A WLAN SCHEDULING ALGORITHM TO REDUCE THE ENERGY CONSUMPTION OF A DISTRIBUTED SPEECH RECOGNITION FRONT-END

Brian Delaney, Nikil Jayant

Georgia Institute of Technology
School of Electrical and Computer Engineering
Multimedia Communications Lab
Atlanta, GA 30332
{*delaney,njayant*}*@ece.gatech.edu*

Tajana Simunic

Hewlett-Packard Laboratories
Mobile and Media Systems Lab
1501 Page Mill Road
Palo Alto, CA 94304

ABSTRACT

In distributed speech recognition, speech features are computed on a mobile device, compressed, and sent to a server that performs the computationally intensive search for the most likely word sequence. Much of the current research in distributed speech recognition has been in the area of feature compression and communication robustness over wireless links, including error correction and concealment techniques. However, another challenge in designing a distributed speech recognition system is minimizing the energy consumption on the mobile device. We consider quality-of-service tradeoffs including compression ratio and overall system latency. Our measurements verify that for high speed wireless interfaces such as 802.11b, small changes in compression rates have little effect on system level energy consumption. However, for wireless networks with lower power/bit-rate ratios such as Bluetooth, the choice of bit-rate and compression ratio becomes more important. We present a wireless LAN scheduling algorithm to minimize the energy consumption of a distributed speech recognition front-end on a mobile device. By powering down the 802.11b interface when not in use, we are able to reduce the energy consumption by up to a factor of 5 in heavy traffic conditions. Increasing the total amount of time spent in the off state to almost one second will allow the system to save power regardless of traffic conditions. We compare the results of this power saving algorithm to the low-power mechanisms of Bluetooth. The lower overhead of Bluetooth allows for greater energy savings with a much lower delay of approximately 300ms.

1. INTRODUCTION

Wireless hand-held devices such as PDAs, cell phones, and other embedded 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. Figure 1 shows a block diagram of a typical speech recognition system. The signal processing front-end is a very 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. Complex tasks require back-end search algorithms that use large amounts of memory and CPU cycles. Therefore, distributing the speech recognition across the network is an attractive alternative for these small wireless devices.

Previous work on distributed speech recognition (DSR) has been mainly focused on techniques for the quantization of speech

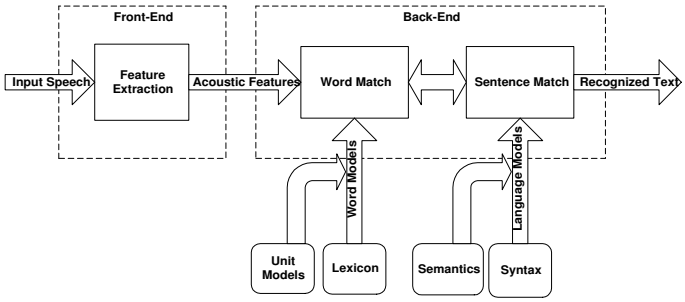


Fig. 1. Block diagram of a typical speech recognizer [1].

parameters. Most involve some form of vector quantization with bit rates in the low kbps range. In [2], a two stage vector quantizer was used to achieve a fixed rate of 4.0 kbps with little loss of recognition accuracy. In [3] a scalable quantization scheme was developed for bit rates ranging from less than 1 kbps to around 3 kbps. A wider range of quantization schemes was investigated in [4] 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 [5]. Finally, the ETSI Aurora DSR standard includes a simple intra-frame vector quantizer with some error detection and framing techniques in [6].

A low-power DSP solution that uses less than 1mW of power and conforms to the ETSI standard is presented in [7]. Some architectural and algorithmic optimization techniques to minimize the energy consumption required to compute speech recognition features in software were presented in [8]. The fully optimized source code consumed 83% less energy than the initial floating-point source code. It also ran 34 times faster, which allowed for further energy savings through dynamic voltage scaling. This work, however, did not consider the quantization and transmission related aspects of the DSR front-end. Given that a wireless interface can consume more than half the total power used on an embedded device, efficient use of the radio in DSR is an important consideration.

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. En-

ergy efficient channel coding and traffic shaping to exploit battery lifetime of portable devices were proposed in [9]. A physical layer aware scheduling algorithm aimed at efficient management of sleep modes in sensor network nodes is illustrated in [10]. Energy efficiency can be improved at the data link layer by performing adaptive packet length and error control [11]. At the protocol level, there have been attempts to improve the efficiency of the standard 802.11b, and proposals for new protocols [12, 13, 14]. Packet scheduling strategies can also be used to reduce the energy consumption of transmit power. In [15] 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 [16]. Savings of as much as 50% in WLAN power consumption relative to just using 802.11 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 [17, 18]. Energy-performance trade-offs based on application needs have been recently addressed [19]. Several authors exploit the energy-QoS trade-off [20, 21, 22, 23]. A different approach is to perform transcoding and traffic smoothing at the server side by exploiting estimation of energy budget at the clients [24]. 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 [25]. 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 [26, 27, 28, 29].

In this paper, we investigate the energy consumption involved in computation as well as wireless transmission of compressed speech data for a fixed-point DSR front-end running on an embedded device. We will consider wireless optimization at the application level and present results specific to DSR but that may be applied to more general types of traffic. We will also examine some energy-QoS trade-offs relating to DSR, including speech recognition accuracy and delay. The architecture of the embedded system used in the experiments is that of the SmartBadge 4 embedded system developed at the Mobile and Media Systems Lab at HP Labs [30]. 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. This hardware platform is similar to the sensor network system studied in [21]. The SmartBadge has speech/audio driven I/O, so DSR can provide some level of user interaction through a voice user interface.

2. DISTRIBUTED SPEECH RECOGNITION

In distributed speech recognition the speech features, typically mel-frequency cepstral coefficients (MFCC), are calculated at the client and sent over the wireless network to a server. This feature extraction step is a small fraction of the overall computation needed to perform speech recognition. Figure 1 shows a block diagram of a typical Hidden Markov Model based speech recognizer. The back-end requires much more computation and memory, especially for large vocabulary tasks. This back-end speech recognition search including HMM state output evaluation and Viterbi search is performed at the server. In order to minimize the bit rate, the MFCCs are first compressed using some quantization scheme.

2.1. 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 consist of 13 mel-frequency cepstral coefficients computed every 10ms. Secondary features, consisting of first and second time derivatives of the cepstrum, are also 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 [31].

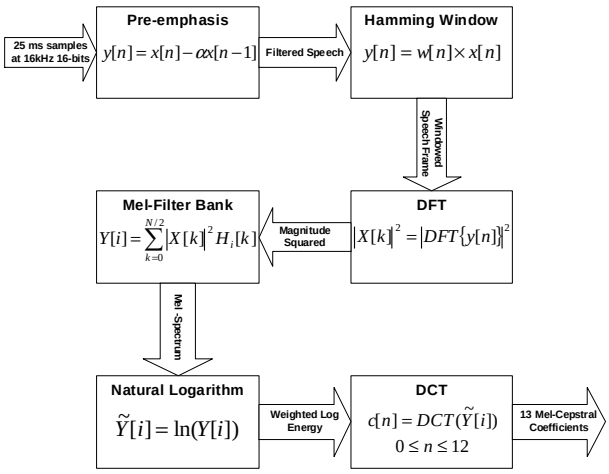


Fig. 2. The algorithm used to compute the mel-cepstrum.

In practice, the mel-frequency cepstral coefficients can be computed using the algorithm in Figure 2. We used digitized speech at 16 linear bits per sample, sampled at 16kHz. A pre-emphasis filter whitens the speech signal and overlapping frames of 25ms are multiplied by a Hamming window. Next the magnitude squared of the discrete Fourier transform (DFT) is computed. The magnitude squared is processed by a set of mel-filter banks to produce an estimate of the mel-spectrum. The mel-filter banks are implemented as a series of overlapping triangle filters, $H_i[k]$, that are centered on equally spaced frequencies in the mel-scale. The result is an estimate of the total energy in the ith critical band:

$$Y[i] = \sum_{k=0}^{N/2} |X[k]|^2 \, H_i[k] \tag{2}$$

where $X[k]$ is the DFT of the windowed speech signal and $H_i[k]$ contains the filter-bank coefficients. Finally, the logarithm of the mel-spectrum is taken to produce a weighted log energy, $\tilde{Y}[i]$. The

weighted log energy is real and even, so the inverse Fourier transform can be implemented as a discrete cosine transform (DCT) with equivalent results. In order to calculate the MFCCs in real-time on the StrongARM processor, we used the low-power fixed-point implementation presented in [8].

2.2. Vector Quantization

Although many different techniques have been proposed, the most 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. The server can then lookup the codebook index to get an approximation to the original speech vector. A Euclidean distance measure can be used to find the best centroid for a given input vector. 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 [4]. We used the existing bit allocation in [4] to train a set of codebooks using a K-means training algorithm with bit rates ranging from 1.2 kbps to 2.0 kbps. However, the speech recognition task considered was a relatively small vocabulary with clean speech. Higher bit rates used in the ETSI standard (4.8 kbps) will provide more robust operation under a variety of speakers and noise conditions [6]. Therefore, we have added an additional bit allocation that is similar to the ETSI standard that will operate at 4.2kbps.[1] Although data regarding the word error rate (WER) of this particular quantization scheme under the speech recognition task in [4] is not available, it should be very close to that of uncoded speech. Source code to perform the quantization on the MFCC data was written in fixed point for the StrongARM processor. Table 1 shows the resulting bit rates and word error rates from [4] on the rows labeled 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%.

Table 1. Word error rate for several bit rates [4].

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	≈ 6.55
16kHz	256	6.55

3. ENERGY CONSUMPTION

In this section, we measure the energy consumption of the DSR front-end. We will use these results to characterize different aspects of a DSR system (including speech recognition accuracy, transmission bit rate, and delay) with respect to energy consumption. We consider energy consumption from both computation and wireless transmission.

For computation energy, the results are computed by a cycle-accurate energy consumption simulator. The simulator includes 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 [32]. In addition to the energy consumption simulator, we also directly measured the average current going into the StrongARM CPU and the WaveLAN 802.11b wireless card. This can be used to compute the average power dissipation.

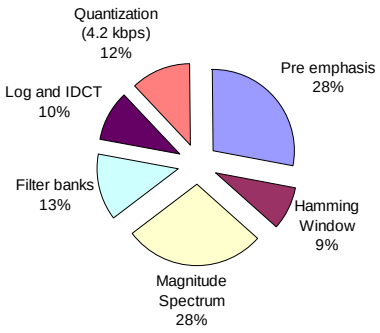


Fig. 3. Energy consumption per DSR functional block.

3.1. Energy Required for MFCC Calculation and Quantization

We ran our fixed-point MFCC algorithm with quantization through the energy consumption simulator for the SmartBadge platform. The relative energy consumption for each basic block of the signal processing algorithm is shown in Figure 3. The energy consumption is shown for a single frame of speech with the exception of the pre-emphasis filter which processes several frames at once. The total energy consumption required to process one frame of speech is approximately $381.6\ \mu$Joules. Here we used the software vector quantization scheme with the highest bit rate and encoding cost. Notice that the quantization cost is still a small fraction of the overall energy consumption required for the computation. 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.

Figure 4 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 to the overall computation. There is a 100 mW increase

[1] We use the same bit allocation as in the ETSI standard with the exception of leaving out the log energy as well as bits required for header and error protection as described in the standard.

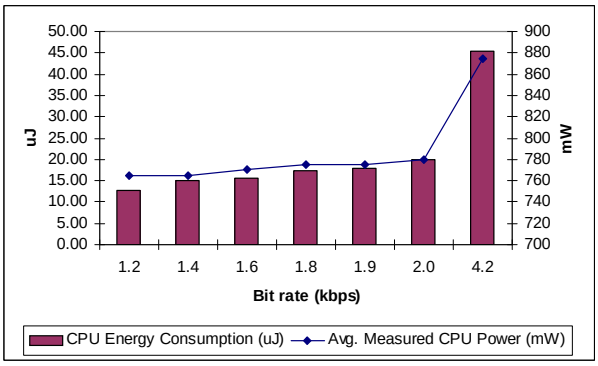


Fig. 4. Computational energy usage and measured average power for different quantization bit allocation schemes.

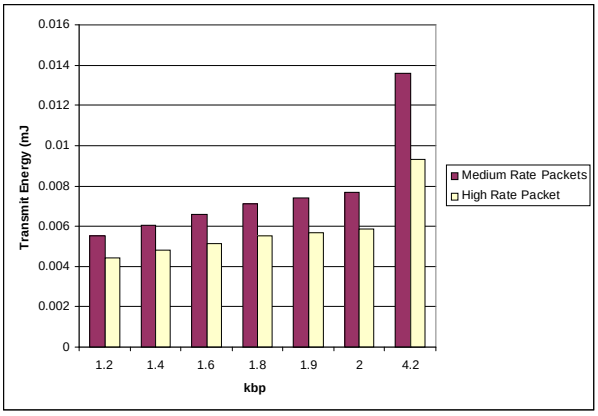


Fig. 5. Energy consumption per frame of speech with varying compression rates for Bluetooth radio.

in CPU power consumption but a greater than 50% reduction in WER between the highest and lowest bit error rate. Even the 2.0 kbps bit rate offers a large reduction in WER with little change in CPU power consumption. Therefore, if better performance is desired, larger codebooks and higher bit rates are probably worth the small extra encoding cost.

3.2. Energy Consumption for Wireless Transmission

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. These measurements were taken for each quantization scheme and for both the "always on" and 802.11b power management (PM) mode configurations. We also consider the energy consumption of a Bluetooth radio, which has a much lower power/bit-rate ratio.

Figure 5 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. High speed packets use a stop-and-wait ACK protocol with some CRC error detection within the packet. Medium rate packets use a 2/3 rate FEC with no ACK protocol. We have assumed that both packet types are only transmitted once without error. We can see in figure 5 that there is a higher energy cost for medium rate packets due to the FEC overhead. However, these packets might 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 wait time between packets which will consume energy as well. Section 3.4 will discuss the implications of this wait time.

For 802.11b, 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 802.11 MAC layer protocols both add significant overhead for small packet sizes. The 11 Mbps WLAN interface is under-utilized 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 as shown in Figure 6. 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. Due to the relatively high bit rates provided by 802.11, the WLAN interface spends most of it's time waiting for the next packet to transmit. The 802.11 PM mode can provide some savings in energy consumption but this does not hold under heavy traffic conditions.

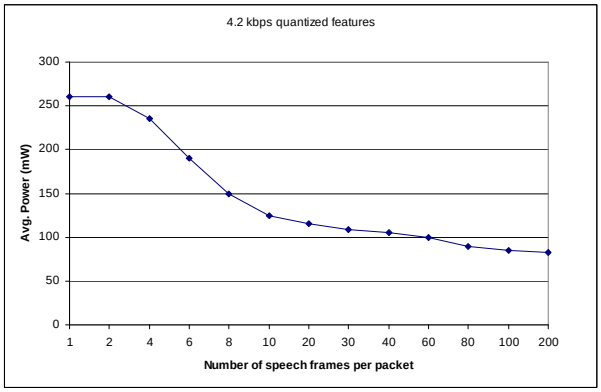
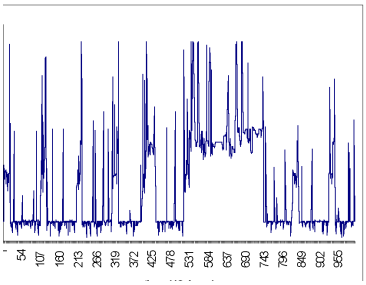
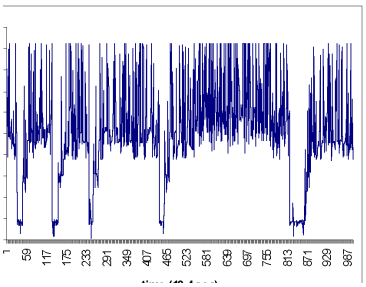


Fig. 6. Average power dissapation for the WLAN device with varying number of speech frames per packet. Each speech frame adds 10ms of delay.

It has been shown in [16] that the 802.11b power management mode may not serve to reduce energy consumption in heavy traffic 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 will happen even if there are no applications running on the mobile host. Figure 7 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.



(a) light traffic



(b) heavy traffic

Fig. 7. WLAN power consumption in 802.11b PM mode in light and heavy traffic conditions.

3.3. Synchronous Scheduling of the WLAN Interface

Since the energy consumption of PM mode on 802.11b networks breaks down in heavy traffic conditions, we consider an alternate algorithm here. 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 which needs to be accounted for.

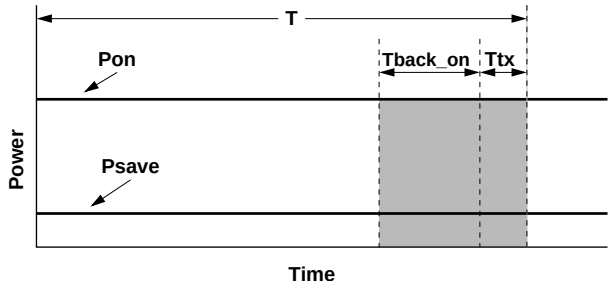


Fig. 8. The timing of the WLAN scheduling algorithm.

Figure 8 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. Larger values of T will result in longer delay for the speech recognition application. 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:

$$E_{on} = P_{on} \times T \tag{3}$$

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

$$E_{save} = P_{save} \times T \tag{4}$$

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 only be on during the shaded region in figure 8. 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:

$$E_{save} = P_{on} \times (T_{back\_on} + T_{Tx}) \tag{5}$$

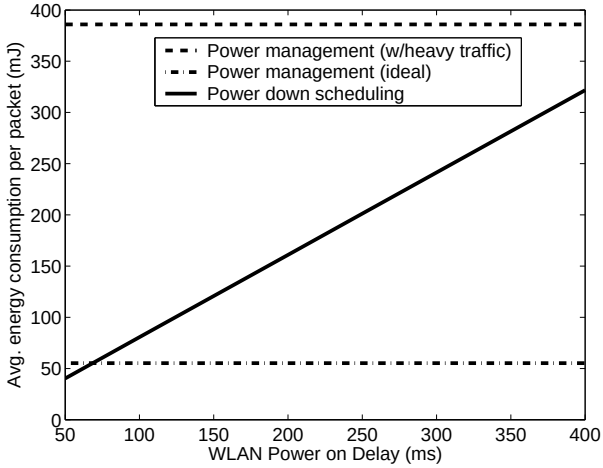


Fig. 9. WaveLAN power on delay vs. energy consumption per packet.

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 9 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 9. This is a reduction in energy consumption by about a factor of 5. 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 802.11 PM mode or the scheduling algorithm.

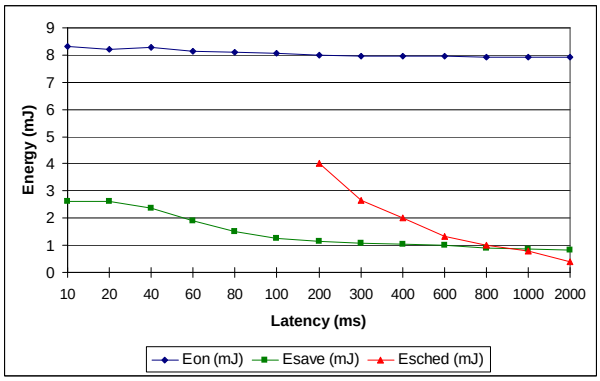


Fig. 10. Average energy consumption vs. number of speech frames sent per packet for various 802.11 power save schemes (WLAN power on delay is fixed at 100ms).

The other parameter to consider is the total delay T which determines the total number of packets sent at once. 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 is taken by the mobile device. A server which 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 1 second may hardly be noticed by the user, whereas delays of around 3 seconds or more may hinder interactivity. For a dictation application, such as 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.

Figure 10 shows the results for varying delay, 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 (Esched) will be able to outperform the PM mode configuration (Esave) 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.

3.4. Bluetooth Power Management

A node within a Bluetooth piconet can operate in a variety of different power management modes [33]. 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 *park* mode the Bluetooth node 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. In *hold* mode, the node goes into a low power state until some specified interval, after which it powers up to transmit. This is very similar to the WLAN scheduling algorithm discussed in the previous section.

A Bluetooth node in hold mode will wait a specified period of time, switch to active mode to transmit, and then switch back to hold mode. The approximate energy consumption can be obtained with the following equation:

$$E = P_{tx} \times T_{tx} + P_{hold} \times T_{hold} + E_{transition} \quad (6)$$

where P_{tx} and T_{tx} are the power consumption and times to transmit the data, P_{hold} and T_{hold} are the power consumption and time spent in the hold state, and $E_{transition}$ is the total energy required to transition from the hold state to the active state and back. The transition from hold to active takes approximately 11.62 ms, the reverse transition takes approximately 1.68 ms.

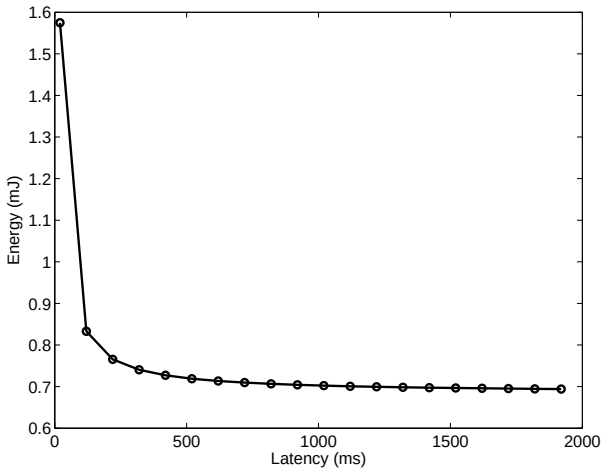


Fig. 11. Energy per frame of speech vs. latency for a Bluetooth node transmitting DSR traffic.

By varying the amount of data transmitted at once, we can increase the amount of time spent in the low-power hold state, similar to specifically turning the card off for T seconds in 802.11. Figure 11 shows the tradeoff between speech recognition latency and energy consumption per frame of speech. The total energy consumption per frame is less than 802.11 for all but the WLAN power off scheduling at the largest delay. Powering off a Bluetooth node between transmits is not an option since the paging/incquiry options required to join a piconet can easily take over 10 seconds. However, since the transition time from hold to active and back is small, we see a much faster reduction in energy consumption with respect to increased latency in Bluetooth. We can achieve significant reductions in energy consumption after around 300-400ms of delay, which would hardly be noticed by any user of a voice-user interface.

4. 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. We show that QoS tradeoffs involving compression rates are less important for high speed wireless networks such as 802.11, but they can have a larger impact for wireless interfaces with lower data rates and power consumption such as Bluetooth.

The total latency of the system, which is dictated by the amount of DSR data buffered before transmitting, has the greatest impact on energy consumption. Larger packets allow the networking overhead to be amortized across a larger number of speech frames, which drives down the total energy cost per 10 ms frame of speech. In 802.11, we can couple the energy savings from larger packet sizes with a power on/off scheduling algorithm to reduce the overall energy consumption by a factor of 5 over the 802.11 PM mode in heavy broadcast traffic conditions. Additionally, we can identify the delay which will always yield a lower energy consumption even in ideal traffic conditions. For a power on delay of 100ms, this value is approximately 900ms or 90 frames of speech data.

The Bluetooth standard provides a low-power scheduling mechanism with considerably less overhead than 802.11. Use of this power-saving scheme allows for substantial reductions in power consumption with about 300-400ms of delay. Increasing the delay past 500ms only gives marginal reductions in energy consumption.

5. ACKNOWLEDGEMENTS

The authors would like to thank John Anckorn and Wajahat Qadeer for their contribution of Bluetooth power consumption measurements.

6. REFERENCES

[1] Lawrence R. Rabiner, "Applications of speech recognition to the area of telecommunications," in *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, 1997, pp. 501–510.

[2] Ganesh N. Ramaswamy and Ponani 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)*, 1998, vol. 2, pp. 977–980.

[3] 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.

[4] 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.

[5] Constantinos Boulis, Mari Ostendorf, Eve A. Riskin, and Scott Otterson, "Graceful degradation of speech recognition over packet erasure networks," *IEEE Transactions on Speech and Audio Processing*, vol. 10, pp. 580–590, 2002.

[6] "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.

[7] Etienne Cornu and Hamid 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.

[8] B. Delaney, N. Jayant, M. Hans, T. Simunic, and A. Acquaviva, "A low-power fixed-point front-end feature extraction for a distributed speech recognition system," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2002, vol. 1, pp. 793–796.

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

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

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

[12] 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.

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

[14] 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.

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

[16] A. Acquaviva, Tajana Simunic, Vinay Deolalikar, and Sumit Roy, "Remote power control of wireless network interfaces," *Lecture Notes in Computer Science*, October 2003.

[17] T. Simunic, L. Benini, P. Glynn, and G. De Micheli, "Event-driven power management," *IEEE Transactions on CAD*, July 2001.

[18] 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.

[19] R. Kravets and P.and Krishnan, "Application-driven power management for mobile communication," .

[20] R. Min and A. Chandrakasan, "A framework for energy-scalable communication in high-density wireless networks," .

[21] Rex Min, *Energy and Quality Scalable Wireless Communicaiton*, Ph.D. thesis, M.I.T., June 2003.

[22] E. Takahashi, "Application aware scheduling for power management on ieee 802.11," .

[23] C. Luna, Y. Eisenberg, R. Berry, T. Pappas, and A. Katsaggelos, "Transmission energy minimization in wireless video streaming applications," .

[24] P. Shenoy and P. Radkov, "Proxy-assisted power-friendly streaming to mobile devices," .

[25] 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.

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

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

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

[29] Y. Lu, L. Benini, and G. De Micheli, "Operating system directed power reduction," in *ISLPED*, July 2000, pp. 37–42.

[30] G. Q. Maguire, M. Smith, and H. W. Peter Beadle, "Smartbadges: A wearable computer and communication system," 6th International Workshop on Hardware/Software Codesign, 1998, Invited Talk.

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

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

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