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 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.**

I. Introduction

Wireless mobile devices such as PDAs and cellular phones typically contain small screens and tiny keypads. Appropriate use of speech recognition technology can allow users to interact with the system in a more natural manner. While many cellular phones currently have voice dialing capability, more sophisticated automatic speech recognition (ASR) tasks require computation capability beyond what these devices currently provide. Applications of ASR for embedded devices may include e-mail dictation, Web browsing, and scheduling and contact management applications, navigation, and command and control.

Mobile devices are limited in computation, memory, and battery energy, therefore complex ASR tasks are difficult to perform on the device due to these resource limitations. A typical ASR 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 ASR across the network is an attractive alternative for these mobile wireless devices. In the absence of a network connection, some limited ASR 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. This client-server approach to ASR has been well studied in the literature. The back-end ASR search including hidden Markov model (HMM) state output evaluation and Viterbi search is performed at the server. This further limits the problem to single hop communication with a mobile host and base station infrastructure. A true distribution of the workload across many wireless nodes of equal processing capability would probably cause too much wireless traffic to be beneficial. 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.

One challenge in designing an ASR 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 of a DSR 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. We compare the energy consumption of both client-side ASR and DSR using two different network interfaces.

The embedded system used in the experiments is the SmartBadge IV 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 ASR 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. It has high-quality audio input suitable for ASR. The StrongARM platform is still used in many high-end PDAs on the market today, such as the HP iPAQ H3800. 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.

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 DSR considered the effects of communication over cellular networks. The effects of using coded speech in ASR is presented in [2]. Low bit rate speech coders, such as those used in cellular telephony, 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. 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. 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 and those that assume to communicate only with an ASR system. We consider the latter, where only the spectral information needs to be included, which can result in better performance with lower bit rates. Vector quantization is the dominant compression technique with bit rates in the low kbps range [3]. The ETSI Aurora DSR standard includes a vector quantizer with some error detection, concealment, and framing techniques [4].

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. The results include energy efficient channel coding, scheduling at the physical layer, new 802.11b protocol proposals and packet scheduling strategies. A server-driven scheduling methodology aimed at reducing power consumption for streaming MPEG4 video is introduced in [5]. 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. Energy-performance tradeoffs based on application needs have been recently addressed, including the energy-QoS tradeoff and cooperation between multimedia applications and the operating system. A more complete list of references is provided in [6].

III. Modeling the Energy Used in Computation

The computation of speech features is a small portion of the overall ASR task in both computation and memory usage. Client-side ASR requires more computation and memory bandwidth due to the back-end search algorithm. Table I shows the average cycle count to process one frame of speech in the Sphinx-III large vocabulary ASR system on an Intel Pentium 4 workstation. The total processing for the front-end is less than 1% of the overall computation, with the majority of time being spent in the hidden Markov modeling step. Porting a full ASR 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 [7].

TABLE I
Cycle counts for the front-end, Gaussian evaluation, and Viterbi search portions of ASR.

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 are mel-frequency cepstral coefficients [8]. 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 applied to speech frames every 10ms. Implementing the front-end feature extraction for a DSR system on an embedded platform requires not only speed, but also power optimization, since the battery lifetime 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. Measurements presented in [9] show that the improvements that can be gained using standard compiler optimizations are marginal compared with 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 reduced energy 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. Due to a lack of floating-point hardware, simulations showed that the StrongARM spent over 90% of its time in floating-point emulation. Any further gains require fixed-point arithmetic.

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. The remaining optimizations involve a fast complex magnitude calculation, dynamic range reduction for the filterbank operation, and a fast logarithm based on bit manipulation. The mathematical details of these optimizations can be found in [6].

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. Some further processing of the output is required to get the desired result, but this overhead is minimal compared with the reduction in computation. Additional savings can be obtained though the use of look-up tables for frequently used mathematical functions.

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. 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 11 different frequencies. 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 StrongARM needs in order to run correctly but with lower energy consumption. We obtained real-time performance at all possible frequency and voltage settings.

4) Software Optimization Results: Simulation results for processing one frame (25ms) of speech on the SmartBadge IV architecture running at 202.4 MHz are shown in Figure 1. 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 general C source optimizations [10]. 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 μJoules). The simulation results are computed by a cycle-accurate energy simulator, and include processor core and L1 cache energy, interconnect and pin energy, energy used by the memory, losses from the DC/DC converter, and battery inefficiency [9]. We have achieved a reduction in the total battery energy required to process one frame of speech data by 83.5%. When using this optimized ASR front-end in both the test and training phases, we observe no significant reduction in ASR accuracy on a connected digit recognition task.

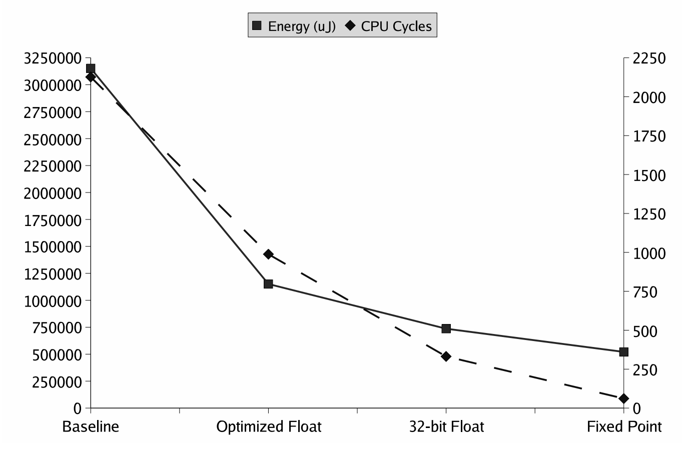

Fig. 1. Cycle count, left, and energy consumption in μJ, right, per frame of speech.

Because the fixed-point version runs 34 times faster than the original baseline source code, it is possible to get further reductions in energy usage by using DVS. Power measurements are performed on the SmartBadge IV system running the Linux O/S and our optimized front-end. At the lowest combined frequency and voltage settings, 59 MHz and 0.78 V, the algorithm still runs in real-time, and the system uses 34.7% less power than at 206 MHz.

5) Vector Quantization: Finally, we include the fixed-point vector quantization code in our profiling and consider different bit rates and quantization levels. For our system, we use a split vector quantization scheme presented in [3] with bit rates ranging from 1.2 to 2.0 kbps. We include an additional bit allocation that is similar to the ETSI DSR standard that will operate at 4.2kbps [4]. The actual bit rate needed for an ASR 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 [3], the range of bit rates was evaluated for a small vocabulary task under ideal acoustic conditions. We can expect the WER to increase under less ideal conditions (i.e. larger vocabulary, more acoustic background noise, etc.). Table II shows the resulting bit rates and word error rates from [3].

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 ASR accuracy. Both hardware measurements and simulation reveal that 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 ASR engines have been optimized for the StrongARM or other mobile processors by several companies, but it has been shown that they use almost all 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 [7]. The ASR system ran just over 2.5 times real-time, and the processor was 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 ASR 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 ASR task, we can estimate the energy consumption to recognize one frame of speech as:

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

Therefore, for an ASR 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. 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

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.

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.

In order to estimate the power consumption for wireless transmission, we directly measure the average current into the network interface. 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 ASR task with respect to channel conditions. A more rigorous analysis of the energy models discussed in this section can be found in [6].

A. 802.11b Wireless Networks

The 802.11b interface operates at a maximum bit rate of 11 Mbps with a range of 100 meters. It uses an automatic repeat request (ARQ) protocol with CRC error detection to maintain data integrity. We used a PCMCIA 802.11b interface card and measured the average current going into the card to get the power dissipation. Our measurements indicate there is a difference of only 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. 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 power management (PM) mode can provide some savings in energy consumption but it was shown in [5] that this does not hold under heavy broadcast traffic conditions, defined as a higher than average amount of broadcast packets. While operating in the 802.11b PM 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 interval. This increase in power consumption will happen even if there are no applications running on the mobile host. Measurements in [5] indicate that, even in less than average amounts of broadcast traffic, significant energy is wasted by the extra processing.

Since the energy consumption of PM mode on 802.11b networks breaks down in heavy broadcast traffic conditions, we consider an alternate algorithm. If we are interested in transmitting only ASR-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.

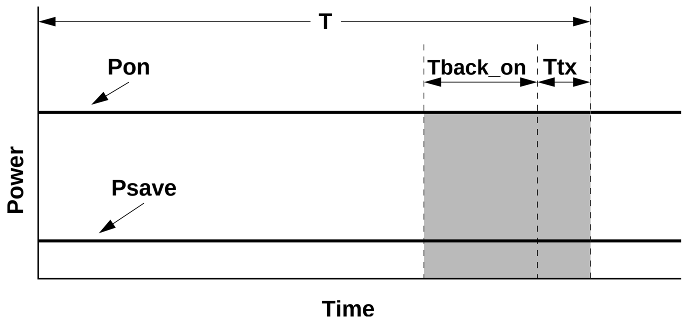

Fig. 2. The timing of the 802.11b scheduling algorithm.

Figure 2 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, T_{back_on}, at the expense of longer delay. Assuming that a speech recognition server is able to process speech at or near real-time, the user will experience delay near the value of T. 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 about one second may hardly be noticed by the user, whereas delays of about three seconds or more may hinder interactivity. For a dictation application, such as e-mail, this delay is less important.

The energy consumption for the always on, E_{on}, and WLAN PM mode, E_{save}, can be estimated as the product of the measured average power dissipation and the cycle time T. Under the proposed scheduling algorithm, the WLAN card will be on only during the shaded region in Figure 2.

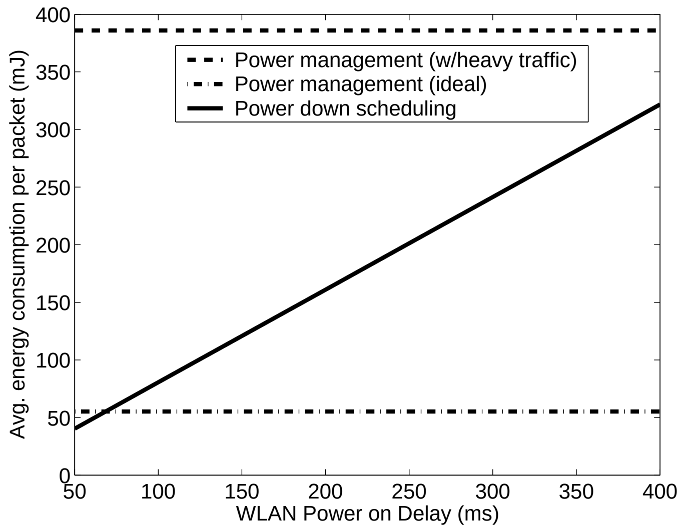

Fig. 3. WaveLAN power on delay vs. energy consumption per packet.

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 3 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. The mobile device will have to monitor the broadcast traffic and decide between the standard 802.11b PM mode or the scheduling algorithm.

Finally, we consider increased delay or latency, T, in Figure 4 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

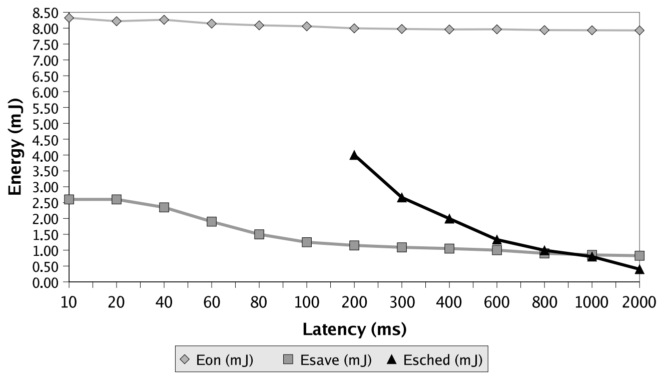

Fig. 4. Average energy consumption per 10ms speech frame vs. DSR latency for various 802.11b power save schemes. (WLAN power on delay is fixed at 100ms.)

scheduling algorithm (E_{sched}) will be able to outperform the PM mode configuration (E_{save}) regardless of traffic conditions. 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 signal to noise ratio (SNR). After the reception of a good packet, an ACK is sent across a robust control channel. If the ACK is not received by the sender, then the packet is retransmitted. The expected number of retransmits for a given bit error rate and packet length can be calculated and used to estimate the energy consumption [11]. The bit error rate is estimated using an expression for 256-QAM modulation in a Rayleigh fading channel. The Rayleigh fading channel asssumption is widely used in wireless communications literature as a more realistic alternative to an additive white Gaussian noise channel [12]. By applying our BER expression and power measurements to the expression for energy per goodput in [11], we can find an expression for the expected energy consumption in a noisy channel.

B. Bluetooth Personal Area Network

The Bluetooth personal area network provides a maximum bit rate of 1 Mbps, and a variety of packet types are available to support different traffic requirements [13]. 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. Media access is handled via a time-division duplex (TDD) scheme in which each time slot lasts 625 μseconds. Data packets are available in both high-rate (DH) and medium-rate (DM) packets that occupy either 1, 3, or 5 time slots. Medium rate packets contain a 2/3 rate error correction 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, in which the number denotes the amount of error correction rather than slot length. 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 use either data packets or voice packets without consideration of packet jitter or delay characteristics.

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. 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, so the energy can again by estimated by the power-time product.

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 [13]. We consider only the *park* mode, where 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.

By varying the amount of data transmitted at once, we can increase the amount of time spent in the park state. Our models show that 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.

We can perform a similar analysis as in Section IV-A to estimate the energy consumption of Bluetooth data packets in the presence of bit errors. During periods of bit errors, data packets will continue to be retransmitted until they are received correctly or a timeout occurs. We assume BFSK modulation with coherent detection under a Rayleigh fading channel [12] and estimate the probability of packet failure [14] as well as the expected number of retransmits to get the final expression for energy consumption. Voice packets are delivered even in error, so the energy consumption can be estimated as a simple power-time product using measured values of power dissipation.

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 ASR 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 II. For the remainder of this analysis, we consider transmission at the highest available bit rate, which offers the best WER. In Figure 5, 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 show the approximate energy savings over client-side ASR operating 2.5 times slower than real-time. We can expect a scaled down ASR 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 to a more constrained vocabulary and speaking style. For the various DSR scenarios in Figure 5 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 III 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 overall 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.

TABLE II
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 about 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 5 by comparing DH1 and DH5 data packets. The longer packet length in DH5 packets causes a sharp increase in retransmits and energy consumption at about 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 about 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 of speech, but they only operate down to about 27 dB. Beyond that, the probability of a bit error exceeds 10^{-3}, which we have determined to have a noticable impact on ASR accuracy [6]. HV1 and HV2 packets can operate down to about 12 and 17 dB respectively, with little noticeable loss in recognition accuracy.

Finally, 802.11b networks allow system wide 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 payload 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 DSR front-end. We considered energy usage from both computation and communication. The advantages of DSR from an energy consumption perspective are clear. Client-side ASR in software can consume several orders of magnitude more energy than a DSR system. However, the use of low-power ASIC chips for ASR may help reduce the energy consumption of client-side ASR in the future.

The computation of an ASR 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 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 about 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. 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

TABLE III

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

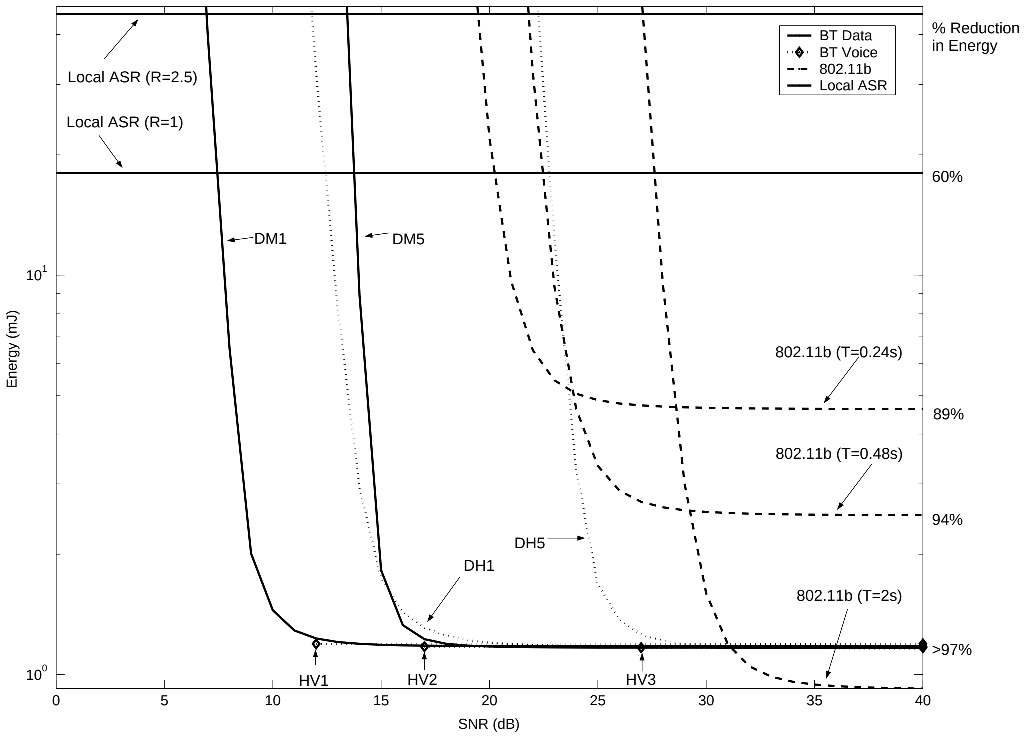

Fig. 5. The energy consumption of client-side ASR and DSR under Bluetooth and 802.11b vs. SNR.

consumption measurements as well as 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] B. Lilly and K. Paliwal, "Effect of speech coders on speech recognition performance," in *ICLSP 96*, vol. 4, 1996, pp. 2344–2347.

[3] 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.

[4] "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.

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

[6] B. Delaney, T. Simunic, and N. Jayant, "Energy aware distributed speech recognition for wireless mobile devices," Hewlett Packard Laboratories, Tech. Rep.

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

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

[9] 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.

[10] Various, "C source code optimizations for arm," Application Note 33, 1996, aRM Inc.

[11] 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.

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

[13] "Bluetooth specification (v1.1)," [http://www.bluetooth.com], 2002.

[14] 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.