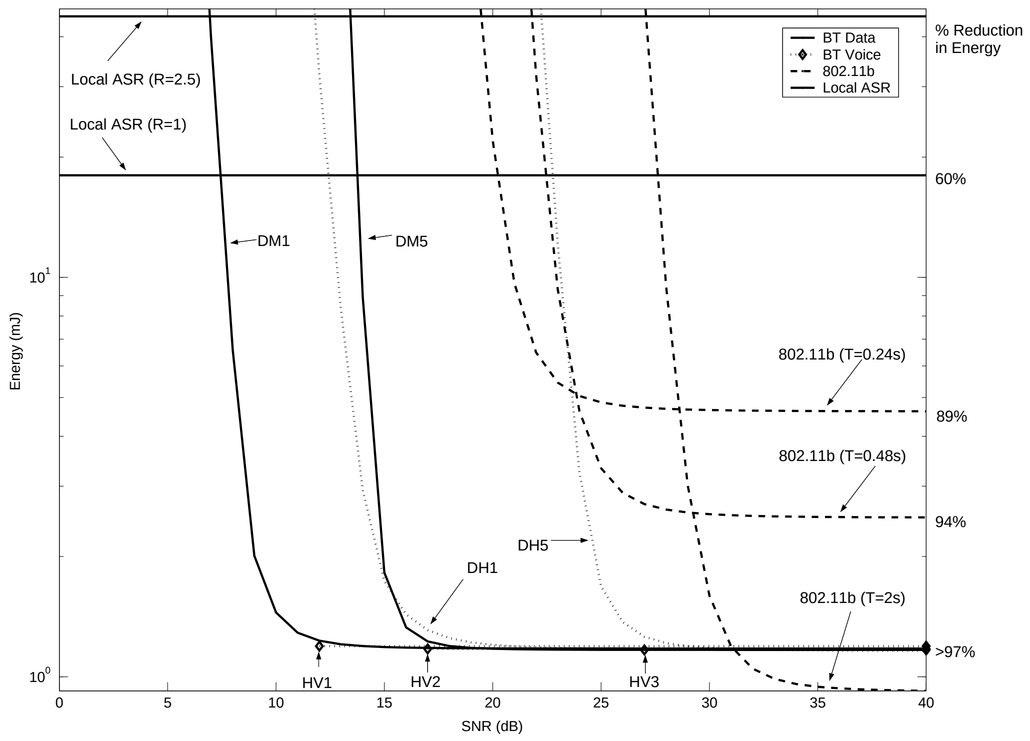


Fig. 12. The energy consumption of client-side ASR and DSR under Bluetooth and 802.11b vs. SNR.

of speech, but they only operate down to around 27 dB. Beyond that, the probability of a bit error exceeds 10^{-3}, which we have determined to have a noticable impact on speech recognition accuracy. HV1 and HV2 packets can operate down to around 12 and 17 dB respectively.

Finally, 802.11b networks allow systemwide energy savings of approximately 89-94% with relatively small values of T. With larger values of T, such as one second or more, we can use less energy than Bluetooth. However, due to the larger packet overhead, larger maximum packet sizes, different modulation, techniques, and lack of error-correcting codes, the 802.11b network does not operate as well in lower SNR ranges. Packet fragmentation or a switch to a more robust modulation technique with lower maximum bit rate can extend the lower SNR range at the cost of increased energy consumption, but we have not considered these effects here. However, 802.11b does offer increased range and may be more appropriate in certain scenarios.

VI. CONCLUSION

In this paper, we investigated the energy consumption of a distributed speech recognition front-end. We considered energy usage from both computation and communication. The advantages of DSR from an energy consumption perspective are clear. Client-side speech recognition in software can consume several orders of magnitude more energy than a DSR system. However, the use of low-power ASIC chips for speech recognition may help reduce the energy consumption of client-side ASR in the future.

The computation of a speech recognition front-end can be optimized for a particular processor to reduce the energy consumption. Savings of more than 80% can be obtained through algorithmic and architectural optimizations. Dynamic voltage scaling can be applied at run-time to minimize the energy consumption even further.

In our analysis of DSR, we have considered both

802.11b and Bluetooth wireless networks. Given the relatively high bit rates these standards provide with respect to DSR traffic, we investigated the use of synchronous bursty transmission of the data to maximize the amount of time spent in a low-power or off state. While this adds a small delay to the end-user, the energy savings can be significant. With 802.11b, we can reduce the energy consumption of the wireless interface by around 80% with modest application delays of just under half a second. Bluetooth offers lower energy consumption for smaller values of delay, T, but as delay increases, the Bluetooth energy consumption is dominated by the time spent in park mode. The 802.11b interface with on/off scheduling can operate with a lower energy consumption than Bluetooth when T exceeds 1.3 seconds.

In the presence of bit errors, we can estimate the energy consumption with respect to SNR and identify the appropriate operating ranges for the various packet types. For voice packets, we recommend a minimum bit error rate after FEC of 0.1%. Given that one packet many contain many frames of speech, packet losses can have significant impact on accuracy. Bluetooth voice packets can operate down to 12 dB SNR without any perceptible reduction in accuracy. The use of FEC codes can allow Bluetooth data packets to operate down to slightly less than 10 dB and still use less energy than client-side ASR.

ACKNOWLEDGEMENT

The authors would like to thank John Anckorn and Wajahat Qadeer for their contribution of Bluetooth power consumption measurements. In addition, we would like to thank Mat Hans and Mark Smith of HP Labs for their continued support of this work.

REFERENCES

[1] G. Q. Maguire, M. Smith, and H. W. P. Beadle, “Smartbadges: A wearable computer and communication system,” 6th International Workshop on Hardware/Software Codesign, 1998, invited Talk.

[2] L. Karray, A. B. Jelloun, and C. Mokbel, “Solutions for robust recognition over the gsm cellular network,” in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing,* vol. 1, 1998, pp. 261–264.

[3] B. Lilly and K. Paliwal, “Effect of speech coders on speech recognition performance,” in *ICLSP 96*, vol. 4, 1996, pp. 2344–2347.

[4] H. K. Kim and R. V. Cox, “Bitstream–based feature extraction for wireless speech recognition,” in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP),* vol. 3, 2000, pp. 1607–1610.

[5] X. Zhong, J. Arrowood, and M. Clements, “Speech coding and transmission for improved automatic recognition,” in *Int. Conf. Spoken Language Processing*, 2002.

[6] G. N. Ramaswamy and P. S. Gopalakrishnan, “Compression of acoustic features for speech recognition in network environments,” in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP),* vol. 2, 1998, pp. 977–980.

[7] N. Srinivasamurthy, A. Ortega, Q. Zhu, and A. Alwan, “Towards efficient and scalable speech compression schemes for robust speech recognition applications,” *IEEE International Confernce on Multimedia and Expo*, vol. 1, pp. 249–252, 2000.

[8] V. Digilakis, L. Neumeyer, and M. Perakakis, “Quantization of cepstral parameters for speech recognition over the world wide web,” *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 82–90, 1999.

[9] C. Boulis, M. Ostendorf, E. A. Riskin, and S. Otterson, “Graceful degradation of speech recognition over packet erasure networks,” *IEEE Transactions on Speech and Audio Processing*, vol. 10, pp. 580–590, 2002.

[10] “Speech processing, transmission and quality aspects (stq); distributed speech recognition; front-end feature extraction algorithm; compression algorithms,” ETSI Standard: ETSI ES 201 108 v1.1.2, 2000, http://www.etsi.org.

[11] E. Cornu and H. Sheikhzadeh, “A low-resource, miniature implementation of the etsi distributed speech recognition front-end,” in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, 2002.

[12] C. Chiasserini, P. Nuggehalli, and V. Srinivasan, “Energy-efficient communication protocols,” in *DAC*, 2002.

[13] E. Shih, P. Bahl, and M. Sinclair, “Dynamic power management for non-stationary service requests,” in *MOBICOM*, 2002.

[14] P. Lettieri, C. Schurgers, and M. Srivastava, “Adaptive link layer strategies for energy efficient wireless networking,” *Wireless Networks*, vol. 5, pp. 339–355, 1999.

[15] C. Jones, K. Sivalingam, P. Agrawal, and J. Chen, “A survey of energy efficient network protocols for wireless networks,” in *DATE*, 1999, pp. 77–81.

[16] R. Krashinsky and H. Balakrishnan, “Minimizing energy for wireless web access with bounded slowdown,” in *MOBICOM*, 2002.

[17] K. Sivalingam, J. Chen, P. Agrawal, and M. Srivastava, “Design and analysis of low-power access protocols for wireless and mobile atm networks,” *Wireless Networks*, vol. 6, pp. 73–87, 2000.

[18] V. Raghunathan, S. Ganeriwal, C.Schurgers, and M. Srivastava, “e^2wfq: An energy effiecient fair scheduling policy for wireless systems,” in *ISLPED*, 2002.

[19] A. Acquaviva, T. Simunic, V. Deolalikar, and S. Roy, “Remote power control of wireless network interfaces,” *Lecture Notes in Computer Science,* October 2003.

[20] T. Simunic, L. Benini, P. Glynn, and G. D. Micheli, “Event-driven power management,” *IEEE Transactions on CAD*, July 2001.

[21] A. Acquaviva, L. Benini, and B. Riccó, “Software controlled processor speed setting for low-power streaming multimedia,” *IEEE Transactions on CAD*, pp. 1283–1292, November 2001.

[22] R. Kravets and P. Krishnan, “Application-driven power management for mobile communication,” *Wireless Networks*, vol. 6, no. 4, pp. 263–277, 2000.

[23] R. Min and A. Chandrakasan, “A framework for energy-scalable communication in high-density wireless networks,” in *The 2002 International Symposium on Low Power Electronics and Design*, 2002.

[24] R. Min, “Energy and quality scalable wireless communicaiton,” Ph.D. dissertation, M.I.T., June 2003.

[25] E. Takahashi, “Application aware scheduling for power management on ieee 802.11,” in *IPCCC*, 2000.

[26] C. Luna, Y. Eisenberg, T. Pappas, R. Berry, and A. Katsaggelos, “Transmission energy minimization in wireless video streaming applications,” 2001.

[27] P. Shenoy and P. Radkov, “Proxy-assisted power-friendly streaming to mobile devices,” in *MMCN*, 2003.

[28] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, “Physical layer driven protocol and algorithm design for energy efficient wireless sensor networks,” in *SIGMOBILE*, 2001.

[29] J. Lorch and A. J. Smith, “Software strategies for portable computer energy management,” *IEEE Personal Communications*, pp. 60–73, June 1998.

[30] F. Bellosa, “Endurix: Os-direct throttling of processor activity for dynamic power management,” University of Erlangen, Tech. Rep. TR-I4-99-03, June 1999.

[31] J. Flinn and M. Satyanarayanan, “Energy-aware adaptation for mobile applications,” in *SOSP*, December 1999.

[32] Y. Lu, L. Benini, and G. D. Micheli, “Operating system directed power reduction,” in *ISLPED*, July 2000, pp. 37–42.

[33] W. H. et. al., “Itsy: Stretching the bounds of mobile computing,” *Computer*, vol. 34, pp. 28–36, April 2001.

[34] Deller, Proakis, and Hansen, *Discrete–Time Processing of Speech Signals*. Upper Saddle River, NJ: Prentice Hall, 1987.

[35] T. Simunic, L. Benini, and G. D. Micheli, “Energy-efficient design of battery-powered embedded systems,” *Special Issue of IEEE TVLSI*, pp. 18–28, May 2001.

[36] Various, “C source code optimizations for arm,” Application Note 33, 1996, aRM Inc.

[37] M. E. Frerking, *Digital Signal Processing in Communications Systems*. Van Nostrand Reinhold, 1994.

[38] J. W. Crenshaw, *Math Toolkit for Real-Time Programming*. Lawrence, Kansas: CMP Books, 2000.

[39] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing, Second Edition*. Cambridge England: Cambridge University Press, 1992.

[40] J. G. Proakis, *Digital Communications*, 3rd ed. McGraw-Hill, 1995.

[41] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Simon and Schuster, 1995.

[42] J.-P. Ebert, S. Aier, G. Kofahl, A. Becker, B. Burns, and A. Wolisz, “Measurement and simulation of the energy consumption of a wlan interface,” Technical University of Berlin, Telecommunication Networks Group, Tech. Rep. TKN-02-010, June 2002.

[43] “Bluetooth specification (v1.1),” [http://www.bluetooth.com], 2002.

[44] M. Valenti, M. Robert, and J. Reed, “On the throughput of bluetooth data transmissions,” in *IEEE Wireless Communications and Networking Conference*, vol. 1, 2002, pp. 119– 123.

[45] E. N. Gilbert, “Capacity of a burst noise channel,” *Bell Syst. Tech. Journal*, pp. 1253–1265, 1960.