Energy-Aware Distributed Speech Recognition for Wireless Mobile Devices

Brian Delaney
MIT Lincoln Laboratory

Nikil Jayant
Georgia Institute of Technology

Tajana Simunic
University of California, San Diego

Editors' note:
Severe limitations in computational, memory, and energy resources make implementing high-quality speech recognition in embedded devices a difficult challenge. In this article, the authors investigate the energy consumption of computation and communication in an embedded distributed speech recognition system and propose optimizations that reduce overall energy consumption while maintaining adequate quality of service for the end user.
—Radu Marculescu, Carnegie Mellon University; and Petru Eles, Linköping University

USERS OF WIRELESS mobile devices such as PDAs and cell phones typically interact with the system through small screens and tiny keypads. The appropriate use of speech recognition technology could enable users to interact with the system in a more natural manner. Although some mobile clients already have automatic speech recognition (ASR) capability, more complex tasks require significant use of system resources, including battery energy. Potential ASR applications for embedded devices include e-mail dictation, Web browsing, scheduling and contact management, navigation, and command and control.

These complex ASR tasks are difficult to perform on mobile devices, which have limited computational power, memory, and battery energy. A typical ASR system includes 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. Each acoustic or language model typically uses some tens of megabytes of storage, with significant computation required for large vocabulary searches. Therefore, distributing the ASR across the network is an attractive alternative for mobile wireless devices. In the absence of a network connection, the devices can perform some limited ASR.

In distributed speech recognition (DSR), the mobile client calculates speech features—typically mel frequency cepstral coefficients (MFCCs)—and sends them over the wireless network to a server. The literature has widely studied this client-server ASR approach. The back-end ASR search, including hidden Markov model (HMM) state output evaluation and Viterbi search, occurs at the server. This limits the problem to single-hop communication using 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. To minimize the bit rate, the client first compresses the MFCCs using a quantization scheme. DSR requires a three-step process of computation, quantization, and communication on the mobile client.

One challenge in designing an ASR system for a mobile device is minimizing the total energy consumption used in the speech recognition task. Indiscriminate use of the CPU, the memory, and the wireless network can cause considerable battery drain. Here, we examine a DSR system's energy use with respect to the quality-of-service metrics pertinent to this application. We

0740-7475/05/$20.00 © 2005 IEEE Copublished by the IEEE CS and the IEEE CASS

consider both communication- and computation-related energy drain and propose techniques for minimizing energy use in both areas while maintaining a useful level of service for the user. We compare the energy consumption of client-side ASR and DSR using two different network interfaces.

The embedded system we used in our experiments is the SmartBadge IV, developed at Hewlett-Packard's Mobile and Media Systems Lab (http://www.it.kth.se/~maguire/badge4.html). The system contains a 206-MHz StrongARM-1110 processor, a StrongARM-1111 coprocessor, flash memory, SRAM, a Personal Computer Memory Card International Association (PCMCIA) interface, and various sensor inputs such as audio, temperature, and accelerometers. It runs the Linux operating system.

The SmartBadge IV has speech- or audio-driven I/O, so ASR can provide some level of user interaction through a voice-user interface. The high-quality audio input is suitable for ASR. The system supports various networking hardware options, including Bluetooth and 802.11b wireless interfaces. Many high-end PDAs on the market today, such as the HP iPAQ H3800, use the StrongARM platform. 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's a better choice for investigating the issues discussed here. Newer PDAs based on XScale processors have an architecture similar to the StrongARM architecture, and we expect similar results with these processors.

Related work

Earlier work on DSR considered the effects of communication over cellular networks. For example, Lilly and Paliwal presented the effects of using coded speech in ASR.[1] The authors found that 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, Kim and Cox calculated cepstral coefficients directly from the wireless bitstream.[2] Although this technique offered some improvement, its fundamental limitation is that traditional speech-coding techniques target human, not machine, listeners. The spectral distortion introduced by speech coding is designed to have minimal impact on human listeners, but speech recognizers rely solely on this spectral information. Thus, currently deployed low-bit-rate speech-coding techniques are not suitable for high-quality ASR applications.

More recent DSR work takes two main approaches: attempting to design ASR-friendly speech coders and communicating only with an ASR system. We consider the latter approach, in which only the spectral information must be included, resulting in better performance with lower bit rates. Vector quantization is the dominant compression technique with bit rates in the low kilobits-per-second range.[3] The European Telecommunications Standards Institute (ETSI) Aurora DSR standard includes a vector quantizer along with some error-detection, concealment, and framing techniques.[4]

This article considers the application of DSR traffic to both Bluetooth and 802.11b networks. Researchers have addressed the wireless network power optimization problem at different abstraction layers, starting from the semiconductor device level to the system and application levels. The results of this research include energy-efficient channel coding, scheduling at the physical layer, new 802.11b protocol proposals, and new packet-scheduling strategies. Acquaviva et al. introduced a server-driven scheduling methodology aimed at reducing power consumption for streaming MPEG-4 video.[5] Traditional system-level power management techniques fall into two categories: techniques for shutting down components and policies that dynamically scale down processing voltage and frequency. Researchers have recently addressed energy-performance trade-offs, including the trade-off between energy and quality of service; and application needs such as cooperation between multimedia applications and the operating system. Our technical report provides additional references to related work.[6]

Modeling computational energy use

Computing speech features is a small portion of the overall ASR task in both computation and memory use. Client-side ASR might be necessary to maintain interactivity with the mobile device when the network is not present. Client-side ASR requires more computation and memory bandwidth than does DSR because it uses a back-end search algorithm. Table 1 shows the average cycle count for processing one frame of speech in the Sphinx-III large-vocabulary ASR system on an Intel Pentium 4 workstation. Total processing for the front end is less than 1% of the overall computation, with most of the time spent in the HMM step. Porting a full ASR system 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 for reducing memory bandwidth. Therefore, we concentrate our

software optimization on the signal-processing front end only and use published results to estimate the complete client-side ASR energy consumption.[7]

Here we compare the results in Table 1 with those of DSR under various channel conditions, error correction methods, and packet sizes to show DSR's energy consumption benefits. Through algorithmic and architectural optimization in software, we reduce the energy consumption of the signal-processing front end by 83%. Runtime dynamic voltage scaling (DVS) techniques can enhance these savings.

Table 1. Cycle counts for front-end, Gaussian evaluation, and Viterbi search portions of automatic speech recognition (ASR).

Module	Average cycles per frame	Percentage 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

Signal-processing front end

The acoustic observations generated by the signal-processing front end are MFCCs.[8] The calculation of MFCCs requires filtering and windowing operations, a magnitude fast Fourier transform (FFT) calculation, a filter bank operation, a logarithm operation, and a discrete cosine transform applied to speech frames every 10 ms. Implementing front-end feature extraction for a DSR system on an embedded platform requires not only speed but also power optimization, because the battery lifetime in such devices is very limited. Here, we discuss both source code and runtime optimizations.

There are two types of source code optimization. The first, architectural optimization, attempts to reduce power consumption while increasing speed with methods targeted at a particular processor or platform. Measurements show that improvements gained from standard compiler optimization are marginal compared with writing energy-efficient source code.[9] The second category of source code optimization is more general and involves changes in the source code's algorithmic implementation with the goal of faster performance and reduced energy consumption.

The final optimization we discuss here is dynamic voltage scaling, which is the most general because you apply it at runtime without making any changes to the source code. DVS algorithms reduce energy consumption by changing processor speed and voltage at runtime, depending on the needs of the applications running. The maximum power savings obtained with DVS are proportional to the frequency savings and the square of the voltage.

Architectural optimization. Signal-processing algorithms, such as the MFCC, are generally mathematically intensive; therefore, we spent a significant amount of effort in optimizing the arithmetic. In addition, we used simple C code optimizations to help the compiler generate more efficient code. Because the StrongARM lacks floating-point hardware, simulations showed that it spent over 90% of its time in floating-point emulation. Any further gains require fixed-point arithmetic.

Implementing a pre-emphasis filter and a Hamming window using fixed-point arithmetic is straightforward. Fixed-point FFTs are common in DSP chips. The remaining optimizations involve a fast, complex magnitude calculation, a dynamic range reduction for the filter bank operation, and a fast logarithm based on bit manipulation. Our technical report provides the mathematical details of these optimizations.[6]

Algorithmic optimization. Profiling the original source code under a StrongARM simulator revealed that most of the execution time was spent in computing the FFT (which is an implementation of a discrete Fourier transform). Because speech is a real-valued signal, we can reduce an N-point complex FFT to an $N/2$-point real FFT. Further processing of the output is needed to get the desired result, but this overhead is minimal compared with the reduction in computation. Lookup tables for frequently used mathematical functions provide additional savings.

Dynamic voltage scaling. Once the code is optimized for both power consumption and speed, we can obtain further savings by changing the processing frequency and voltage at runtime. Users can configure the StrongARM processor on the SmartBadge IV at runtime with 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 µs. Because the typical processing time for the front end is much longer than the transition time, we can change the CPU frequency without perceivable overhead. For each frequency, the StrongARM needs a minimum voltage to run correctly with lower energy consumption. We obtained real-time performance at all possible frequency and voltage settings.

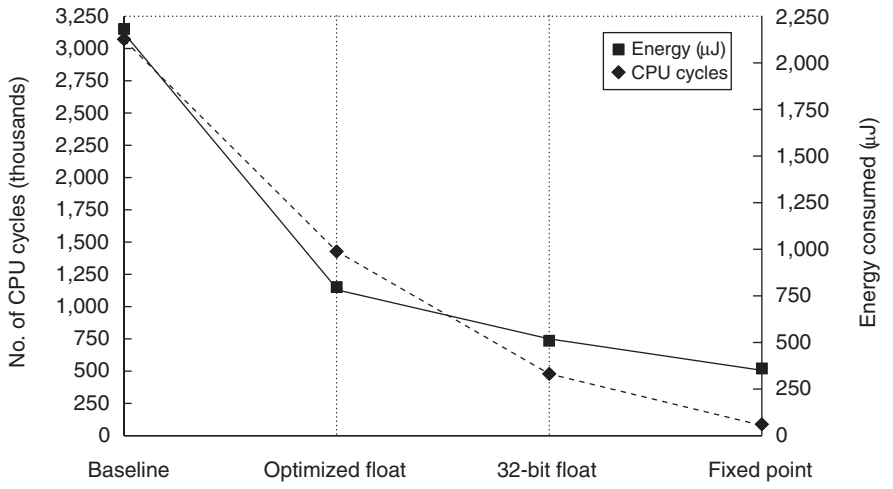

Figure 1. Cycle count and energy consumption per frame of speech.

Table 2. Total computation and communication energy consumption versus bit rate for Bluetooth and 802.11b, where T, the cycle time, is 0.48 seconds.

		Computation + communication	
Bit rate (Kbps)	**Word error rate (%)**	**Bluetooth (μJ)**	**802.11b (μJ)**
1.2	16.79	1,127.9	2,466.1
1.4	11.71	1,131.5	2,468.8
1.6	9.3	1,132.3	2,469.8
1.8	8.1	1,133.8	2,471.7
1.9	6.99	1,135.8	2,471.9
2.0	6.63	1,138.0	2,474.9
4.2	6.55	1,170.1	2,504.4

Software optimization results. Figure 1 shows simulation results for processing one frame (25 ms) of speech on the SmartBadge IV architecture running at 202.4 MHz. The *x*-axis shows the various stages of source code optimization. The *baseline source code* contains no software optimizations. The *optimized-float code* contains the set of algorithmic optimizations described earlier, as well as some general C source optimizations. The *32-bit-float* stage changes double-precision floating-point numbers to single-precision 32-bit floats. Finally, the *fixed-point* implementation contains all the source code optimizations described here. For each code version, the figure shows performance (in CPU cycles) and total battery energy consumed (in μJoules).

We computed the simulation results with a cycle-accurate energy simulator, and included processor core and L1 cache energy, interconnect and pin energy, energy used by the memory, losses from the DC-to-DC converter, and battery inefficiency.[9] We reduced the total battery energy required to process one frame of speech data by 83.5%. When using the optimized ASR front end in both the test and training phases, we observed no significant reduction of ASR accuracy in a connected-digit recognition task.

Because the fixed-point version runs 34 times faster than the original baseline source code, using DVS can further reduce energy consumption. We performed power measurements on the SmartBadge IV system running the Linux operating system 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.

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 by Digilakis, Neumeyer, and Perakakis,[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 operates at 4.2 Kbps.[4] The actual bit rate needed for an ASR task depends on many factors, including acoustic and speaker conditions as well as vocabulary size and complexity of the acoustic models. Digilakis, Neumeyer, and Perakakis evaluated the range of bit rates for a small-vocabulary task under ideal acoustic conditions. Table 2 shows the bit rates and word error rates they obtained. We expect the word error rate (WER) to increase under less ideal conditions (larger vocabulary, more background noise, and so forth).

We wrote fixed-point source code to perform the MFCC data quantization on the StrongARM processor

and profiled the code with the energy consumption simulator. The total energy consumption required to calculate MFCCs for one frame of speech, including vector quantization at 4.2 Kbps, was approximately 380 μJ. Even at the highest bit rate, vector quantization was only 12% of the total energy budget. This suggests that speeding up the quantization process by using smaller code books would produce minimal reductions in energy consumption and have a much greater impact on ASR accuracy. Hardware measurements and simulations reveal about a 14% increase in CPU power consumption but a greater than 50% reduction in WER between the highest and lowest bit rates. Therefore, because the reduction in energy consumption is minimal, we advocate the use of higher, more robust bit rates.

Client-side ASR

In the absence of a network connection, it might be necessary to perform ASR on the mobile device. Several companies have optimized speaker-dependent ASR engines for the StrongARM or other mobile processors, but these engines use almost all available resources and run several times more slowly than real time for many tasks. Hamburgen et al. present power measurements for an embedded ASR dictation system running on a StrongARM-based processor.[7] The system ran just over 2.5 times real time, and the processor was never idle during the task.

For our purposes, it's sufficient to describe local ASR energy requirements as the product of the average power dissipation of the processor and the memory under load and the time required to perform the ASR task. For the SmartBadge IV, we measured the average CPU and memory power dissipation as $P_{CPU} = 694$ mW and $P_{mem} = 1{,}115$ mW under load. Given the ASR task's real-time factor R, we estimate the energy consumed in recognizing one frame of speech as

$$E_{local} = (P_{CPU} + P_{mem}) \times R \times 1/100$$

Therefore, for an ASR task that runs $R = 2.5$ times slower than real time, we can expect to use approximately 45,000 μJ of battery energy to process one frame. Using smaller vocabularies and simpler acoustic and language modeling techniques should lower the total runtime and energy consumption at the cost of reduced performance. A reduced ASR task running in real time on a SmartBadge IV would use approximately 18,000 μJ of energy per frame of speech, but the trade-off is reduced utility for the end user.

Modeling communication energy use

Measurements taken on the SmartBadge IV hardware show that an 802.11b interface card can use up to 45% of the total power budget. Reducing this energy consumption is an important consideration. We analyze energy use in both 802.11b and Bluetooth wireless networks.

Because of the relatively low bit rates used in DSR, both networks operate well below their maximum throughput range. We can develop additional energy-saving opportunities by transmitting more data less often to exploit moderate increases in application latency. This allows the network interface to either power down or enter a low-power state between transmissions.

To estimate power consumption for wireless transmission, we directly measured the average current going into the network interface. Using this measurement as a baseline, we tailored a simple energy consumption model to investigate the effects of increased application latency. By buffering compressed speech features, we maximized the amount of time spent in the low-power or off state. We introduced a power-on or -off scheduling algorithm for the 802.11b device that exploits this increased latency. The medium access control (MAC) scheme of 802.11b and Bluetooth let us incorporate the effects of channel errors into the energy model. We used these results to investigate which techniques we should use to maintain a minimum quality of service for the ASR task with respect to channel conditions. We present a more detailed analysis of the energy models discussed here in our technical report.[6]

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 cyclic redundancy code (CRC) error detection to maintain data integrity. To measure power dissipation, we measured the average current entering the PCMCIA 802.11b interface card. Our measurements indicated a difference of only a few milliwatts in power consumption between the highest and lowest bit rates. We expected this because 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 adds significant overhead for small packet sizes. Because of the relatively high data rates provided by 802.11b, the wireless local area network (WLAN) interface spends most of its time waiting for the next packet to transmit.

The 802.11b power management (PM) mode provides some energy consumption savings, but not under heavy broadcast traffic conditions, defined as a higher

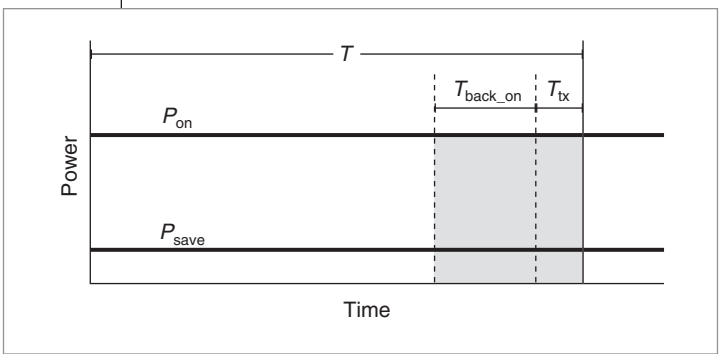

Figure 2. The 802.11b scheduling algorithm's timing. We denote power consumption as P_{on} for the always-on mode, and P_{save} for the WLAN PM mode.

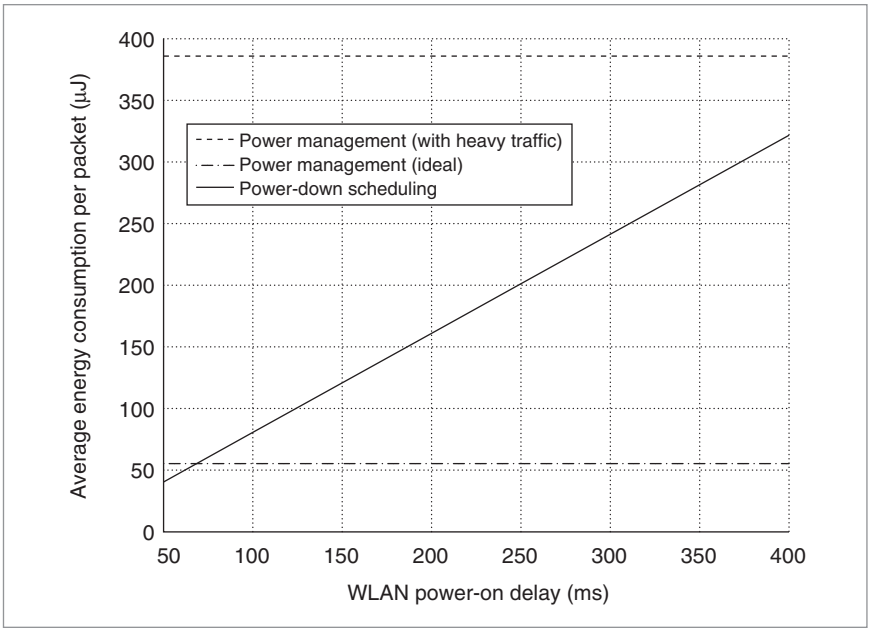

Figure 3. Power-on delay versus energy consumption per packet for IEEE 802.11b.

than average number of broadcast packets.[5] While operating in the 802.11b PM mode, a WLAN card goes into an idle state. Every 100 ms, it wakes up and receives a traffic indication map indicating when the base station will be transmitting data to this particular mobile host. With heavy broadcast traffic, the WLAN interface is rarely in the idle state, and it consumes power as if it were in the always-on mode because the time required to analyze the broadcast packets is larger than the sleep interval. This increase in power consumption occurs even if no applications are running on the mobile host. Measurements indicate that even in less than average amounts of broadcast traffic, the extra processing wastes significant energy.[5]

Because the energy consumption of the 802.11b PM mode increases dramatically in heavy broadcast traffic conditions, we consider an alternate algorithm. If we want to transmit 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 must be accounted for.

Figure 2 shows the timing of this scheduling algorithm. The number of speech frames sent in one packet determines T, the cycle time. Transmission is synchronous, so every T seconds, we send that amount of compressed speech features (T_{tx} is the total amount of time required to transmit the speech features), and the WLAN card stays in the off state the remainder of the time. With larger values of T, we hope to amortize the cost of turning the WLAN card on and off, T_{back_on}, at the expense of longer delay. If we assume that a speech recognition server can process speech at or near real time, the user will experience a 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, reducing delay to maintain interactivity is important. Delays of about 1 second are hardly noticed by the user, whereas delays of about 3 seconds or more can hinder interactivity. For a dictation application, such as e-mail, this delay is less important.

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

Two interesting parameters are power-on time T_{back_on} and the number of speech frames transmitted at once, which dictates total cycle time T. Figure 3 shows power-on delay on the x-axis, and estimated energy consumption on the y-axis. We fixed the value of T at 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 range from 100

ms to 300 ms.) However, in heavy traffic conditions, the PM mode approaches always-on power consumption (the top line in Figure 3), so the scheduling algorithm can give better performance under these conditions. The mobile device must monitor the broadcast traffic and decide between the standard 802.11b PM mode or the scheduling algorithm.

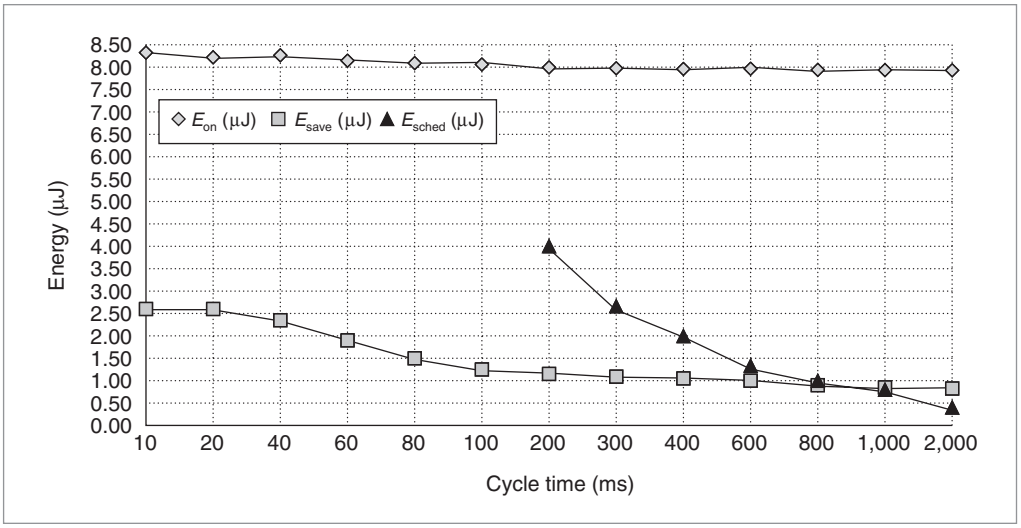

Figure 4. Average energy consumption per 10-ms speech frame versus DSR cycle time for various 802.11b power-saving schemes. (WLAN power-on delay is fixed at 100 ms.)

Finally, in Figure 4, we consider cycle time (T), with T_{back_on} fixed at 100 ms. For this plot, we determined energy cost using measured values of power consumption. We normalized the energy cost to show the average energy required to transmit one frame of speech data. As the total number of frames approaches 80 (T = 800 ms), we can see that the scheduling algorithm (E_{sched}) will 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 2 seconds or more, can further reduce energy consumption and are suitable for applications requiring low interactivity, such as dictation.

Because the 802.11b MAC protocol uses an ARQ protocol with CRC error detection to maintain data integrity, energy consumption is a function of channel signal-to-noise ratio (SNR). After the reception of a good packet, the receiver of the data packet sends an acknowledgment packet (ACK) across a robust control channel. If the sender of the data packet does not receive the ACK, the packet is retransmitted.

We can calculate the expected number of retransmits for a given bit error rate (BER) and packet length and use this number to estimate energy consumption.[10] To estimate the BER, we use an expression for 256-quadrature-amplitude modulation (QAM) in a Rayleigh fading channel. The Rayleigh fading channel assumption is widely used in wireless communications literature as a more realistic alternative to an additive white Gaussian noise channel.[11] By applying our BER expression and power measurements to the expression for energy per *goodput*—a measure of the successfully transmitted bits across an error-prone channel[10]—we can find an expression for the expected energy consumption in a noisy channel.

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.[12] Bluetooth supports a range of about 10 meters, considerably less than the range for 802.11b. Bluetooth supports both data and voice traffic packets. It handles media access using a time-division duplex (TDD) scheme in which each time slot lasts 625 μs. Data packets are available in both high-rate (DH) and medium-rate (DM) packets that occupy either one (DH1 or DM1), three (DH3 or DM3), or five (DH5 or DM5) time slots. Medium-rate packets contain a 2/3 rate error correction code in addition to the ARQ protocol. Voice packets do not use an ARQ protocol because they are time sensitive. Voice packets are available in HV1, HV2, or HV3 types, where the number denotes the amount of error correction: 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 of speech data intended for a machine listener, we can use either data packets or voice packets without considering packet jitter or delay characteristics.

We construct a simple energy model for Bluetooth packet transmissions based on the packet types and error correction overhead. 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 must transmit three times as many HV1 packets as HV3 packets for the same amount of user data. For data packets, energy consumption depends on the size of the data packet being transmitted. Because data packets occupy either one, three, or five TDD slots, we can estimate the energy by using the power-time product.

To account for the idle time between packets, we incorporate the Bluetooth power-saving modes into our model. A node within a Bluetooth piconet can operate in various power management modes.[12] We consider only the park mode, whereby the Bluetooth node temporarily gives up its membership in the piconet to join a list of parked nodes. The node's only activity in park mode is to periodically listen for synchronization and broadcast packets. A Bluetooth node in park mode wakes up to transmit some data, and then returns to 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 performs better than 802.11b, but as T approaches 1.3 seconds, 802.11b uses less energy. The reason for this is that the 100-ms start-up cost of 802.11b is amortized across a larger number of frames, whereas the Bluetooth node remains in the park mode and still consumes power. Powering off a Bluetooth node between packet transmissions is not an option, because the paging and inquiry actions required to join a piconet can easily take more than 10 seconds.

We perform a similar analysis as we did for 802.11b to estimate the energy consumption of Bluetooth data packets in the presence of bit errors. During periods of bit errors, data packets continue to be retransmitted until they are received correctly or until a time-out occurs. We assume binary frequency shift keying (BFSK) modulation with coherent detection under a Rayleigh fading channel.[11] We estimate the packet failure probability[13] and the expected number of retransmissions to get the final energy consumption expression. Voice packets are delivered even in error, so we can estimate energy consumption as a simple power-time product using measured power dissipation values.

DSR trade-offs

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 trade-offs with respect to channel quality, delay, and ASR accuracy. With higher bit rates come small increases in system-level energy consumption, which are due to the overhead of the power-saving algorithms on the wireless device. Table 2 shows this trade-off. 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 DSR and client-side ASR in 802.11b and Bluetooth wireless networks with respect to channel quality (SNR). For DSR, we include both communication and computation (feature extraction and quantization) energy costs. For 802.11b, we show energy consumption of the power on/off scheduling algorithm, with a latency of 240 ms, 480 ms, and 2 s, and unlimited ARQ retransmissions. For the Bluetooth interface, we show energy consumption for both medium- and high-rate data packets as well as the three types of voice packets, with a latency of 480 ms. 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 expect a scaled-down ASR task (with simpler acoustic and language models or a smaller vocabulary) running at real time to give a 60% energy savings. However, this comes at a cost of reduced functionality for the user, perhaps requiring a more constrained vocabulary and speaking style. For the various DSR scenarios in Figure 5, we assume little or no reduction in quality for the end user by maintaining sufficient data integrity through source-coding techniques and ARQ retransmissions.

Table 3 shows the percentages of computation and communication energy for various configurations, as well as the expected battery lifetime with a 1,400 mA-hour, 3.6 V lithium-ion cell. The 802.11b interface with long delays has 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 because of the park mode's overhead, this percentage does not decrease significantly with increased delay. Expected battery lifetimes exceed those of typical cell phones because we don't require real-time communication. Even modest delays of less than 0.5 s can yield a significant battery lifetime.

In a good channel with a high SNR, Bluetooth allows systemwide energy savings of more than 95% compared with full client-side ASR. DH5 packets offer the lowest overhead and best energy savings, whereas DM1 packets offer the most robust operation down to about 10 dB with minimal energy cost. The ARQ retransmission pro-

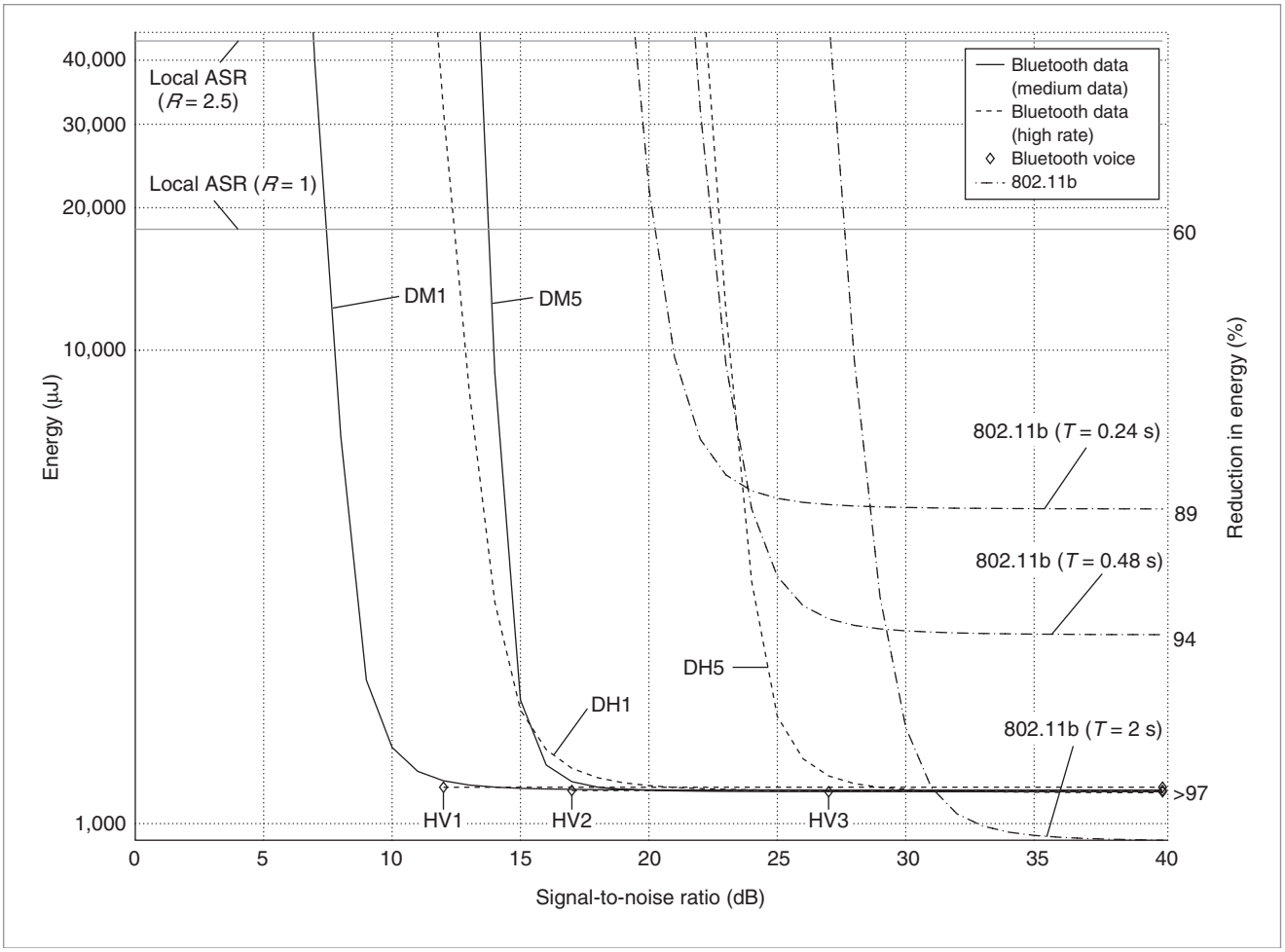

Figure 5. Energy consumption versus signal-to-noise ratio of distributed speech recognition (DSR) and client-side ASR in Bluetooth and 802.11b networks. The *y*-axis is plotted on a logarithmic scale.

Table 3. Summary of energy consumption for ASR and distributed speech recognition (DSR) with high channel signal-to-noise ratio (SNR), where T is the cycle-time delay, and R is the real-time factor.

Type	Computation energy (%)	Communication energy (%)	Total per speech frame (μJ)	Battery lifetime (hr)
Bluetooth DSR (T = 0.48 s)	32	68	1,170	43.1
802.11b DSR (T = 0.48 s)	15	85	2,500	20.2
802.11b DSR (T = 2 s)	42	58	920	54.8
Local ASR (R = 2.5)	100	0	45,000	1.12

tocol rapidly increases energy consumption after reaching an SNR threshold. It's possible to operate at a lower SNR through packet fragmentation, reducing the probability of a packet's being received in error. Comparing the DH1 and DH5 data packets in Figure 5 makes this evident. The longer packet length in DH5 packets causes a sharp increase in retransmissions and energy consumption at about 25 dB, whereas DH1 packets can operate down to 15 dB before the number of retransmissions becomes excessive. In addition, using forward error correction (FEC) bits can lower the probability of a packet retransmission. The Hamming code in DM1 and DM5 packets allows operation to about 10 and 16 dB, respectively.

The energy consumption in Bluetooth voice packets is independent of SNR because Bluetooth doesn't use an ARQ protocol. Uncoded HV3 packets have the lowest overhead and therefore the lowest energy consumption per frame of speech, but they operate to only about 27 dB. Beyond that, the probability of a bit error exceeds 10^{-3}, which has a noticeable impact on ASR accuracy.[6] HV1 and HV2 packets operate to about 12 and 17 dB, respectively, with little noticeable loss in recognition accuracy.

Finally, 802.11b networks allow systemwide energy savings of approximately 89% to 94% with relatively small values of *T*. With larger values of *T*, such as 1 second or more, 802.11b-based systems use less energy than Bluetooth-based systems. However, because of larger packet overhead, larger maximum packet sizes, different modulation techniques, and lack of payload error-correcting codes, 802.11b networks don't operate as well in lower SNR ranges. Packet fragmentation or a more robust modulation technique with a lower maximum bit rate can extend the lower SNR range at the cost of increased energy consumption, but we haven't considered these effects here. However, 802.11b does offer increased range and might be more appropriate in certain scenarios.

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. In the future, however, the use of low-power ASIC chips for ASR might help reduce client-side ASR's energy consumption. Advancements in this area include analog signal processing techniques, where very low-power implementations of hidden Markov models are possible. Such contributions will shift the emphasis from distributed to client-centric computing. Other speech-related applications, such as speaker identification, could require distributed computation because it might not be practical to store models of all speakers on every mobile host. Future work in this area might include a transmission strategy based on channel quality to minimize the energy overhead of packet retransmissions. ■

Acknowledgments

We thank John Anckorn of Hewlett-Packard Laboratories and Wajahat Qadeer of Stanford University for their contribution of Bluetooth power consumption measurements, and we thank Mat Hans and Mark Smith of HP Labs for their continued support of our work.

References

1. B.T. Lilly and K.K. Paliwal, "Effect of Speech Coders on Speech Recognition Performance," *Proc. 4th Int'l Conf. Spoken Language Processing* (ICSLP 96), vol. 4, IEEE Press, 1996, pp. 2344-2347.
2. H.K. Kim and R. Cox, "Bitstream-Based Feature Extraction for Wireless Speech Recognition," *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing* (ICASSP 00), IEEE Press, 2000, vol. 3, pp. 1607-1610.
3. V.V. Digilakis, L.G. Neumeyer, and M. Perakakis, "Quantization of Cepstral Parameters for Speech Recognition over the World Wide Web," *IEEE J. Selected Areas in Communications*, vol. 17, no. 1, Jan. 1999, pp. 82-90.
4. *Speech Processing, Transmission and Quality Aspects (STQ); Distributed Speech Recognition; Front-End Feature Extraction Algorithm; Compression Algorithms*, Std. ETSI ES 201 108 v1.1.2, 2000, European Telecommunications Standards Institute, http://www.etsi.org.
5. A. Acquaviva et al., "Remote Power Control of Wireless Network Interfaces," *Proc. 13th Int'l Workshop Power and Timing Modeling Optimization and Simulation (PATMOS 03)*, LNCS 2799, Springer-Verlag, 2003, pp. 369-378.
6. B. Delaney, T. Simunic, and N. Jayant, *Energy-Aware Distributed Speech Recognition for Wireless Mobile Devices*, tech. report HPL-2004-106, Hewlett Packard Laboratories, 2004.
7. W.R. Hamburgen et al., "Itsy: Stretching the Bounds of Mobile Computing," *Computer*, vol. 34, no. 4, Apr. 2001, pp. 28-36.
8. J.R. Deller Jr., J.G. Proakis, and J.H.L. Hansen, *Discrete-Time Processing of Speech Signals*, Prentice Hall, 1987.
9. T. Simunic, L. Benini, and G.D. Micheli, "Energy-Efficient Design of Battery-Powered Embedded Systems," *IEEE Trans. VLSI Systems*, vol. 9, no. 1, Feb. 2001, pp. 15-28.
10. J.-P. Ebert et al., *Measurement and Simulation of the Energy Consumption of a WLAN Interface*, tech. report TKN-02-010, Telecommunication Networks Group, Technical Univ. of Berlin, 2002.
11. J.G. Proakis, *Digital Communications*, 3rd ed., McGraw-Hill, 1995.
12. *Bluetooth specification,* v1.1, 2002, https://www.bluetooth.org/spec.
13. M. Valenti, M. Robert, and J. Reed, "On the Throughput of Bluetooth Data Transmissions," *Proc. IEEE Wireless Communications and Networking Conf.* (WCNC 02), vol. 1, IEEE Press, 2002, pp. 119-123.

Brian Delaney is a member of the technical staff at the Massachusetts Institute of Technology's Lincoln Laboratory. He was pursuing his PhD at the Georgia Institute of Technology when he performed this work. His research interests include speech recognition, energy-aware computing, multimedia communications, and the interaction of these three areas in speech-enabled mobile wireless clients. Delaney has a BS in computer engineering, and an MS and a PhD in electrical engineering, all from the Georgia Institute of Technology. He is a member of the IEEE Signal Processing Society.

Tajana Simunic is a faculty member of the Computer Science Department at the University of California, San Diego. She was a full-time researcher at Hewlett-Packard Laboratories and a research faculty member at Stanford University when she performed this work. Her research interests include low-power system design, embedded systems, and wireless system design. Simunic has an MS in electrical engineering from the University of Arizona, and an MS in engineering management and a PhD in electrical engineering from Stanford University. She is a member of the IEEE Circuits and Systems Society.

Nikil Jayant is on the faculty of the Electrical and Computer Engineering Department at Georgia Tech, where he is a Georgia Research Alliance Eminent Scholar, holds the John Pippin Chair in Wireless Systems, and is the director of the Georgia Tech Broadband Institute. He is also executive director of the Georgia Centers for Advanced Telecommunications Technology. His research interests include broadband access, lifestyle computing, and ubiquitous multimedia. Jayant has a PhD in electrical communication engineering from the Indian Institute of Science, Bangalore, India. He is a Fellow of the IEEE and a member of the National Academy of Engineering.

■ Direct questions and comments about this article to Brian Delaney, MIT Lincoln Laboratory, Rm. C-260, 244 Wood Street, Lexington, MA 02420; bdelaney@ll.mit.edu.