Reduced Energy Consumption and Improved Accuracy for Distributed Speech Recognition in Wireless Environments

A Thesis
Presented to
The Academic Faculty

by

Brian Delaney

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Electrical and Computer Engineering
Georgia Institute of Technology
September 2004

Reduced Energy Consumption and Improved Accuracy for Distributed Speech Recognition in Wireless Environments

Approved by:

Nikil Jayant, Advisor

Mary Ann Ingram

Mark Clements

Mat Hans

Chin-Hui Lee

Date Approved: September 15, 2004

To my family

ACKNOWLEDGEMENTS

Around twelve years ago my high school guidance counselor described Georgia Tech as an up and coming school with a top-notch engineering program as well as the host of the 1996 Summer Olympic village. I decided to apply, and after more than a decade, I am receiving my third degree from Georgia Tech. This would not be possible without the encouragement and support of my family. My parents, Barbara and Jim, have been very supportive both financially and emotionally throughout the years. My brother, Mike, and sister-in-law, Kris, have also helped me in countless ways with encouragement when I needed it the most as well as plenty of free meals. My neice and nephew, Morgan and Dylan, showed us all that play time is more important than work. My girlfriend, Laura Lo, helped not only with her keen editing skills but, more importantly, with her unconditional support of my goals and aspirations. My fellow students provided many hours of interesting conversation as well as a sounding board for my ideas. My lab mates, Jang-Hyun Yoon, Cagatay Candan, Roberto Uzcategui, and Babak Firoozbakhsh were all friendly, outgoing, and sympathetic to the trials and tribulations of graduate school. Jon Arrowood was always cheerful and encouraging, even during those long Yamacraw poster presentations. Several people at HP Labs were extremely helpful in developing my thesis work. Tajana Simunic was a great source of ideas, and her constructive criticism was instrumental in getting my publications accepted. Mat Hans and Mark Smith continually supported my work through an HP funded assistantship as well as through useful professional advice. Finally, my thesis advisor, Nikil Jayant, was a great mentor and set an example of professionalism that his students will carry with them throughout their careers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The central theme of this dissertation is the study of a multimedia client for pervasive wireless multimedia applications. Speech recognition is considered as one such application, where the computational demands have hindered its use on wireless mobile devices. Our analysis considers distributed speech recognition on hardware platforms with PDA-like functionality (i.e. wireless LAN networking, high-quality audio input/output, a low-power general-purpose processing core, and limited amounts of flash and working memory.) We focus on quality of service for the end-user (i.e. ASR accuracy and delay) and reduced energy consumption with increased battery lifetimes. We investigate quality of service and energy trade-offs in this context. We present software optimizations on a speech recognition front-end that can reduce the energy consumption by over 80% compared to the original implementation. A power on/off scheduling algorithm for the wireless interface is presented. This scheduling of the wireless interface can increase the battery lifetime by an order of magnitude. We study the effects of wireless networking and fading channel characteristics on distributed speech recognition using Bluetooth and IEEE 802.11b networks. When viewed as a whole, the optimized distributed speech recognition system can reduce the total energy consumption by over 95% compared to a client-side ASR implementation. We present an interleaving and loss concealment algorithm to increase the robustness of distributed speech recognition in a burst error channel. This improvement allows a decreased reliance on error protection overhead, which can provide reductions in transmit energy of up to 46% on a Bluetooth wireless network. The findings presented in this dissertation stress the importance of energy-aware design and optimization at all levels for battery-powered wireless devices.

CHAPTER I

INTRODUCTION

Mobile wireless devices are a driving force in the computer and communications industry. The demand for tetherless access to data will drive the industry toward smaller but more capable devices. It has already begun with the widespread use of Personal Digital Assistants (PDAs) and cellular telephones. The trend continues toward smaller devices which offer high quality wireless web browsing, multimedia e-mail and messaging services, as well as personal data management (scheduling, contacts, etc.) These pocket sized devices have small screens and little to no keyboard input, so appropriate use of speech technology can allow users to interact with the system in a natural manner. However, the problems of speech recognition, speech synthesis, and wireless connectivity are far from solved, thus the currently fielded solutions have many deficiencies. Integrating each of the technologies into a robust wireless voice user interface is a difficult task given the problems associated with each of the enabling technologies. The high computational demands of multimedia processing applications on digital hardware further complicates the problem. Given that portable wireless devices are limited in computation, memory size, wireless bandwidth, and battery energy, distributing the speech recognition task across the network is an attractive alternative. Speech recognition can be a computationally demanding application that can easily use all available resources. An in-depth understanding of these issues in the context of a distributed speech recognition system enables designers of future systems to build more efficient devices and algorithms. This thesis involves the study of a pervasive multimedia client specifically applied to distributed speech recognition.

In particular, we study the effects of wireless networking and fading channel characteristics on distributed speech recognition. We investigate quality of service and energy trade-offs in this context.

1.1 Mobile Wireless Devices

Mobile wireless devices are portable computers or communication devices with wireless networking support. They include cellular telephones, PDAs, wearable computers, and other embedded devices. They are meant to be easily transported, so they are typically small and light with limited battery power. Portable information devices are becoming smaller and more capable each year. However, due to cost factors, small memory allocation, power limitations, slower processors, and software limitations, a completely speech enabled device remains out of reach for the time being. However, through thin client design, we can speech enable a device by offloading ASR computation to a server.

1.1.1 Cellular Telephones

The cellular telephone is perhaps the most ubiquitous wireless device. Digital cell phone manufacturers and service providers have phones available with limited wireless Internet access and include more PDA-like functionality such as personal information management applications (i.e. contacts, calendar, and e-mail access.) Cell phones are characterized by very small displays, and little processing power for applications. Their main function is for voice communication, although new phones will add more features such as cameras and Internet access with data rates in the hundreds of kbps. There is a lack of well defined operating systems for phones and adding software requires the use of the Internet or data connection to a PC. The Java 2 Micro Edition (J2ME) allows for portable software to be written for many cellular phone devices, but it has limitations. J2ME applets run in a limited virtual machine that does not allow full access to the hardware and network, and the interpreted Java byte-code can

be slow on many devices. The limitations make cellular phones, for the time being, well suited to audio only interfaces. However, the close talking microphone is of great benefit to the VUI since it will be less likely to pick up background noise.

1.1.2 Personal Digital Assistant

A Personal Digital Assistant (PDA) is a small hand-held computer with limited capability. A comprehensive discussion of emerging PDA standards is given in [12]. PDAs have a higher quality display than a cell phone, 240-by-320 pixels is not uncommon, with color displays in higher-end models. However, this resolution is still one sixth the size and one fourth the resolution of the absolute minimal acceptable display on a modern desktop PC.

PDAs are limited in memory (usually 64 Mbytes) as well as processing power. The current top of the line PDA runs the 400 MHz Intel XScale processor. Other models run a 200 MHz version of the same or a 206 MHz StrongARM processor. New PDAs are being designed which will run the new low-power Crusoe processor from Transmeta [32]. Unlike cellular phones, PDAs typically use a standard operating system, although there are several to choose from (Windows CE, Palm OS, and Embedded Linux). Current PDAs use a pointing device and a few buttons as the sole medium for user interaction, although some have folding keyboards available. Many PDAs have some form of handwriting recognition, but this can be slow and often requires special glyphs for certain characters to improve accuracy. Through practice a user can become quite proficient at handwriting input, but it still remains cumbersome for many types of input. Due to their limited processing power and memory, PDAs are minimally suitable for large vocabulary continuous speech recognition (LVCSR) and lack the memory required for high-quality concactenative TTS without dedicated hardware support. A microphone for a small PDA will most likely be small, cheap, and mounted directly on the device itself. Using a far field microphone gives a noisy

signal, and therefore speech recognition accuracy will drop without environmental robustness techniques.

Wireless connectivity options for PDAs have grown over the years. Early wireless PDAs used a network such as BellSouth's Mobitex or GoAmerica, which are characterized by poor coverage, low data rates (9.6 kbps), and high cost compared to cellular phones but with a range that covered most metropolitan areas. Currently many PDAs are designed with builtin 802.11b or Bluetooth support. Bluetooth may provide the solution for data rates less than 1 Mbps with low cost and short range, while 802.11b offers increased range and data rates up to 11 Mbps. Many businesses, schools, and public gathering places are being equipped with 802.11 hotspots that allow wireless internet access either for free or a small subscriber fee.

Finally, due to the small display, several alternatives to HTML have been defined. Among them are Hand-held Device Markup Language (HDML), Wireless Application Protocol (WAP), Wireless Markup Language (WML), and i-MODE which uses HTML tags. These standards are for visual content and have no provisions for audio interfaces. There is an increasing amount of content available using these standards as Internet capable phones and PDAs permeate the marketplace.

1.1.3 Wearable Computers

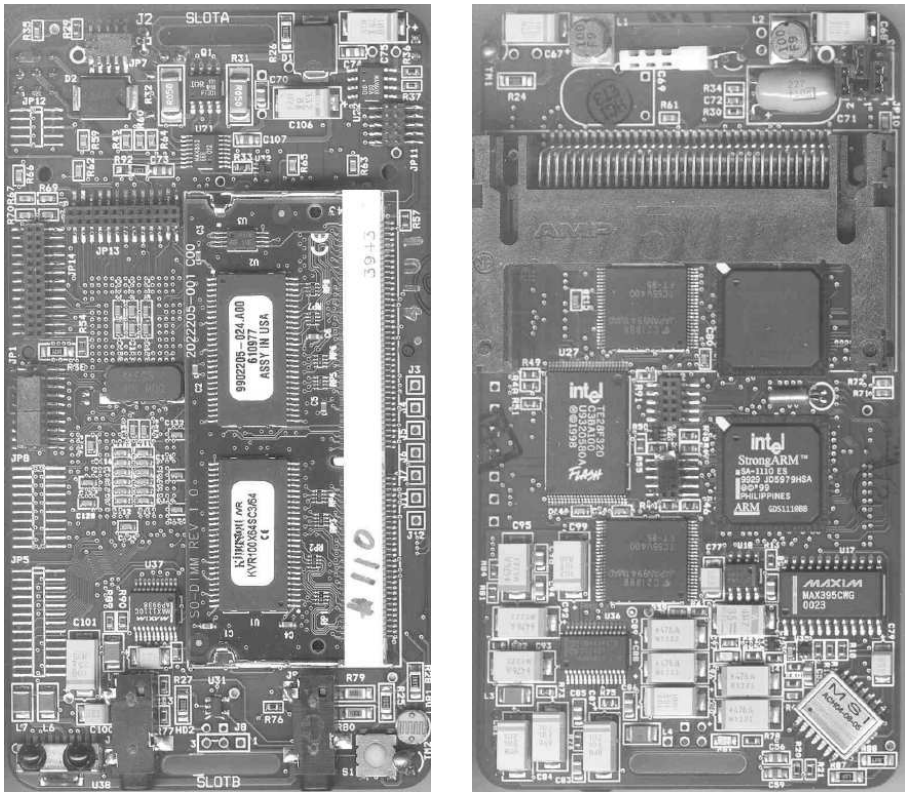
A wearable computer is more capable than a PDA, but it is more bulky and awkward. They are typically worn around the waist. They typically have the processing power of low-end laptop computers, although more capable models exist in the higher price range. They can run popular operating systems such as Windows or Linux. CPUs, disk drives, batteries, and other peripherals are often worn around the waist on a belt. Input devices range from pointing sticks, touch screens, small keyboards worn on the wrist, to voice input. Display devices can be head mounted displays or small LCD panels worn on the arm or wrist. Wearable PCs are also expensive, ranging

anywhere from $5000 to $10,000.

Since they are much like PCs, they are able to run the software and peripherals for desktop and notebook computers, therefore they offer greater flexibility in applications. They can have battery lifetimes that exceed laptops as several batteries can be worn around the waist. They are often used for industrial applications such as aircraft maintenance, nuclear power plant maintenance, manufacturing plants, etc. These locations are generalized by high noise, so a voice interface is not a viable option. However, other uses such as in the medical profession or inventory control, where quiet office environments can be expected, are suitable for VUIs.

1.1.4 HP Labs Smartbadge

The HP Labs Smartbadge IV is a research prototype of a wearable smart identification badge. It is a useful platform for biometric applications, diverse context sensors, audio and speech processing, video streaming, and human-to-human communication. It supports power optimization and measurement as well as core scaling and adaptive power policy generation [38]. Figure 1 shows a photograph of both sides of an assembled Smartbadge IV. The Smartbadge uses a 206 MHz Intel StrongARM (SA1110) processor and SA1111 coprocessor with configurable Flash, SRAM, and SDRAM. It can be equipped with a variety of I/O interfaces including PCMCIA, Compact Flash, USB, Serial, and infrared. It supports CD-quality stereo audio input and output through headphone and speaker jacks. It can be configured for a variety of additional sensors including accelerometers, temperature sensors, humidity sensors, and photo (light) sensors [99]. The lack of user keypads or displays make the Smartbadge an ideal candidate for a speech-only interface. It runs the embedded Linux operating system. The software/hardware is supported by a variety of power measurement tools including a cycle accurate energy consumption simulator and prototype boards prewired for power measurements. Many of the hardware measurements and simulations



(a) Side 1 (b) Side 2

Figure 1: The HP Labs Smartbadge IV

in this thesis are based on the Smartbadge hardware because of its ease of use as a research platform. 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 thesis. Newer PDAs based on the XScale processor have a similar architecture to the StrongARM, and we expect similar results with these processors.

1.2 Applications of Speech Recognition for Mobile Devices

TTS and ASR technology are particularly well suited to task specific applications. The constraints imposed by those tasks can allow for greater accuracy within ASR and greater fluency with TTS. Rabiner suggests some guidelines for the use of speech technology in [81] and [82]:

- The use of ASR and TTS should benefit the user.
- The system should be user friendly in the sense that little or no training is required to use it.
- The speech interface should be accurate.
- The system should responde at or near real-time. Delays should be indicated through auditory "earcons" or visual feedback.
- The system must be able to gracefully handle errors and allow the user to backtrack through the system.
- The system must allow barge-in of audio prompts for expert users.

- The speech interface must coordinate with the graphical user interface where applicable.

Sharman emphasizes the need for speech interfaces on miniature devices such as PDAs and cell phones, but predicts that most applications of speech interfaces will settle out to specific applications such as telephony applications, and database information access [89]. Finally, Tanaka stresses the importance of synergies with the visual user interface through multimodal interaction [103]. Tanaka believes that "the multimodal system will become main-stream in human-machine interface."

There are many systems being built in both industry and academia that address many of the issues involved in VUI technology. Certain groups are working on hardware solutions specific to ASR and TTS [6],[9]. Other groups are concerned with wearable computing and other mobile devices. Standards groups are attempting to provide common interfaces to speech technology. Finally, new application classes involving natural language processing are becoming a reality.

1.2.1 Wearable Computing

The constraints of wearable computing were discussed earlier. There are several academic and industry groups working to speech enable wearable computers [86], [98], [35], [26]. The most pervasive application seems to be inspection database access for industrial technicians. The Portable Maintenance Terminal (PMT), as it is called in [35], gives the technician hands free access to inspection checklists, schematics, troubleshooting information, and data entry through a combination of a head mounted display and speech input/output. The PMT can cut costs, decrease worker fatigue, and increase the quality of work.

A distributed speech recognition system over a wireless link can alleviate the processing requirements for the PMT while providing higher quality for the user.

Carnegie Mellon University's ISAAC system uses this approach. A wireless microphone/headphone system allows the user to issue voice commands to turn on devices, access e-mail, browse the web, etc. The wireless link is a low cost infrared link using RF extenders to go beyond line of sight communication. All ASR and TTS operations take place on a server.

1.2.2 Information and Control in the Home

Home automation is an especially important application for the elderly and disabled. The smart home of the future could look after its occupants who may otherwise need constant care. An audio warning could alert the occupants of dangerous conditions such as an unattended stove. Disabled persons could have access to lights, temperature control, and appliances through voice commands. The X10 home automation modules allow a variety of appliances, air conditioners, and lamps to be easily connected to a PC with very low cost. The home PC acts at the nervous system of the house. Hand-held web pads could give Internet access through wireless links to the PC. The Open Services Gateway Initiative (OSGI) [108] seeks to standardize network interaction throughout the home via a standard interface supported by multiple vendors. Chowdhury and Chia from Intel discuss home automation, Internet telephony, and personal assistant applications in [11]. They feel the the VUI when used properly is a valuable tool to provide easy hands free interaction with devices in the home.

The *smart intercom* project in the Aware Home at Georgia Tech is another example of VUI technology in the home [48]. Speech recognition will allow occupants to locate a particular occupant of the house and open a two way audio connection between them. The occupants themselves will be identified through speaker verification and a positioning system within the home. This hands free intercom poses many problems in speech processing including environmentally robust ASR, speech coding, echo and noise cancellation, and robust speaker verification.

1.2.3 The Automobile Environment

The automobile is a harsh acoustic environment for speech recognition. Wind, engine, and tire noise can vary drastically with speed and vehicle type. In general, automobile environments are characterized by low and highly variable SNR. Close talking microphones, small vocabulary speech recognition, or appropriate noise robustness techniques are required. Automobile applications should be hands/eyes free with minimal displays if any. With many US states passing hands-free mobile phone laws, the use of ear-mounted headsets is becoming more common. As Bluetooth enabled cellular phones hit the marketplace, we are seeing increasing numbers of wireless headsets. Such headsets can be used for functions other than telephone communication such as hands-free voice command and control in the automobile, including navigation and climate control. A Bluetooth enabled headset is a good candidate for a low-power distributed speech recognition system.

Finally, in a system built by Muthusamy *et al.*, a client server approach was employed to perform speech recognition and synthesis in the car [71]. The Java Speech API (JSAPI) [67] was used to provide a standard interface to recognition/synthesis technology. Applications included voice dialing, voice e-mail, and navigation.

1.2.4 Standardized Speech Application Markup Languages

Standardized speech application markup languages provde a framework for the development of voice-enabled applications. Until the advent these standards, building applications, such as interactive voice response (IVR) systems, required a team of skilled programmers familiar with telephony hardware, speech recognition, and text to speech software. A standard markup language hides the speech application designer from the details and software APIs of the underlying technologies. Speech applications written for a particular standard can be executed on any conforming hardware/software platform without modification.

Earlier work on voice browsing attempted to augment HTML with speech specific tags since HTML does not naturally lend itself to speech interfaces [15]. The World Wide Web Consortium continues this work with the XHTML+Voice profile, which specifies speech tags that can be added to HTML documents [13]. A browser can interpret or ignore these tags depending on its capability.

A standard was release in March, 2000 called the Voice eXtensible Markup Language (VoiceXML) [111]. XML is the standard format for documents and data on the Internet. VoiceXML uses the XML syntax to allow speech enabled applications to be created and placed on centralized servers much like HTML pages. Voice browsers can access the VoiceXML applications and interpret them in a platform independent manner. Danielson and White give some overview of the language in [17] and [114] respectively. Originally developed for telephony applications, VoiceXML has no support for a visual interface. It supports TTS and digitized audio output and DTMF and speech input. The language shields application developers from the difficult issues involved in speech interfaces. Finite state grammars are used to constrain the allowed input phrases for better speech recognition accuracy. A new version of the standard was released in February, 2002 and is rapdily gaining acceptance in the industry.

1.2.5 Towards Multimodal Interaction

The VoiceXML standard does not support any multimodal interaction. Multimodal systems couple speech interaction with other input/output methods such as keyboard, pointing device, or touch screen. A scalable multimodal system could provide a common interface to data across a variety of different platforms from telephones and PDAs to desktop computers. A multimodal interface can be a more efficient medium than a voice only interface. For example, TTS is serial in nature while text is random access and allows for greater bandwidth between the user and machine. In addition, speech input is very efficient while text based menus can become cumbersome and

slow to use. A voice-in web-out type of user interface can capture the best of both interaction methods on a small PDA-like device.

Multimodal input can be either redundant or complementary. An example of complementary input is a user pointing to an icon of a product while saying, "I want two of these." Redundant input is when the user uses multiple methods to simultaneously to indicate the same action. For example, circling an icon of a green shirt with a stylus while saying, "I want the green shirt." Complementary input can increase the total bandwidth between the user and machine by allowing separate tasks to be completed at the same time, while reduntant input can increase the total accuracy of the system. Mills and Alty did a study on redundancy in multimodal input systems in [66]. By using both speech and a pointing device, the machine can disambiguate the input when one channel becomes noisy if redundant input is used. For example, a user can say "I want to buy widget S" or "I want to buy widget F". Using telephone quality speech, a human would have difficulty disambiguating the utterance. However, with the addition of a simple click or circling some icon representing widget F, the system can easily tell which one the user meant.

Finally, the Speech Application Language Tags (SALT) is an emerging standard whereby visual and audio input/output are combined through a standard language [105]. Unlike VoiceXML, SALT allows some multimodal interaction and is targeted more towards PDA's, high-end cellphones, or other mobile wireless devices with visual display components.

1.2.6 Natural Language Systems

An ultimate goal of the VUI is to use natural language to interact with the system. An example is the work done by Lee in [55] on natural language call routing. The front end to the system was a speech recognizer with about 24% WER for telephone quality speech. However, by training a system to route calls based on natural language

queries, the system only misrouted around 10% of the calls. Olive also suggests a task driven recognition system, where the most probable task T_i is chosen from a set of acoustic features [72]. This is a difficult problem and current systems need to be trained for the specific task using large amounts of training data. Unlike the standard markup languages, building and designing natural language applications requires domain expertise on the part of the application developer.

1.3 Contributions, Outline, and Scope

The central theme of this thesis is the rigorous study of a multimedia client for pervasive wireless media applications. Speech recognition is considered as one such application, where the computational demands have hindered its use on wireless mobile devices. Our analysis considers distributed speech recognition on hardware platforms with PDA-like functionality (i.e. wireless LAN networking, high-quality audio input/output, a low-power general-purpose processing core, and limited amounts of flash and working memory.) We focus on quality of service for the end-user (i.e. ASR accuracy and delay) and reduced energy consumption with increased battery lifetimes. We present the following technical contributions in this context:

- A software-based signal processing front-end optimized for low-power, general-purpose processors present in many mobile multimedia devices [18, 19].
- A characterization of arithmetic requirements for small vocabulary ASR on embedded systems.
- Power management of 802.11b and Bluetooth wireless interfaces to reduce energy consumption with distributed speech recognition traffic while maintaining quality of service for the end-user [20].
- A discussion of tradeoffs and energy saving opportunities for client-side and distributed automatic speech recognition in wireless LAN environments including

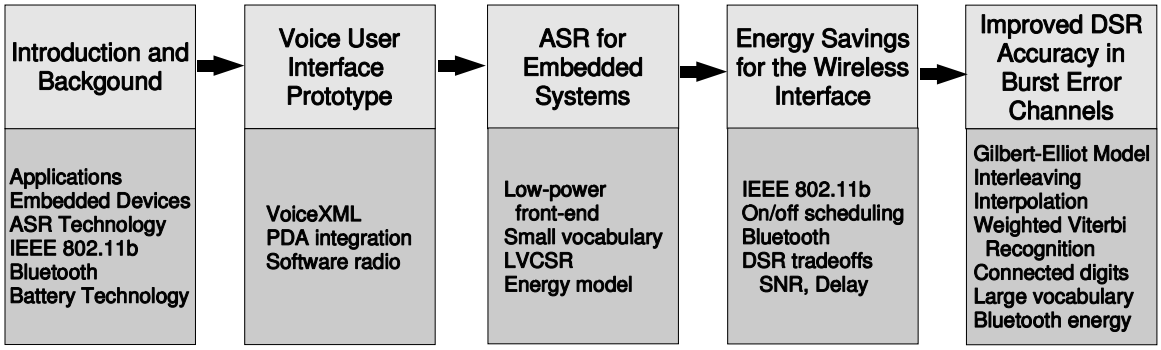


Figure 2: Thesis outline.

channel quality, battery energy, and delay characteristics [21, 22].

- A novel algorithm for distributed speech recognition accuracy improvement in burst error channels.

We consider energy consumption as applied to the HP Labs Smartbadge IV embedded system. The Smartbadge has support for both hardware measurements and software simulation to determine energy consumption. We consider local area wireless networks such as IEEE 802.11b and Bluetooth networks. Single hop communication is assumed with the typical mobile host/base station scenario. Simulations involving distributed speech recognition make use of portions of the ETSI DSR standard where applicable [109]. Improved methods for quantization of speech features for distributed speech recognition exist, but our simulations deal mainly with loss protection and energy consumption issues. Therefore, it is sufficient to use portions of the ETSI standard without loss of generality.

Figure 2 shows a general outline for this thesis. The majority of the work focuses on the wireless multimedia client. In Chapter 2 we present a brief technology overview, including automatic speech recognition, distributed speech recognition, wireless networks, and battery technology. In Chapter 3, we introduce a prototype implementation of a voice user interface for a multi-modal Web browsing application

on a PDA. Chapter 4 includes a low-power front-end feature extraction, a complexity analysis of small vocabulary connected word recognition, and a literature review of large vocabulary speech recognition performance on modern computers. Some current issues hindering the implementation of high-quality large vocabulary speech recognition on mobile devices are discussed. In Chapter 5, we present a power on/off scheduling algorithm to reduce the energy consumption of the wireless interface and discuss tradeoffs pertaining to energy consumption and quality of service for both data and multimedia traffic types. We present a novel loss protection technique for distributed speech recognition in burst error channels in Chapter 6. Portions of this technique require operation at the ASR server, but the improvement in channel robustness can be translated to energy savings for the mobile client.

CHAPTER II

BACKGROUND

In this chapter, we give a brief overview of the technologies discussed in this thesis. In section 2.1, we describe the current state-of-the-art speech recognition systems based on the hidden Markov model. In section 2.2, we introduce distributed speech recognition. In addition, we give a brief introduction to the IEEE 802.11 and Bluetooth data networks used throughout this thesis in sections 2.3 and 2.4. Finally, we discuss current limitations in battery technology in section 2.5.

2.1 Automatic Speech Recognition

Research on automatic speech recognition has spanned about forty years. It started with recognition of isolated digits for a single speaker in 1952 at Bell Laboratories [80]. Despite this long history, the problem of automatic speech recognition by a machine is far from being solved. Large vocabulary conversational speech recognition remains a difficult task, and current systems have difficulty with high levels of background noise even with small vocabularies. However, despite these imperfections, specific applications of ASR are finding their way into the marketplace.

In ASR, we need to choose the most probable word sequence from some set of acoustic observations. The following is the Bayesian approach to ASR discussed in [80]:

$$\hat{W} = arg\max_{W} P(W|\mathbf{O}) = arg\max_{W} P(\mathbf{O}|W)P(W) \tag{1}$$

where $\mathbf{O}$ is some acoustic data, $\hat{W}$ is the estimated word sequence, and W represents all possible word sequences. It is not convenient to estimate $P(W|\mathbf{O})$ since the total number of word sequences is effectively infinite. However, we can create stochastic

models of word sequences using training data and then estimate $P(\mathbf{O}|W)$, the probability that a particular set of acoustic observations comes from some model of a word sequence. The probability of individual word sequences, $P(W)$, can be estimated by analyzing large amounts of text data or, for more constrained input, built by hand through a series of grammar rules.

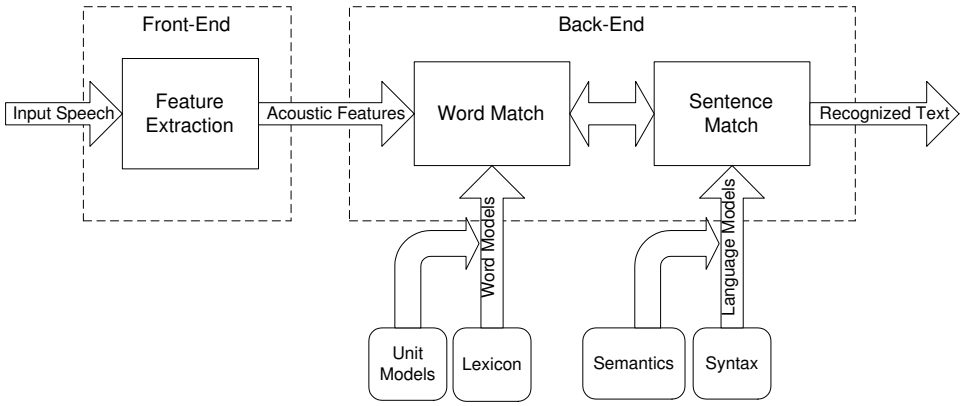


Figure 3: Architecture of a typical automatic speech recognizer [82].

Figure 3 shows the fundamental blocks of a typical automatic speech recognizer. Some digitized speech is analyzed using signal processing algorithms to produce a vector of acoustic observations, $\mathbf{O}$. These features are compared with some acoustic models stored on disk. Individual words are concatenated from smaller units using a lexicon. In addition to acoustic models, a language model which estimates $P(W)$ is used to provide context and increase accuracy. The next few sections will explain each of the various parts in Figure 3 in some detail.

2.1.1 The Signal Processing Front-End

The purpose of the signal processing front-end is to produce acoustic features which can be used to recognize speech. A good signal model will produce *perceptually meaningful*, *invariant*, and *uncorrelated* parameters. They should be perceptually meaningful in that they represent aspects of the speech signal used by the human auditory system. Robustness to noise and variations in channels and speakers make

them invariant. Uncorrelated features are not redundant in the information they convey. More features require more computation in both estimation and evaluation, so it is essential to keep the feature set to a minimum while still providing sufficient accuracy.

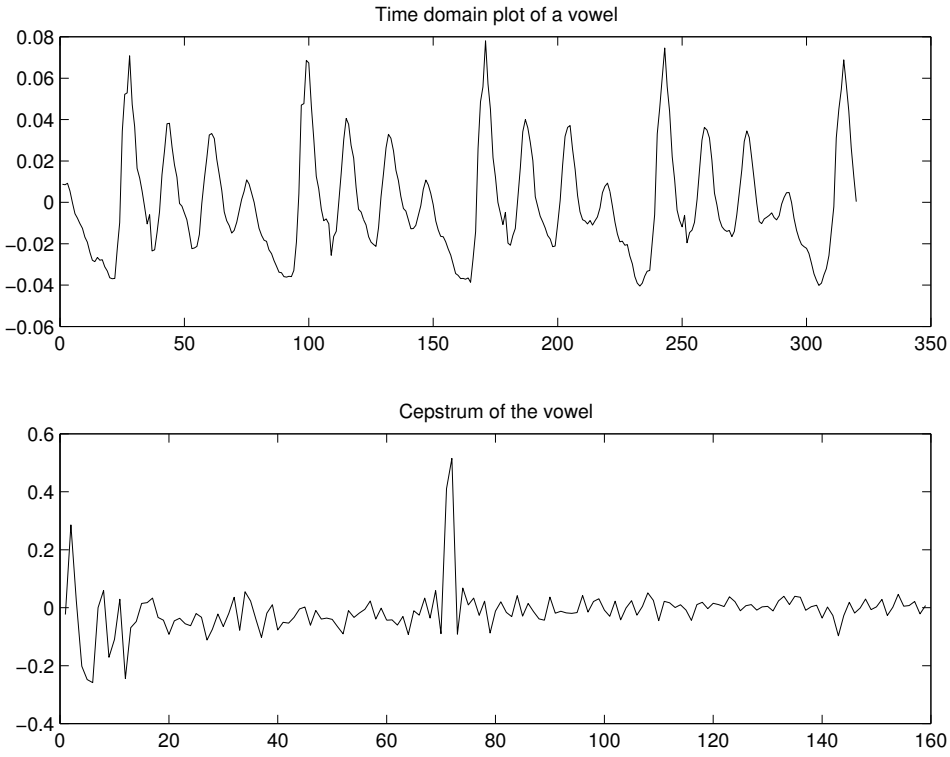


Figure 4: The real cepstrum of a voiced segment of speech.

A homomorphic signal processing technique known as the cepstrum is used by many state-of-the-art speech recognition systems to date. The cepstrum is defined as the inverse Fourier transform of the log spectrum:

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

where $S(\omega)$ is the spectrum of the speech signal. The cepstrum effectively separates the glottal excitation signal from the vocal tract parameters. In the frequency domain, the slowly varying vocal tract and quickly varying excitation signal are multiplied to produce the speech spectrum. However, if we take the logarithm of both spectra, then

they are linearly combined in the log-spectral domain. By taking the inverse DFT of the log-spectrum, the first coefficients in the cepstrum represent the slowly varying vocal tract parameters, and the remaining coefficients model the quickly varying excitation signal and pitch (see Figure 4). The slowly varying parameters in the log spectrum (i.e. the spectral shape) will occupy the first part of the cepstrum or lie at the beginning of the *quefrency* scale. The quickly varying parameters (i.e. the glottal excitation) will occur further down on the quefrency scale, with a large peak occurring at the pitch frequency. In speech recognition, we are usually only interested in the first 13 cepstral coefficients, which give information about the vocal tract shape. The vocal tract parameters are influenced by the location of the articulators (lips, tongue, etc.). By separating the effects of the glottal excitation, the cepstrum is less affected by variations between and within speakers.

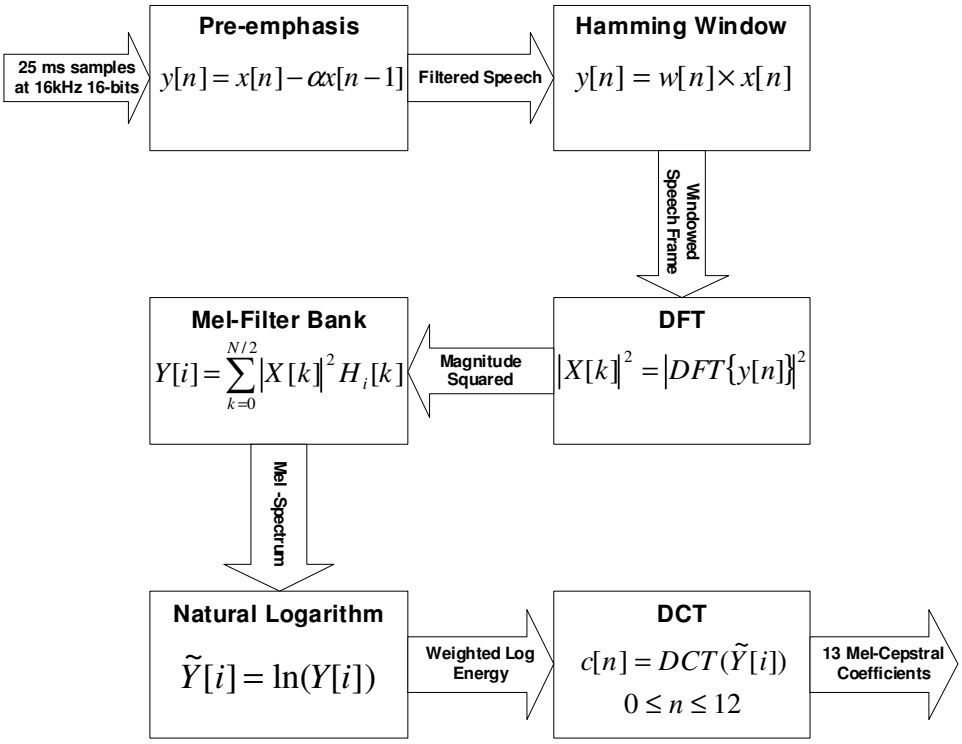


Figure 5: The algorithm used to compute the mel-cepstrum.

An extension to the cepstrum that is used with much success is the mel-frequency cepstrum. The mel-frequency scale is used to model the perceived pitch of a tone. It

is a non-linear, empirically determined, frequency scale that gives more emphasis to lower frequencies. It can be estimated by the following formula:

$$F_{mel} = 2595 \log \left[1 + \frac{F_{Hz}}{700} \right] \tag{3}$$

In practice, the mel-frequency cepstral coefficients can be computed using the algorithm in Figure 5. 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:

$$Y[i] = \sum_{k=0}^{N/2} |X[k]|^2 H_i[k] \tag{4}$$

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.

Most state-of-the-art speech recognizers currently use 13 mel-frequency cepstral coefficients as a base feature set. Overlapping, windowed, frames of approximately 20-30ms are computed at a rate of one every 10ms. Secondary features, consisting of first and second temporal derivatives of the cepstrum, are also used, resulting in a 39 dimensional feature set generated 100 times per second. These secondary features account for spectral dynamics that the acoustic models may not be able to model explicitly. A more in depth discussion of the theory and properties of the cepstrum can be found in [23]. An account of other signal processing techniques used in speech recognition can be found in [74] and [41].

2.1.2 Acoustic Modeling

In acoustic modeling, the goal is to produce models of the acoustic observations from the signal processing front-end that are accurate, trainable, and generalizable. In small vocabulary systems, accurate models of individual words can be trained without difficulty. It is easy to obtain sufficient training data for whole word models for such a small vocabulary. However, such a model is not generalizable because new words cannot be added to the vocabulary without additional training. Also, as the vocabulary size grows, it is difficult to get enough training data to accurately model the acoustics of every single word. Since word models are no longer appropriate, the recognition unit needs to be broken up into something smaller.

Individual phonemes do not work well in practice because there is too much variation due to articulatory effects from adjacent sounds. Context dependent units such as the triphone are the standard recognition unit for most large vocabulary applications. A triphone is an acoustic unit which models the effects of the preceding and following phoneme. However, the total number of triphones is about 80,000 compared to around 50 base phonemes. In practice, the total number of triphones used in a speech recognizer is smaller than this since many triphone combinations occur very infrequently or never at all. Since many triphones share parts which are acoustically indistinguishable, parameters are often shared between them. This aids both in training and decoding. Cross-word triphones are used to model the articulatory effects of adjacent words. Cross-word triphones give better accuracy, but they add complexity to the system. There has been some research emphasis on using syllables as the recognition unit since there are only around 11,000 syllables in the English language [31]. Early results gave a marginal improvement in word error rate (WER). A lexicon will list the phonetic representations of words using whichever phonetic unit is chosen for the recognizer. Words not in the lexicon can be estimated using letter to phoneme rules, so an acoustic model based on context dependent sub-word units

is generalizable. New words can be added by concatenating the sub-word models.

2.1.2.1 The Hidden Markov Model

Many acoustic modeling techniques have been proposed for automatic speech recognition. Among these are Dynamic Time Warping, Artificial Neural Networks, and Stochastic Segment Models [45]. Perhaps the most successful and widely used method for acoustic modeling is the Hidden Markov Model (HMM). The HMMs popularity can be attributed to its simple structure, straightforward implementation, and good performance [44]. The HMM can best be described as a probabilistic state machine for the study of time series. In speech recognition the time series is given by an observation vector $\mathbf{O} = (\mathbf{O}_1, \mathbf{O}_2, \ldots \mathbf{O}_T)$, where each $\mathbf{O}_i$ is some acoustically meaningful vector of speech data for the *ith* frame. HMMs are Markov chains whose state sequence is *hidden* by the output probabilities of each state.

A Markov chain models a class of random processes with a minimum of memory [41]. Such models are often represented as state machines where the transitions between states are described by the random processes. A first-order Markov chain is based on the assumption that the probability of the Markov chain being in a particular state is dependent only on the state of the chain at the previous time. Let $\mathbf{X} = X_1, X_2, X_3, ...X_n$ be a sequence of random variables from some finite alphabet of observations, $\mathbf{O}$. Under the first-order Markov assumption and the application of Bayes' rule:

$$P(X_1, X_2, X_3, ..., X_n) = P(X_1) \prod_{i=2}^{n} P(X_i|X_{i-1}) \tag{5}$$

The probability of a random variable at a given time depends only on the value at the preceding time ($P(X_i|X_{i-1})$). Associating each X_i with a state, we have $P(X_i = s_i|X_{i-1} = s_{i-1}) = P(s_i|s_{i-1})$, where s_i is the ith state of the system. A stationary Markov chain models time-invariant events, where the time index, i, is independent of the state sequence.

In the observable Markov model, each state outputs an event that can be directly associated with that state. Therefore, the state sequence is always known with an observable Markov model. However, the hidden Markov model outputs an event according to another stochastic process, or output distribution, within each state. By simply observing events, it may not be possible to determine the exact state sequence since multiple states may output the same events. However, given the state output distributions and transition probabilities, it is possible to determine the most probable state sequence. HMMs can use either discrete or continuous output probabilities.

Consider a HMM with N states indexed as $\{s_1, s_2, \ldots, s_N\}$, a state, s_k, contains an output probability distribution, B, which describes the probability that a particular observation is produced by that state (i.e. $p(\mathbf{O}_i|q_t = s_k)$, where q_t represents the state at time t.) B can be either discrete or continuous. Continuous probability density functions add complexity but are more robust to parameter variations. Gaussian mixture densities are used to approximate any finite, continuous density function, which adds robustness to noise and speaker variation. A Gaussian mixture density for a particular HMM state is a summation of scaled Gaussians:

$$b(\mathbf{o}) = \sum_{k=1}^{M} c_k \mathcal{N}(\mathbf{o}, \mu_k, \mathbf{U}_k)$$

where $\mathbf{o}$ represents the observation vector being modeled, c_k is the mixture weight, $\mathcal{N}$ is the multi-variate Gaussian pdf with μ_k as the mean and $\mathbf{U}_k$ as the covariance matrix. The parameters for the mean, covariance, and mixture weights for each of the mixtures are estimated during training, and the number of mixtures, M, is a design parameter. A *discrete* observation probability requires that observations be chosen from some finite alphabet via vector quantization. Discrete HMMs can have a speed advantage over continuous output density HMMs since a significant amount of processing is involved in the Gaussian mixture evaluation.

Each model contains a transition matrix, A, which describes the transition probabilities from one state to another based on the first-order Markov chain assumption

(i.e. $a_{ij} = P(q_{t+1} = s_j | q_t = s_i)$). For example,

$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \tag{6}$$

could describe the state transition matrix for a three state HMM. The probability for going from state 1 to state 2 would be 0.3. Figure 6 shows a typical HMM used in speech recognition where the state transitions are limited such that only left to right movement of no more than two states is permitted. Other topologies are possible, and this is generally a design decision. In general, more non-zero probability state transitions in a model topology will result in higher decoding costs. Finally, the HMM has an initial state distribution, π, which describes the probability of starting in any one of the N states. For convenience in notation, the HMM parameters can be written as $\lambda = (A, B, \pi)$.

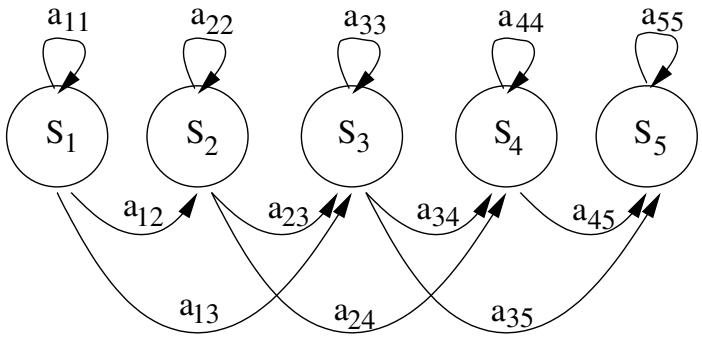


Figure 6: A left-to-right HMM showing transition probabilities.

For speech recognition, we are primarily interested in the $P(\mathbf{O}|\lambda)$, the probability that a speech observation, $\mathbf{O}$, came from a given model, λ. If λ represents a string of words, then we have $P(\mathbf{O}|W)$ as discussed earlier. These model parameters are estimated through training with a set of labeled and transcribed speech data. The

training attempts to maximize $P(\mathbf{O}|\lambda)$ by manipulating the model parameters, λ. Efficient techniques for model estimation include the Viterbi algorithm and Baum-Welch estimation discussed in [80], [44], and [75].

2.1.3 Language Modeling

Using some fundamental acoustic unit (phonemes, triphones, syllables, etc.), we can obtain a set of hypothesized words for a particular set of observations. However, using acoustic models alone will yield poor recognition accuracy because there is too much acoustic similarity between many words and phonemes. Language models are used to estimate the probability of word sequences and thus provide better accuracy. Language models are typically based on the N-gram model, which estimates the probability of the next word on the previous $N-1$ words. For an arbitrary N, the probability is:

$$P(w_i|w_{i-N-1}, w_{i-N-2}, ..., w_{i-1}) \tag{7}$$

In practice, trigrams, $N = 3$, and bigrams, $N = 2$, are most commonly used. Bigrams contain one word history, $P(w_i|w_{i-1})$, and trigrams contain a two word history, $P(w_i|w_{i-2}, w_{i-1})$. Unigrams contain no word history, $P(w_i)$. Bigrams and trigrams for a particular task are difficult to estimate without large amounts of training data.

When combining acoustic and stochastic language models, it is desirable to include a weighting on the language model. The acoustic probabilities are typically underestimated, and a direct multiplication of the two models, as in (1), often yields a probability estimate that favors the language model too much. In practice, the following weighting is applied:

$$\hat{P}(\mathbf{W}) = P(\mathbf{W})^{LW} \cdot IP^{N(W)} \tag{8}$$

where $\hat{P}(\mathbf{W})$ is the new N-gram probability, LW is the weighting applied to the language model, IP is the word insertion penalty, and $N(W)$ is the number of hypothesized words in the current Viterbi search path. The weighting, LW, is typically > 1,

and this has the effect of reducing the overall contribution of the N-gram probability. The IP is generally between zero and one, and it has the effect of reducing the tendency of the recognizer to favor longer strings of short words. Both LW and IP can be determined emperically for a given task.

For more constrained speech input, we can use finite-state grammars (FSG) which compute phrase probability based on a set of grammar rules. These rules generally give a set of choices useful for command and control applications. At the risk of limiting what the user can say, we can achieve greater accuracy with lower complexity. The VoiceXML standard is designed for use with this form of grammar. There are a few standard representations for this format, including the Speech Recognition Grammar Specification (SRGS), Backus-Naur format and the Java Speech Recognition Grammar Format (JSGF). These grammars are typically built by hand which is a time consuming processes often requiring much tweaking and tuning for real-world applications.

Certain tasks allow any word to follow any other word, such as digit recognition. These tasks have no grammar and rely only on the quality of the acoustic model. Words that are acoustically similar are difficult to recognize in this case because there is no context information to discern them.

2.1.4 Speech Recognition Decoding

Given a set of acoustic observations, an acoustic model, and a language model, how do you find the most likely word sequence? For each frame of input, all HMM states for all words must be evaluated using the acoustic observations. Language model probabilities must also be included when transitioning between words. An efficient search algorithm is needed to carry out these calculations. Speech recognition search is often called decoding, which is borrowed from information theory terminology for choosing a likely set of output symbols over a noisy channel.

The most common decoding algorithm in speech recognition is the time-synchronous Viterbi beam search. It is time-synchronous in that each frame of speech can be consumed as soon as it becomes available. If the search can consume feature vectors at a rate of 100 frames per second, then it can perform real-time recognition.

The Viterbi search is a dynamic programming technique used to find the optimal path through a trellis. Figure 7 shows a typical trellis used in speech recognition. It is a composite HMM where all states in all words are evaluated for each frame of speech input. The solid lines show the non-zero state transitions for the acoustic models. The HMM topology used in Figure 7 only allows left-to-right transitions of one state or a self-transition. The dotted lines represent null transitions between words. These transitions do not accumulate a frame of input, but they do have probabilities associated with them from the language model. Null arcs only occur at word ending states and only transition to word beginning states.

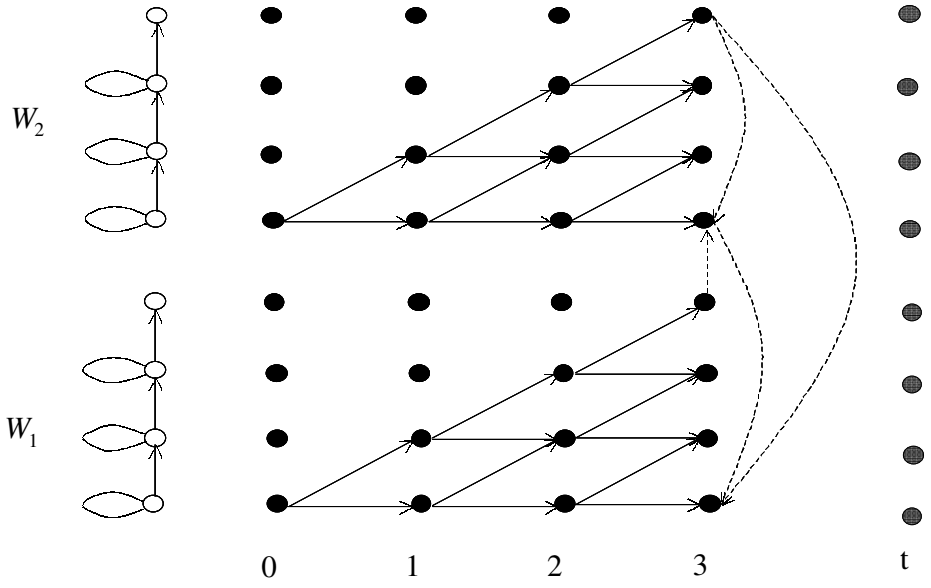


Figure 7: A composite HMM used in Viterbi search [41].

The beam pruning heuristic removes unpromising hypotheses from the search. The cost of the best path is determined, and a threshold is set at some smaller cost by subtracting the beam width from the best path cost. All paths that fall below this

threshold are removed from the search. Beam pruning can occur at different levels as in the hierarchical search presented in [24].

2.1.5 Accuracy and Complexity

A block diagram of a complete continuous speech recognizer is shown in Figure 3. It contains all of the components mentioned thus far (feature extraction, word matching, and language modeling). What type of performance can we expect from this type of recognizer in terms of memory, CPU time, and accuracy?

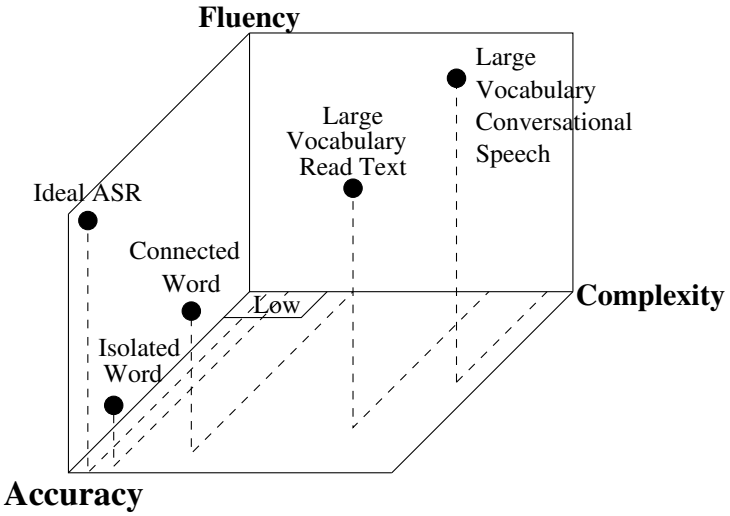


Figure 8: Fluency and accuracy vs. complexity for ASR.

Accuracy is largely a function of the task and environment under which the recognizer is running. Noise will certainly degrade performance. Small vocabulary tasks perform better than large vocabulary tasks. Speaker independent systems generally have higher error rates than speaker dependent systems. Figure 8 shows the tradeoffs between accuracy, fluency and complexity. Fluency is a measure of the ability to recognize speech from a variety of different vocabularies, speaking styles, and speakers. The accuracy drops and complexity increases as we allow the system to be more fluent.

Table 1 shows some basic word error rates for selected speech recognition applications. WER drops to worse than 10% as soon as the transition from large vocabulary read speech to large vocabulary spontaneous speech is made. These measurements are made under laboratory conditions with clean speech and little mismatch between test and training sets. Significantly worse performance can be expected from fielded systems. Sentence error rates are another measure of performance.

Table 1: Typical word error rates for speech recognition applications [82].

Speech Corpus	Type	Vocabulary Size	WER
Connected Digit Strings	Spontaneous	10	0.3%
Airline Travel Information	Spontaneous	2,500	2.0%
Wall Street Journal	Read Text	64,000	8.0%
Radio	Mixed	64,000	27.0%
Switchboard	Conversational Telephone	10,000	38.0%
Call Home	Conversational Telephone	10,000	50.0%

Finally, what sort of requirements are needed to perform large vocabulary speech recognition in terms of memory and CPU power? The answer is largely dependent on the details of the recognition system and the accuracy desired. Rabiner suggests that upwards of 1 gigaflops may be needed for real-time operation in certain complex tasks [81]. Other authors estimate 200 MIPS for continuous speech recognition in simpler tasks [11]. "WER and the maximum amount of memory used both vary exponentially with real-time performance. For example, a 50% reduction in error rate requires an increase in performance by a factor of three" [24, pg. 101]. Table 2 shows the computational requirements for the ISIP speech recognizer. The ISIP system is a research system, and as a result, the speed is lower than that of a commercial system. However, significant improvements in speed have been made since [24] was published. The authors believe they can transform the system into one which performs at ten times real-time with a minimal reduction in WER. Regardless, it is easy to see that the processing requirements for speech recognition are significant and that even small

increases in accuracy can require large increases in computation time.

Table 2: Memory and CPU requirements for the ISIP speech recognition system. Baseline system is a Pentium 333MHz with 512MB RAM [24].

	Connected alphanumeric digits			Large Vocabulary Spontaneous Speech		
Recognition Unit	WER	Mbytes	xRT	WER	Mbytes	xRT
Context-independent phones	17.5%	18	3.3	N/A	N/A	N/A
Context-dependent word-internal phones	13.6%	24	4.3	52.6%	220	240
Context-dependent cross-word phones	9.4%	41	9.6	48.7%	300	470

2.2 Distributed Speech Recognition

Given that complex speech recognition tasks can be computationally intensive for portable devices, the wireless network can be used to offload all or part of the speech recognition task. This has become known in the literature as distributed speech recognition (DSR). Recent research on DSR can be lumped into several categories: software issues, DSR over existing cellular networks, direct quantization of speech features, and error correction and concealment techniques.

2.2.1 Software Issues

A framework to enable speech recognition as a distributed service is presented in [63]. The authors propose an efficient server arhcitecture that can be built into the network to handle speech recognition requests. The server runs on the QNX real-time operating system and handles multiple speech recognition requests in an efficient manner. Clients are guaranteed timely responses through the appropriate use of multi-threading and parallelism. In [88], a CORBA based distributed computing model is presented. CORBA, the Common Object Request Broker Architecture, is a

distributed computing standard for object oriented languages. In this work, multiple sensors and media recognition applications are interconnected to provide robust access to data and services. The use of CORBA allows for a common framework across the various software to enable distributed computing in a simple and manageable manner. Finally, in [122], another distributed speech recognition software system is presented. It focuses on the archtecture design, implementation, and optimization of DSR systems. Issues such as bandwidth and network protocols, scalability and load balancing of servers, and paralellism of servers are discussed.

2.2.2 DSR Over Existing Digital Cellular Telephone Networks

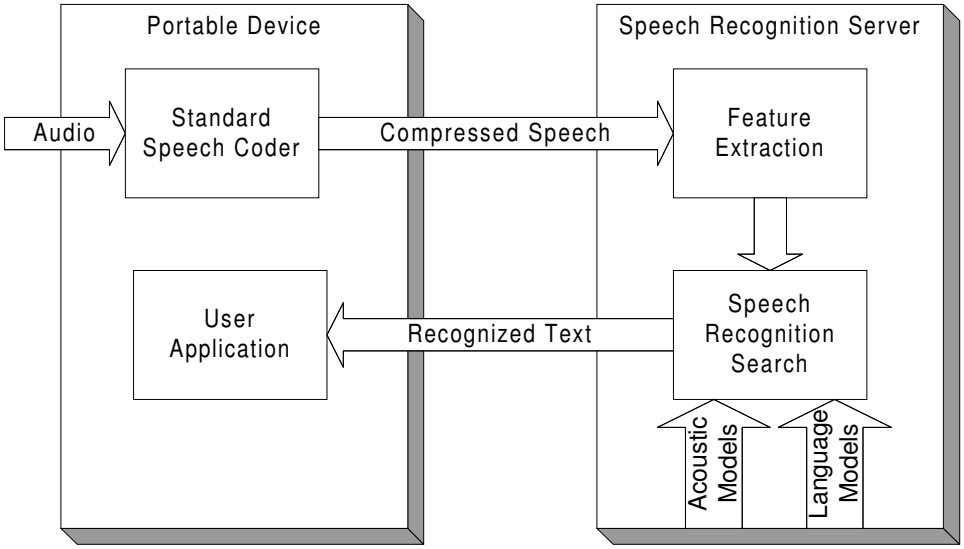


Figure 9: Distributed speech recognition over digital cellular telephone networks.

Digital cellular telephone networks use standard speech coding techniques to reduce the bit rate before transmission. In Figure 9, the speech is compressed on the mobile device and sent to across the network where it eventually gets to a speech recognition server. While this may seem like the ideal solution, speech coding schemes (especially low bit-rate ones) alter the spectral content of the speech signal. While this distortion is intended to be unnoticeable by a human, the added noise and distortion effects the speech recognition system [58]. One solution is to train the speech

recognizer using only coded speech, and this method is used with some success for speech recognition over cellular telephones. Another solution is to compute the speech features directly from the coded bit-stream without reconstructing the speech signal as in [49], [50], and [76]. These techniques have been able to match the recognition performance attainable with toll quality speech (i.e. land-based telephone systems.) However, the bitstream contains a set of quantized spectral parameters and some parametric representation of the residual signal (i.e. codebook index). Since the spectral analysis is imperfect and the spectral components are aggressively quantized, this residual signal helps recover some of the spectral distortion for a human listener after reconstruction. However, since these bitstream-based front-ends discard the residual signal, they work with an especially noisy set of parameters due to the quantization.

Another consideration when using standard speech coders in a DSR scenario is that the speech coders and the channels they typically operate over are intended for real-time communication. Packets must be delivered with minimal delay, and losses must be concealed to maintain continuity of the signal for a human listener. A speech recognition system has different requirements than a human listener. Packet delay, within limits, is tolerable as the speech recognizer can simply wait until error prone packets are retransmitted. Incoming packets can be stored in a buffer until the ASR system catches up. These issues regarding traffic types can be effectively handled at the MAC layer. For networks supporting real-time traffic, errors in the coded speech parameters can cause significant reductions in speech recognition accuracy [5]. Some techniques for dealing with packet loss over real-time wireless communication channels are given in [112], [68], [46], [47] and [51].

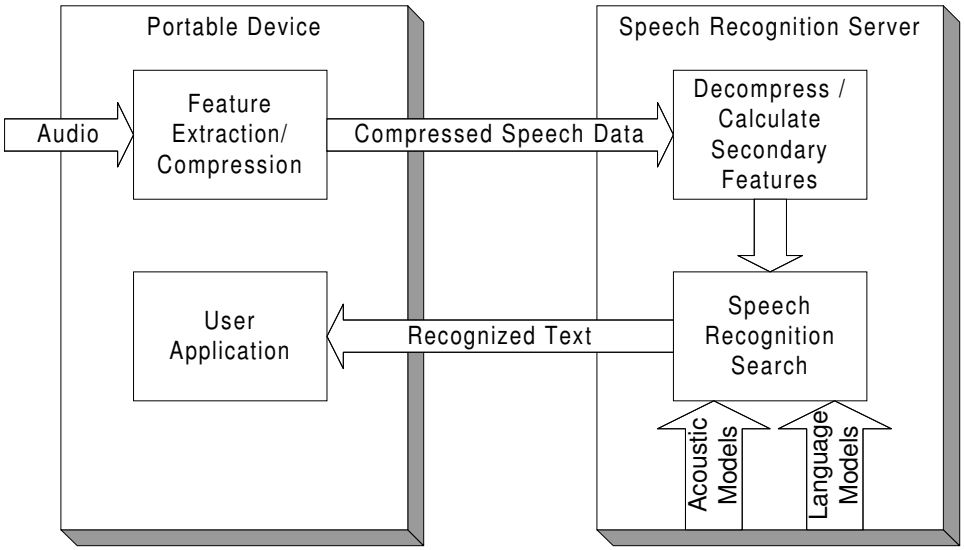


Figure 10: Distributed speech recognition over wireless data networks using compressed speech features.

2.2.3 DSR over Wireless Data Networks

Wireless data networks include the 802.11 family, the Bluetooth personal area network, and others. There may be provisions for multimedia traffic, but these networks are primarily intended for data traffic. We assume that there is no human listener in this category, therefore the client will perform feature extraction for ASR and only transmit coded feature vectors. This scenario is shown in Figure 10. Typically the primary features are calculated and compressed and any additional features such as first and second temporal derivatives are calculated at the server.

Many algorithms have been proposed for quantization of speech recoginition feature vectors. A multi-stage vector quantization technique was proposed in [84]. A one-step predictor is coded along with the prediction error vector. A single 13-dimensional tree-structured vector quantizer was used for each stage. The system operated at 4.0 kbps and provided minimal degradation in accuracy under a variety of large and small vocabulary tasks. A product code vector quantization technique was used in [25], where two or three adjacent cepstral coefficients were quantized together rather than all 13 at once. This exploits the correllation between adjacent cepstral parameters.

The vector quantization search is greatly reduced because smaller codebooks of fewer dimensions can be used. A range of bit allocations and bit rates were presented (from 10.4 kbps to 2.6 kbps), with lower accuracy resulting from lower bit rates.

A scalable coding algorithm was introduced in [100], where entropy constrained scalar quantization was used followed by uniform scalar quantization on the residual error. The result was then processed by a Huffman entropy coder to further remove redundancy. A variety of bit rates can be obtained by this system by varying the quantization levels. In [123], a two dimensional DCT is used to compress speech feature vectors. A twelve by twelve block of speech vectors (twelve vectors by twelve frames) is transformed via a DCT. The result is then quantized and entropy coded, similar to the JPEG image compression standard. An attempt to standardize DSR front-ends is presented in, [109], the ETSI Aurora speech processing, transmission and quality aspects. The ETSI standard uses a product code vector quantizer, specifies the bitstream framing, and provides error detection through cyclic redundancy checks (CRC) on frame pairs. It operates at 4.8 kbps. Perceptual linear prediction coefficients were quantized as low as 400 bps with minimal reductions in accuracy on a digit recognition task in [36]. It is not likely that this aggressive coding scheme will hold up under more complex speech recognition tasks. In [7], the differences in intra-frame vs. inter-frame quantization are discussed. By exploiting the corellation between consecutive frames of speech inter-frame vector quantization can operate at lower bit rates with less degradation in accuracy than intra-frame vector quantization.

2.3 IEEE 802.11 Wireless Data Networks

In 1997, the first wireless ethernet standard, IEEE 802.11, was adopted. It supported a relatively modest data rate of 2 Mbps via a direct sequence spread sprectrum (DSSS) or frequency hopping spread spectrum (FHSS) technology in the 2.4 GHz frequency range. Because of its initially high price tag and low bit rates compared

to wired Ethernet, the 802.11 standard had limited success. Two years later, the 802.11b standard was released. It supported a significantly higher 11 Mbps data rate. Currently, the 802.11b standard is inexpensive and widely deployed. The 802.11a standard ventured into the 5 GHz frequency range with a maximum data rate of 54 Mbps through orthogonal frequency division multiplexing (OFDM). However, because of the change in radio spectrum, the 802.11a standard is not backward compatible with existing 802.11 or 802.11b networks. A more recent standard, 802.11g, employs OFDM in the 2.4 GHz range with a maximum data rate of 54 Mbps. It maintains backward compatability with existing standards. However, there is some concern over the "pollution" of the 2.4 GHz frequency band from various emerging technologies operating in these frequency ranges. Table 3 summarizes the IEEE 802.11 family of wireless networks. The 802.11b standard is widely deployed, well studied in the literature, and sufficient for low-bandwidth speech data, therefore we will consider operation on 802.11b for most of this thesis.

Table 3: A summary of IEEE 802.11 networking standards.

Standard	Freq. Selection	Spectrum	Max Data Rate
802.11	FHSS & DSSS	2.4 GHz	2 Mbps
802.11b	DSSS	2.4 GHz	11 Mbps
802.11a	OFDM	5 GHz	54 Mbps
802.11g	OFDM	2.4 GHz	54 Mbps

All 802.11 standards will operate in either an ad-hoc or infrastructured topology. In ad-hoc mode, two or more wireless stations will communicate with each other over temporary peer-to-peer connections. Communication outside of this ad-hoc network is not possible. In infrastructure mode, a wireless access point is used to provide connectivity to a wired network such as a corporate network or the internet. The range that 802.11b supports is in the range of one hundred meters and is intended for in-building local area networks.

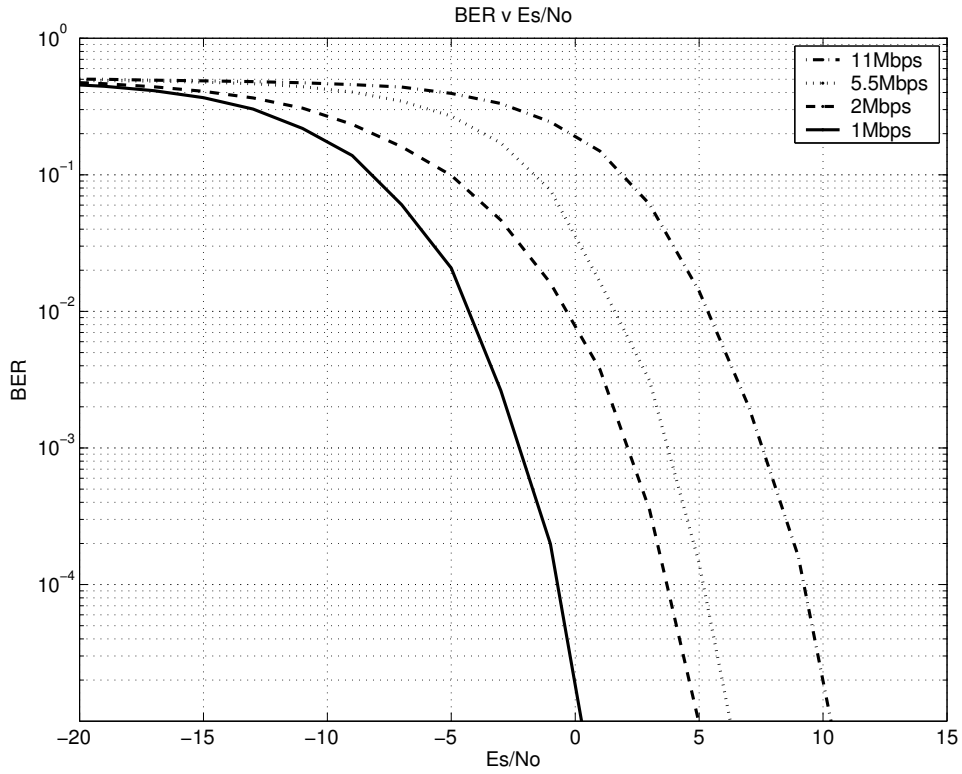


Figure 11: Comparison of bit error rates for various 802.11b modulation techniques in an AWGN channel.

.

As shown in Table 3, 802.11b uses direct sequence spread spectrum frequency selection at 2.4 GHz. It supports a variety of different modulation techniques, which dictate the supported bit rate. The 1 Mbps data rate is supported by the differential binary phase shift keying DBPSK modulation. All control channels in 802.11b operate using this robust but low data rate modulation scheme. IEEE 802.11b also supports a 2 Mbps using differential quadature phase shift keying (DQPSK). Finally, the 5.5 and 11 Mbps data rates use the complementary code keying (CCK) technique. Figure 11 shows the bit error rates for each of these modulation schemes under an additive white Gaussian noise (AWGN) channel. The lower bit rates offer more robust operation in lower SNR.

Medium access in 802.11b is handled by a distributed coordination function (DCF) with both physical and virtual carrier sensing. The physical carrier sense is handled

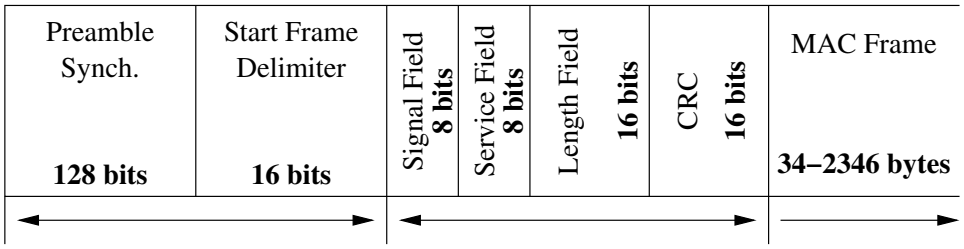


Figure 12: The IEEE 802.11b framing format [28].

via a carrier sense multiple access with collision avoidance (CSMA/CA) scheme with a random back-off time in the event of a busy condition. The virtual carrier sense mechanism uses a series of Request To Send (RTS) and Clear To Send (CTS) frames sent between the sender and receiver. This helps to alleviate the hidden terminal problem. The data frame format is shown in Figure 12. There is significant overhead in this packet format, and we will show how this effects the throughput and energy consumption. In addition to the frame overhead shown here, the MAC frame itself contains more overhead bits, including 24 bytes for address information and additional protocol information (i.e. TCP/IP).

The 802.11b also has a power saving mode which only works in the infrastructure mode when an access point is present. An 802.11b station will first send information to the access point indicating that it is entering the low power state. The station will then wake up every 100ms to receive a traffic indication map packet that will tell the wireless node when it should wake up to receive data. The duration of this traffic indication beacon can be configured in some hardware implementations. The node remains in the low power state, typically 1/10 the power dissipation of the always on mode, until the next beacon or scheduled data reception. However, this technique is not always robust as we shall see in section 5.1.1.

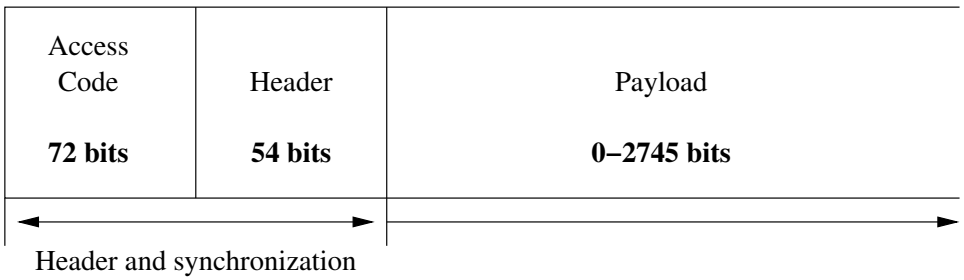


Figure 13: The Bluetooth framing format [28].

2.4 Bluetooth Personal Area Network

The Bluetooth wireless technology is a specification for a small, low-cost personal area network to connection various mobile computing devices such as laptops, cellular phones, and PDAs. It has been used as a wire replacement medium for computer peripherals such as keyboards and mice and headsets for mobile phones. It supports low data rate, low range applications and hardware implementations have low power requirements [110].

A basic Bluetooth network is called a piconet. A piconet consists of at most seven active mobile stations operating under the control of a master device. A collection of piconets is called a scatternet. A given spectrum allocation can support a fixed number of piconets. An inquiry and paging process is used to set up a piconet, and this process can take on the order of tens of seconds to complete. Bluetooth supports time-sensitive multi-media traffic as well as general data traffic. The range of a Bluetooth piconet is on the order of ten meters with a maximum data rate of 1 Mbps. The maximum throughput is 723 kbps in one direction or 433 kbps for a symmetric data traffic link. Multimedia traffic throughput is considerably less at 64 kbps in each direction.

At the physical layer, the Bluetooth network operates in the 2.4 GHz spectrum, which is divided into 79 channels each occupying 1 MHz. Not all countries have

Table 4: Summary of Bluetooth packet types.

Type	Max. User Payload (bytes)	TDD slots	FEC	Max. Symmetric Rate (kbps)	Max. Asymmetric Rate (kbps)
DM1	17	1	2/3	108.8	108.8
DH1	27	1	none	172.8	172.8
DM3	121	3	2/3	258.1	387.2
DH3	183	3	none	390.4	585.6
DM5	224	5	2/3	289.7	477.8
DH5	339	5	none	433.9	723.2
HV1	10	1	1/3	64	N/A
HV2	20	1	2/3	64	N/A
HV3	30	1	none	64	N/A

allocated this much bandwidth for Bluetooth, with Japan, Spain, and France only allocating 23 channels. Medium access is handled by a frequency hopping time division duplexing (FH/TDD) technique. Each time slot lasts for $625\mu s$, but a packet can last for more than one slot. For a symmetric data link, the master and slave nodes transmit in alternating time slots. The modulation scheme is a Gaussian-shaped binary frequency shift keying. The general packet format is shown in Figure 13. It has considerably less overhead than 802.11b packets, but the payload size is significantly smaller.

Bluetooth supports a variety of packet types for various traffic and channel conditions. Table 4 summarizes these packet types. Packet types begining with the letter 'D' are data packets, while packet types begining with the letter 'H' are multimedia or voice packets. For data packets, the number specifies the number of TDD slots a packet occupies. Voice packets only occupy one TDD slot with varying payloads. Data packets are divided into medium and high rate types, indicated by the middle letter. Medium rate packets employ a 2/3 rate Hamming code for error correction. High data rate packets do not use any error correcting codes, but both data packets use a CRC error detection field to check for transmission errors. Voice packets use either a 1/3 repetition code, a 2/3 Hamming code, or no error protection. Three times as many HV1 packets must be sent for every single HV3 packet. The header

bits in Figure 13 are always encoded using the robust 1/3 rate repetition code.

Bluetooth supports a variety of different power saving modes. A node can be in the active, sniff, hold, or park modes. In active mode, the node listens to the channel to see if the packet is addressed to it. In sniff mode, the slave only listens during a specified interval. This interval is configured between the master and slave nodes. In hold mode, the slave sleeps for some period of time and then restarts data transfers as soon as that time is over. Finally, in park mode the slave gives up its active piconet membership to enter a low power state. While a piconet can have only seven active members, it can handle up to 256 parked nodes. The node simply puts itself back into the active node list when it is ready for data transfers. Park mode has the lowest power consumption among the three power save modes.

2.5 Battery Technology for Mobile Devices

In the past 30 years, processor speeds and memory sizes have increased at a staggering rate, while battery technology has only increase by a factor of two to three. New battery technologies are being developed to minimize this gap, but the fact remains that battery technology has traditionally lagged behind advances processor and memory technology. With the proliferation of portable electronic devices, this fact emphasizes the need to use battery resources efficiently.

A battery technology can be rated according to several factors [34]:

Energy density The amount of energy stored per unit volume (Wh/l^3)

Specific energy The energy per unit weight of a battery (Wh/kg)

Nominal Voltage The average rated voltage output throughout the discharge cycle (V)

Rated Capacity The amount of current the battery can deliver over a specified period of time (mAh, milli-amp-hours).

The energy density and specific energy are used to rate the amount of energy with respect to the size and weight of the battery. A battery with a rated capacity of 1000 mAh will be able to deliver current of 1000 mA for one hour, 500 mA for two hours, or 2000 mA for half an hour. Given the rated capacity and nominal voltage, one can find the energy in the battery multiplying the two values.

Table 5: A comparison of battery technologies (AA size) [34, 87]

Technology	NiCd	NiMh	Li-ion
Voltage	1.2	1.2	3.6
Capacity (mAh)	620	1100	830
Energy Density (Wh/l^3)	90	158	270
Specific Energy (Wh/kg)	35	53	123
Total Energy (Joules)	2678	4752	10757
Cycle Lifetime	700	500	1200

In the area of recharageable battery technology, there have been three main types over the years. The first, nickel-cadmium (NiCd) technology is virtually non-existent in the marketplace today. This battery technology suffered from low energy densities and a memory effect that reduced the capacity after relatively few charge/discharge cycles. Nickel-metal hydride (NiMH) technology alleviates some of the memory effect of NiCd with increased energy densities, but the total lifetime of the battery is reduced. The most common technology is lithium-ion (Li-ion). Li-ion batteries have a much longer life cycle with increased energy densities, but the charging process requires more sophisticated electronics, which drives up cost [8]. Table 5 shows a comparison of battery technologies. For the time being, Li-ion is the predominant battery technology, giving the most energy with the longest lifetime (but with increased cost). Figure 14 shows the progress in lithium-ion technology specific energy over the last decade. It is expected that the specific energy will plateau in the years to come. While new technologies are being developed, such as lithium polymer or fuel cells, there is an ever-increasing demand for improved battery technology.

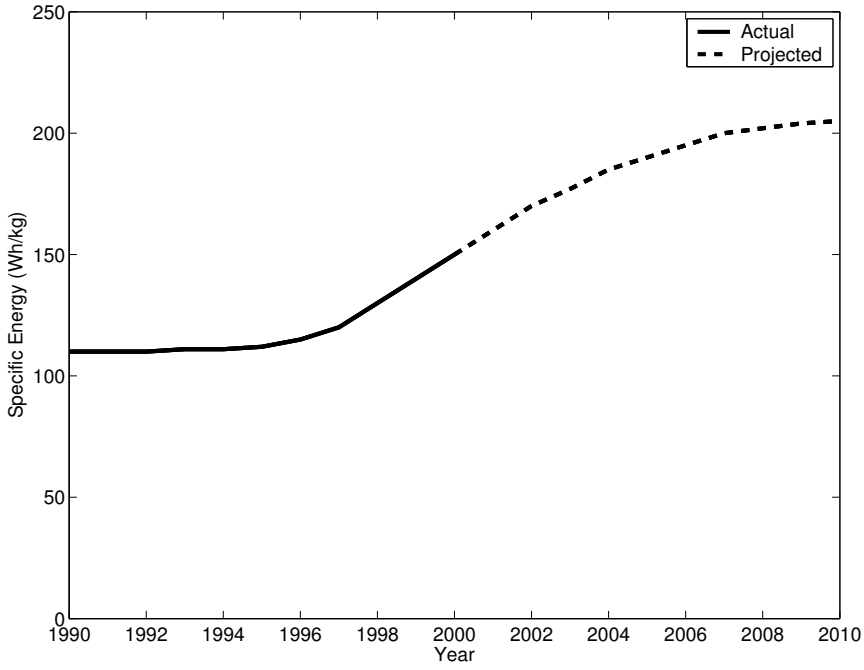


Figure 14: Progress in lithium ion battery technology [34].

2.5.1 Energy-Aware Design Principles

Given a fixed amount of battery energy, there has been an emphasis on energy-aware design principles in the literature. The goals of energy aware design are to put hooks or knobs into the hardware, software, or applications that allow scalability in quality vs. energy. This is different from low-power design, which often does not seek to allow scalability. The result of energy-aware design is a system that can adapt to changing conditions and modify it's energy usage accordingly. For example, a hand-held video streaming application might opt to send and decode video of decreased quality to extend battery lifetime. In another situation, the user might demand high-quality video, even if only for a short time.

Energy-aware design and scalability must take place at all levels, from the device level to the application layer. Many CPUs already allow energy scalable operation through techniques such as dynamic voltage and frequency scaling. Running a particular application at it's lowest frequency and voltage setting that still provides

acceptable performance will save energy. Dynamic application of this technique can be difficult since the operating system must have knowledge of the operating requirements of various applications running on the system. For applications such as speech recognition, this information may be difficult to predict far in advance. Operating systems are becoming increasingly aware of energy considerations, but fine grained control requires the assistance from the application layer. Memory subsystems can be designed such that entire banks of memory are shut off when not needed, but, once again, the operating system must maintain control of these adaptations.

Software optimization techniques can also help to reduce energy consumption. By writing software that will run efficiently on a particular platform, the program can use less resources, including battery energy. Compiler optimizations only offer marginal improvements in energy consumption. Any significant gains will require optimizations that address bottlenecks with respect to the particular architecture studied. This may include limiting the mathematical precision (i.e. fixed point arithmetic), efficient data structure organization to reduce cache misses, and the use of approximate algorithms when hardware accelerated versions are not available, such as square root, logarithmic, or trigonometric functions. Finally, the wireless network can use significant amounts of power in an embedded system. Table 6 shows the power dissipation of various components of the HP Labs Smartbadge IV embedded system. These are average power measurements during some moderate CPU processing and wireless network activity. The 802.11b network interface used almost half of the power of the total system, therefore wireless network optimization is an important consideration.

Table 6: Power dissipation for major subsystems of the HP Labs Smartbadge IV.

Subsystem	Power (mW)	Percentage
CPU	694	21%
Memory	1115	34%
802.11b	1500	45%
Total	3309	100%

The wireless network power optimization problem has been addressed at different abstraction layers, starting from the semiconductor device level to the system and application level. Energy efficient channel coding and traffic shaping to exploit battery lifetime of portable devices were proposed in [10]. A physical layer aware scheduling algorithm aimed at efficient management of sleep modes in sensor network nodes is illustrated in [91]. Energy efficiency can be improved at the data link layer by performing adaptive packet length and error control [57]. At the protocol level, there have been attempts to improve the efficiency of the standard 802.11b, and proposals for new protocols [43, 52, 97]. Packet scheduling strategies also can be used to reduce the energy consumption of transmit power. In [83], 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 [2]. Savings of as much as 50% in WLAN power consumption, relative to just using 802.11b power management, were reported.

Traditional system-level power management techniques are divided into those aimed at shutting down components and policies that dynamically scale down processing voltage and frequency [95, 1]. Energy-performance tradeoffs based on application needs have been recently addressed [53]. Several authors exploit the energy-QoS tradeoff [69, 70, 101, 61]. A different approach is to perform transcoding and traffic smoothing at the server side by exploiting estimation of energy budget at the clients [90]. 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 [92]. 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 [59, 4, 29, 60].

CHAPTER III

VOICE USER INTERFACE PROTOTYPE

As part of the Yamacraw Wireless Systems Prototyping effort, we have built a working voice user interface for a portable wireless device The goal of project is to develop a system prototype for a hand-held device capable of high-speed wireless networking. The voice user interface will serve as the main method of interaction with the device. The system was written using off-the-shelf speech technology components coupled with a custom HTML/VoiceXML language to provide multi-modal input/output. In this chapter, we outline the design and implementation of this system, including integration with a software radio testbed and implementation on an actual PDA device.

3.1 VoiceXML/HTML Browser

The goals of the voice user interface was to provide web browsing capability through a multimodal interface. It must support the following objectives:

- Accurate, real-time speech recognition capability.
- High quality text-to-speech synthesis.
- Pre-recorded digital waveform output.
- Out-of-the box functionality with little or no training for a new user.
- Pen-based input as a fall back when audio input/output is not appropriate.
- Robustness to environmental noise.
- Web-browsing capability.

- Intregration with other applications such as streaming video.

When we started this project, a standard language for this kind of interaction did not exist. The VoiceXML standard was recently released, but it did not support the kind of multimodal interaction required for this application. The SALT and XHTML+Voice standards had not been published.

Initially, we investigated a technique to parse standard HTML documents for navigation information such as hyperlinks which could then be used to construct a voice grammar for speech based navigation. This proved to be too difficult as most HTML documents use ambiguous or repetitive text to indicate navigation points within a document (i.e. [click here]). In addition, the use of more sophisticated navigation techniques such as image maps and javascript would require significant effort to extract a semantically meaningful phrase or word for use in voice navigation.

Since we required both visual and audio interfaces, we decided to augment HTML pages with VoiceXML tags. This allows a "click-or-say" input paradigm where the user can select hyperlinks via the stylus or navigation items can be spoken. Form-filling scenarios can be handled in a similar manner if there is some context to allowed input (i.e. stock quotes, city names, dates and times, etc.) Listing 3.1 shows an example of this hybrid VoiceXML/HTML input. The visual component can display the HTML output, while the voice user interface can parse an interpret the VXML portion of the input, including queueing of text to speech prompts, activation of grammars, and turning the microphone on and off.

A block diagram of this system is shown in Figure 15. The device itself will support stylus and audio input and both video and audio output. Audio output can be either text to speech or pre-recorded audio prompts. Documents are served from standard web servers on the internet. Our architecture allows for a flexible approach where speech recognition and text to speech synthesis can occur on the mobile device or elsewhere on the network, depending on the capabilities of the device. The

Listing 3.1: A hybrid VoiceXML/HTML Document

```
<?xml version="1.0"?>
<html>
  <body>
    <center>
      <h6>Enter stock ticker symbol or just say the company name: </h6>
      <form method="get" action="/cgi-bin/demo/stock_quote2.pl">
        <input name="stock" type="text" size="5"/>
        <input type="submit" value="Get_Stock_Quote"/>
      </form>
      <table width="550" border="0">
        <tr/>
        <td align="left" width="50%">
          <h6>
            <a href="/demo/main_menu.vxml">Previous</a>
          </h6>
        </td>
        <tr/>
        <td align="left" width="50%">
          <a href="/demo/main_menu.vxml">
            <img src="/demo/images/left.gif" border="0"/>
          </a>
        </td>
      </table>
    </center>
  </body>
  <vxml version="1.0">
    <link next="/demo/main_menu.vxml">Main Menu</link>
    <link next="/demo/main_menu.vxml">Back</link>
    <link next="/demo/main_menu.vxml">Previous</link>
    <form>
      <field name="stock">
        <prompt>Please say a company name.</prompt>
        <grammar src="sp500.bnf" type="application/x-jsgf"/>
      </field>
      <filled mode="all" namelist="s">
        <submit next="/cgi-bin/demo/stock_quote2.pl" method="get"/>
      </filled>
    </form>
  </vxml>
</html>
```

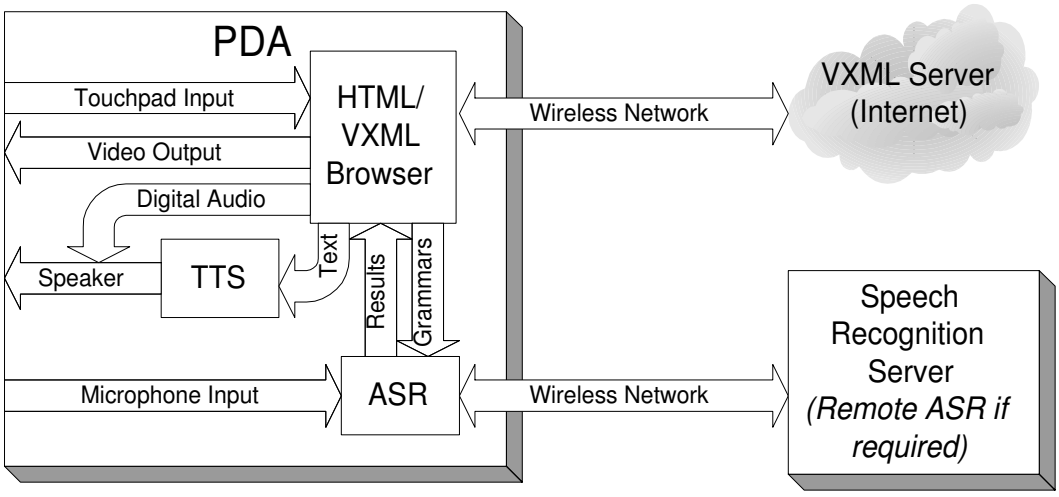
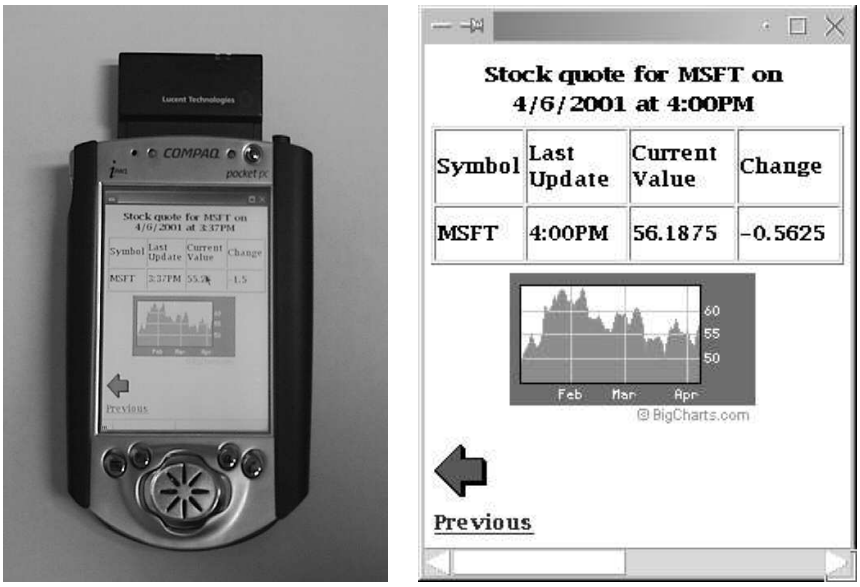


Figure 15: System architecture of the Yamacraw Voice User Interface.

VoiceXML/HTML browser is the heart of the system, and it coordinates the navigation aspects via voice or stylus input, the display component, and the queueing of audio prompts for the user. Finite state speech grammars are fetched from the network and activated or deactivated depending on the current state of the application. Using this prototype framework, we were able to build a number of successful applications, including real-time weather information and stock quotes.

3.1.1 Compaq iPAQ Implementation

We have ported this generic voice user interface system to a Compaq iPAQ PDA running Linux. The iPAQ contains a 206 MHz StrongArm processor with 32 Mbytes of memory. The PDA uses a WaveLAN wireless network PCMCIA card. Audio input is from the integrated microphone that comes with the device. The iPAQ offers no external microphone input. Although the system currently runs best from a headset microphone, noise robustness techniques will allow for easier interaction using an open air microphone mounted on the unit. Other solutions to increase the audio quality could include adaptive microphone arrays capable of acoustic beam-forming. However, the speech recognition accuracy is surprisingly good in rooms with light office noise, but it fails in more noisy environments such as hallways or lecture rooms.



(a) Compaq iPAQ PDA (b) Screen Capture

Figure 16: A Voice User Interface Demonstration on a Compaq iPAQ PDA.

The entire VoiceXML/HTML browser was written in Java for portability. This includes the VoiceXML parser and interpreter, the HTML viewing component, and the various interfaces to the text to speech and speech recognition servers. The demonstration system supported either local or remote text to speech synthesis via the Festival TTS system, which is a public domain TTS system intended for academic research. The system also supports the ViaVoice Outloud TTS system but it cannot run on the iPAQ itself. Speech recognition takes place on a Linux server using the IBM ViaVoice speech recognition engine. The server is capable of simultaneous connections from multiple clients. Raw audio is sent over the network from an audio server running on the client, and recognized text is sent back to the client from the speech recognition server when a phrase or word is recognized.

The iPAQ runs a simple demonstration which allows for access to real-time stock and weather data as well as providing some basic personal information management tools (e.g. scheduling, contacts, etc.) Screen size is limited to 240-by-320 pixels.

This size is one sixth the size and one fourth the resolution of the absolute minimal acceptable display on a modern desktop PC. Efficient use of on-screen real estate is essential. Figure 16 shows a screen shot of the voice user interface prototype running on the iPAQ device.

3.1.2 Software Radio Integration

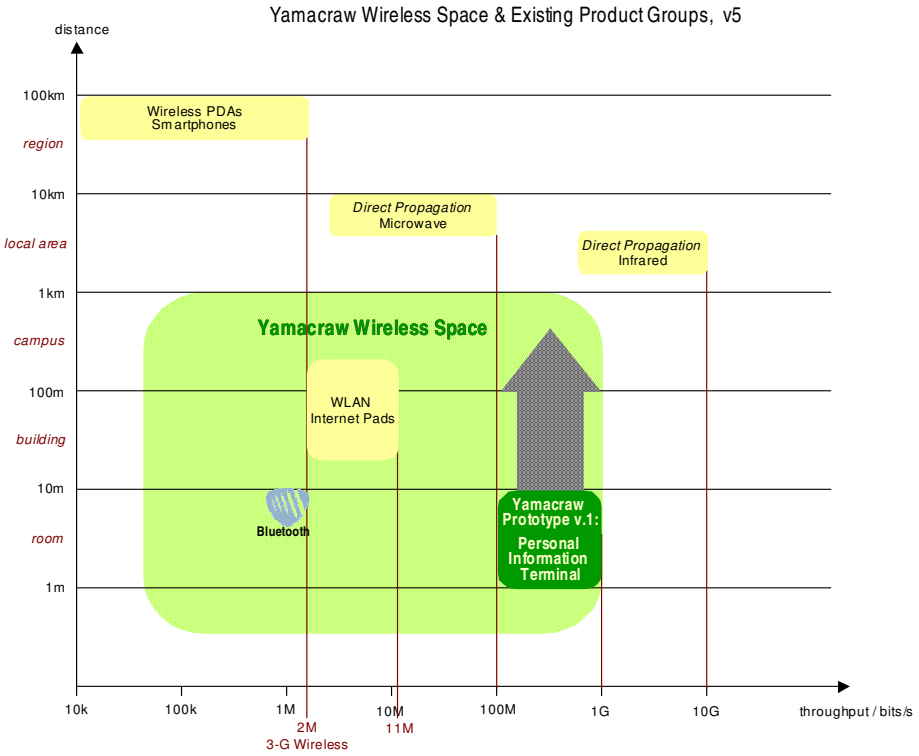


Figure 17: Bit rate and range of the Yamacraw Wireless Prototype System.[1]

As part of the Yamacraw wireless system prototyping research group, we integrated the voice-user interface with the software radio platform. The wireless prototype is a high-speed (100 Mbps to 1 Gbps) wireless network. Figure 17 shows the bit rate and range space of this research effort. The software radio platform is a collection of hardware for prototyping new wireless networking technologies from the physical layer to the link layer. The software radio testbed handles the physical layer

[1]From the Yamacraw Prototyping Notes of Prof. Jayant.

interface, modulation, antennae, and channel coding. If Figure 18, we show the block diagram of the wireless system. The MIMO transmitters and receivers are part of the software radio *cage*, which contains various DSP boards and high speed interfaces. Connected to the cage via a high speed interface are host workstations that run both the MAC layer and network layers as well as the applications themselves.

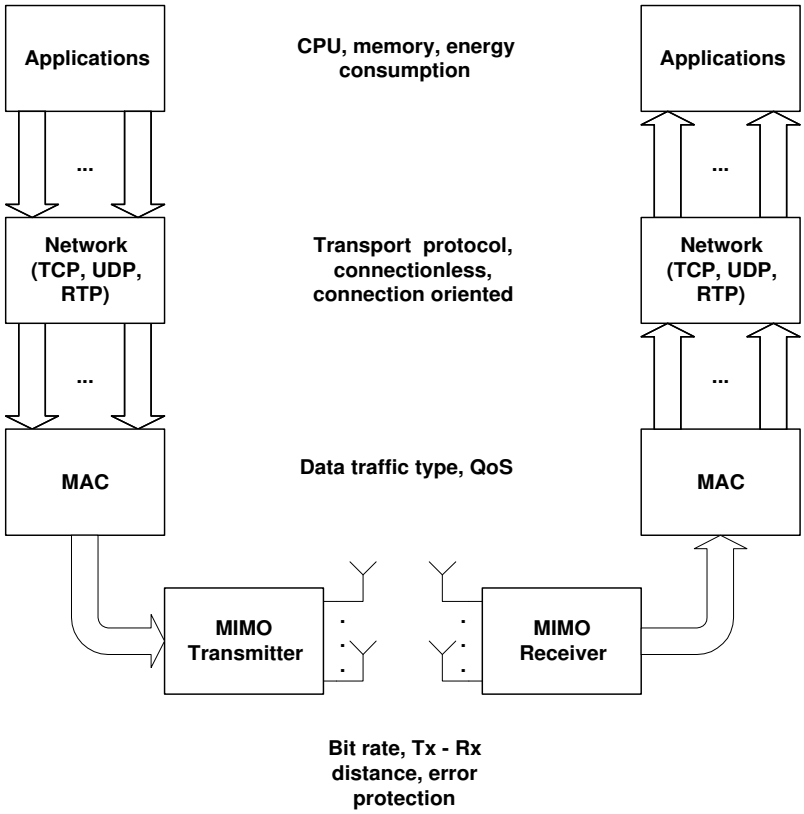


Figure 18: A one-way wireless network showing research issues related to distributed speech recognition applications.

The wireless infrastructure is designed in parallel with the voice user interface. Therefore, the current version of the voice user interface uses and 802.11b wireless interface, with a maximum supported bit rate of 11 Mbps. The 802.11b version of the system currently runs on a hand-held device, while the integrated system will run on a PC connected to a software radio system. The host PC communicates with the software radio system via a high speed bi-directional FPDP interface. We were able to perform generic TCP/IP networking across the software radio system in order

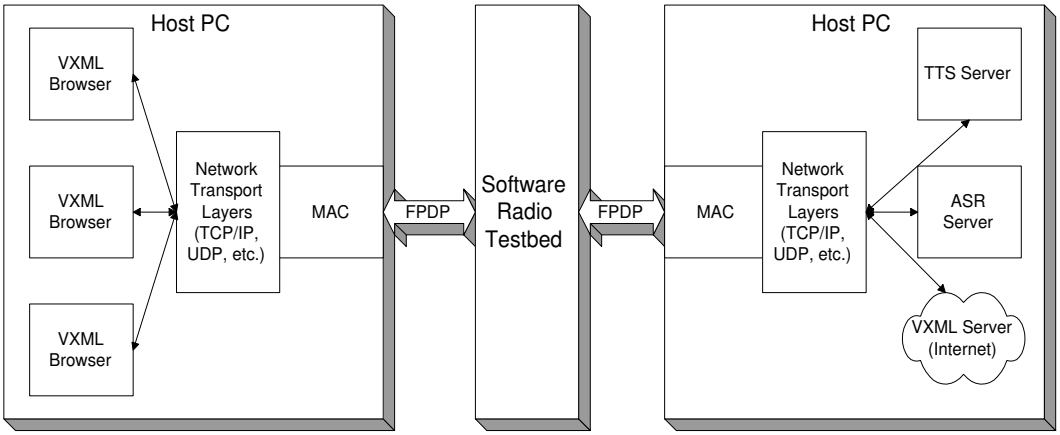


Figure 19: Software Radio Testbed Integration.

to demonstrate a working end-to-end system. Integration with a streaming video application demonstrates the capability of the MAC layer to handle multiple traffic connections and types. Figure 19 shows a block diagram of the integrated system.

CHAPTER IV

SPEECH RECOGNITION FOR EMBEDDED SYSTEMS

As we have seen, implementing high quality speech recognition on an embedded system, such as a cellular phone, PDA, or other device is a difficult challenge. In this chapter, we discuss some of these challenges in detail and present some solutions. In section 4.1 we present a software based front-end feature extraction for a distributed speech recognition system that is designed for minimal power consumption. Through algorithmic, architectural optimizations, and dynamic voltage scaling, we are able to reduce the energy consumption of the signal processing algorithm on a general purpose processor by 89%. In Section 4.2, we present a model for the arithmetic complexity of a small vocabulary connected word speech recognizer. The model shows how the various parts of speech recognition scale with increasing word count. Although we do not consider search reduction techniques such as beam pruning, the model still serves as a useful comparison between various parts of an ASR system for small vocabulary applications. In section 4.3 we present some hardware based solutions that will help enable large vocabulary speech recognition on a mobile device. In addition, we discuss some profiling results of the Sphinx III speech recognition system, including memory bandwidth requirements and computation required for back-end processing. We also present a coarse level model of energy consumption for client-side ASR on the Smartbadge IV embedded device. This will be used as a comparison for DSR techniques in Chapter 5.

4.1 Low-Power Front-End Feature Extraction for a Distributed Speech Recognition System

This section describes the optimization of a signal processing front-end feature extraction for a distributed speech recognition system. The baseline system used in the experiments is version 0.3 of the open-source Sphinx II speech recognizer from Carnegie Mellon University [106]. The optimization methods used for the algorithm substantially decrease the power usage while increasing speed (measured in processor cycle counts). Estimates of total power usage are performed using a cycle-accurate energy consumption simulator [96]. The architecture of the embedded system simulated in the experiments mimics that of the Smartbadge IV system developed at the Appliance Platform department of HP Labs [62]. In addition to performing energy consumption simulations to evaluate the quality of source code optimizations, we also implemented and ran the optimized version of the front-end on Smartbadge IV hardware. We found that real-time signal processing of speech is possible at eleven discrete CPU frequency and voltage settings, thus enabling further power savings.

A block diagram of a speech recognition system is shown in Figure 3. It can easily be divided into two parts, a front-end and a back-end. The front-end produces a set of acoustic observations which are useful in recognizing speech. The back-end is where most of the computation and memory usage takes place. The back-end can easily use hundreds of Mbytes of memory and hundreds of MIPS of computation.

Since the front-end feature extraction step is relatively low in complexity, it is desirable to perform this step on the embedded device and to send compressed features across the network. It has been shown that these features can be compressed with little effect on the error rate of the speech recognizer [123]. The ETSI standard for distributed speech recognition describes algorithms to compute, compress, and transmit these speech features [109]. We consider several bit rates and quanitzation levels, including one that is similar to the ETSI standard.

4.1.1 Low-Power Optimization

Implementing the front end feature extraction for a distributed speech recognition system on an embedded platform requires not only speed, but also power optimization, since the battery 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 (e.g. an embedded system with no floating-point hardware). Ideally, many of these optimizations should be done by a compiler. However, currently available compilers for most embedded systems do not have these optimizations built-in. In addition, measurements presented in [96] show that the improvements that can be gained using standard compiler optimizations are marginal compared to writing energy efficient source code. The second category of source code optimizations is more general and involves changes in the algorithmic implementation of the source code with the goal of faster performance with less power consumption.

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

4.1.1.1 Architectural Optimization

Signal processing algorithms such as the one in Figure 5 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 [107].

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

Fixed-point arithmetic uses scaled integers to perform basic math functions using the existing integer hardware. The scaling factor (or location of the decimal point) is fixed at design time and is designated by Qn, where n is the number of bits to the right of the decimal. For example, consider the following 8-bit number in $Q4$ format:

$$0101.0101_b = 4 + 1 + 2^{-2} + 2^{-4} = \frac{85}{16} = 5.3125 \tag{9}$$

The basic rules of arithmetic still hold; adding two numbers requires that the decimal points must line up. Multiplying two numbers in Qn format yields a number in $Q2n$ format.

Implementing a pre-emphasis filter and Hamming window using fixed-point arithmetic is straight-forward. Fixed-point FFTs are well studied and have often been implemented on digital signal processor chips. There is an average of one bit of growth in each FFT stage due to the multiply and accumulate operation. Therefore, the appropriate number of upper bits must be zero to prevent overflow.

After passing the input frame through the FFT, the mel filter bank must be applied. The filterbank amplitudes are calculated using the squared magnitude. This presents some challenges since this squared number multiplied by the filter coefficients, $H_i[k]$, can easily overflow the 32-bit registers. A 64-bit result can be obtained from the StrongARM multiplier using assembly language, but overflow can be avoided simply

by rewriting the filter bank equation (4) to use just the magnitude:

$$Y[i] = \sum_{k=0}^{N/2} \left(|X[k]| \sqrt{H_i[k]} \right)^2 \tag{10}$$

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

The one drawback to this method is that computing the magnitude requires a square root operation. Fast integer square root algorithms exist, but they must be used on each output from the FFT, which is costly. Fortunately, the magnitude can be estimated as a linear combination of the real and imaginary parts using the following equation [30]:

$$|x| \approx \alpha \max(|\Re\{x\}|, |\Im\{x\}|) + \beta \min(|\Re\{x\}|, |\Im\{x\}|) \tag{11}$$

where α and β are chosen to minimize a particular kind of error, and $\Re\{x\}$ and $\Im\{x\}$ represent the real and imaginary parts of the complex number x. Taking the absolute value of the real and imaginary parts forces the result to lie within the first quadrant. The min and max functions limit the angle further to between 0 and $\frac{\pi}{4}$ radians. A linear combination of the real and imaginary portion in this range is a reasonable approximation of the magnitude. The values of α and β are chosen to have an easy fixed-point representation that minimizes the mean error.

Computing the first 13 coefficients of the DCT is relatively easy to do in fixed-point arithmetic, but taking the natural logarithm is a more difficult task. One possible option is to perform a floating-point logarithm, but profiling showed that the logarithm itself as well as the transition to and from fixed-point is costly. A fixed-point logarithm using a polynomial expansion requires some divides, which are slow on the StrongARM. However, there is an interesting algorithm to estimate $\log_2(x)$ using simple bit manipulation, which is faster than other methods of calculating the logarithm. This algorithm, described in [14], is very low in complexity and gives an approximate fixed-point result. The leftmost non-zero bit position (starting with 0 as

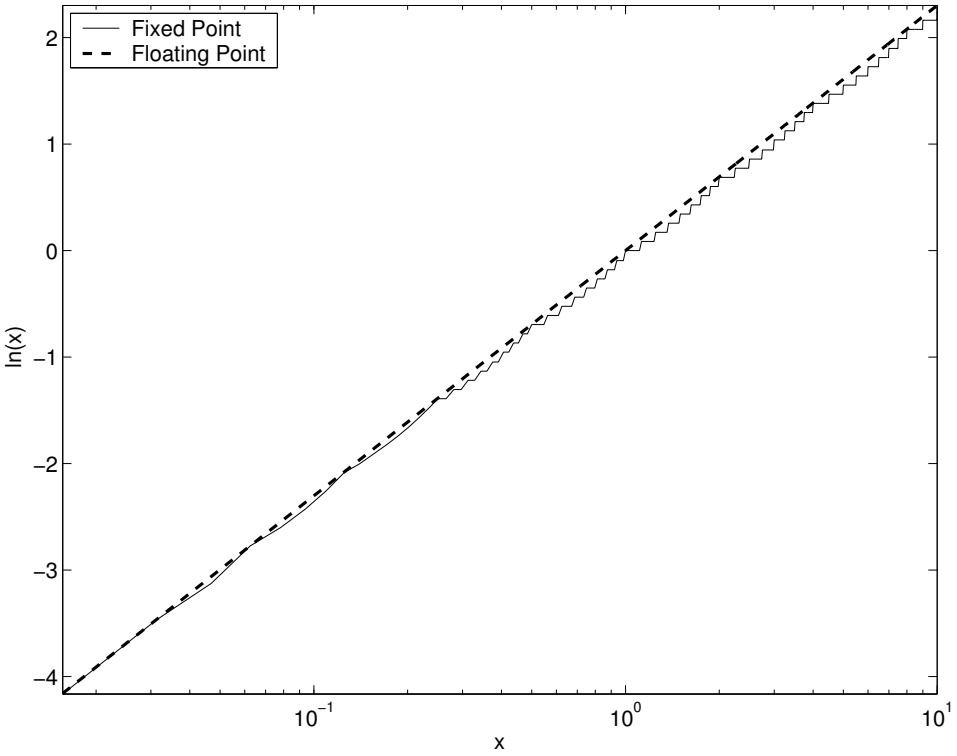


Figure 20: Estimate of the natural logarithm.

the low order bit) is the integer portion of the logarithm. Call this number b, which is the exact answer for powers of 2. Mask off the next three bits and shift them all the way to the right. Call this number n. Then $8b + n$ is the logarithm base 2 in $Q3$ format (i.e. 3 bits of precision to the right of the decimal). Masking off the three bits after the high order bit gives a crude interpolation of the logarithm to the next power of two. The $\ln(x)$ can be determined by multiplying by a constant as follows:

$$\ln(x) = \log_2(x) \ln(2) \tag{12}$$

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

$$\ln\left(\frac{x}{2^n}\right) = [\log_2(x) - \log_2(2^n)] \ln(2) \tag{13}$$

$$\ln\left(\frac{x}{2^n}\right) = [\log_2(x) - n] \ln(2) \tag{14}$$

Equation (14) is the expression used to calculate the natural log of a fixed-point number. Using precision of Q3, this estimate of the logarithm has a maximum error of around 0.152 and an average error of around 0.0866. The results of Equation 14 as well as the floating point logarithm are shown in Figure 20. The fixed-point estimation of the natural logarithm is very close to the actual floating-point value.

4.1.1.2 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 [77]. Some further processing of the output is required to get the desired result, but this overhead is minimal compared to the reduction in computation. Additional savings can be obtained when the trigonometric functions used in the computation of the FFT are pre-computed and stored in a lookup table, thus eliminating multiple function calls in the FFT loop.

4.1.1.3 Dynamic Voltage Scaling

Dynamic voltage scaling (DVS) algorithms reduce energy consumption by changing processor speed and voltage at run-time depending on the needs of the applications running. Processors can often operate over a range of frequencies. For each frequency there is a minimum allowed voltage that still guarantees correct results. When the processor is run at the minimum frequency and voltage required to sustain the performance level required by the application, it is possible to obtain large power savings. If only processor frequency is scaled, the total energy savings are small as power is inversely proportional to cycle time and energy is proportional to the execution time and power. When both frequency and voltage are scaled, the power savings are proportional to frequency and to the voltage squared as shown in (15) where C is the

capacitance switched, f is the switching frequency, and V is the voltage.

$$P \propto CfV^2 \tag{15}$$

Early DVS algorithms set processor speed based on the processor utilization of fixed intervals and did not consider the individual requirements of the tasks running. There has been a number of voltage scaling techniques proposed for real-time systems. The approaches presented in [39, 40, 42, 116] assume that all tasks run at their worst case execution time (WCET). The workload variation slack times are exploited on task-by-task basis in [93], and are fully utilized in [56]. Work presented in [73] introduces a voltage scheduler that determines the operating voltage by analyzing application requirements. The scheduling is done at task level, by setting processor frequency to the minimum value needed to complete all tasks. The voltage scheduling to adjust for variations within tasks, such as variation in MP3 audio frame arrivals or MPEG video frame decoding times, has been considered in [94, 1].

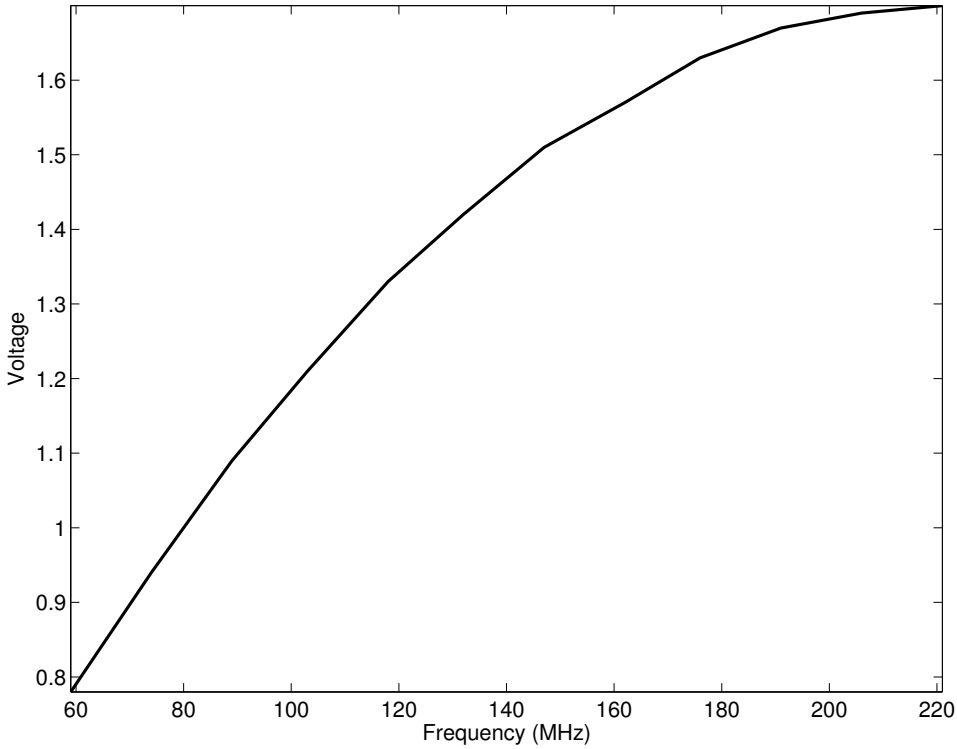


Figure 21: Frequency vs. Voltage for Smartbadge IV Strongarm CPU

Once the code is optimized for both power consumption and speed, we investigate the energy savings from DVS. The StrongARM processor on Smartbadge IV can be configured at run-time by a simple write to a hardware register to execute at one of eleven different frequencies. Figure 21 shows the frequency-voltage tradeoff. Note that the number of frequencies, eleven, is predefined by the design of the StrongARM processor. We measured the transition time between two different frequency settings at 150 microseconds. Since typical processing time for the front-end is much longer than the transition time, it is possible to change the CPU frequency without perceivable overhead. For each frequency, there is a minimum voltage the SA-1110 needs in order to run correctly, but with lower energy consumption. The easiest way to determine the lowest possible frequency and voltage for such stand alone application is to run it at all possible frequency settings, with voltage set to minimum allowed, and observe if the code still runs in real time. In our case, we obtained real time performance at all possible frequency and voltage settings.

4.1.1.4 Results of MFCC Optimization

Three main criteria are considered in order to evaluate the effectiveness of a particular optimization: performance (in terms of processor cycle count), energy consumption, and accuracy or word error rate (WER). Simulation results for processing one frame (25ms) of speech on the Smartbadge IV architecture running at 202.4 MHz are shown in Figure 22. 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 4.1.1.2 as well as some of the C source optimizations described in [107]. 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 the cycle-accurate energy simulator, and include processor core and level 1 cache energy, interconnect and pin energy, energy used by the memory, losses from the DC/DC converter, and battery inefficiency [96]. The reduction in energy consumption is not as dramatic as the performance improvement for the fixed-point version due to an increase in memory references per unit of time. In fixed-point code, basic math operations are reduced to a few cycles as opposed to long iterations of floating-point emulation which do not require as many memory references. However, we have still achieved a reduction in the total battery energy required to process one frame of speech data by 83.5%. The front-end was tested using the TIDIGITS

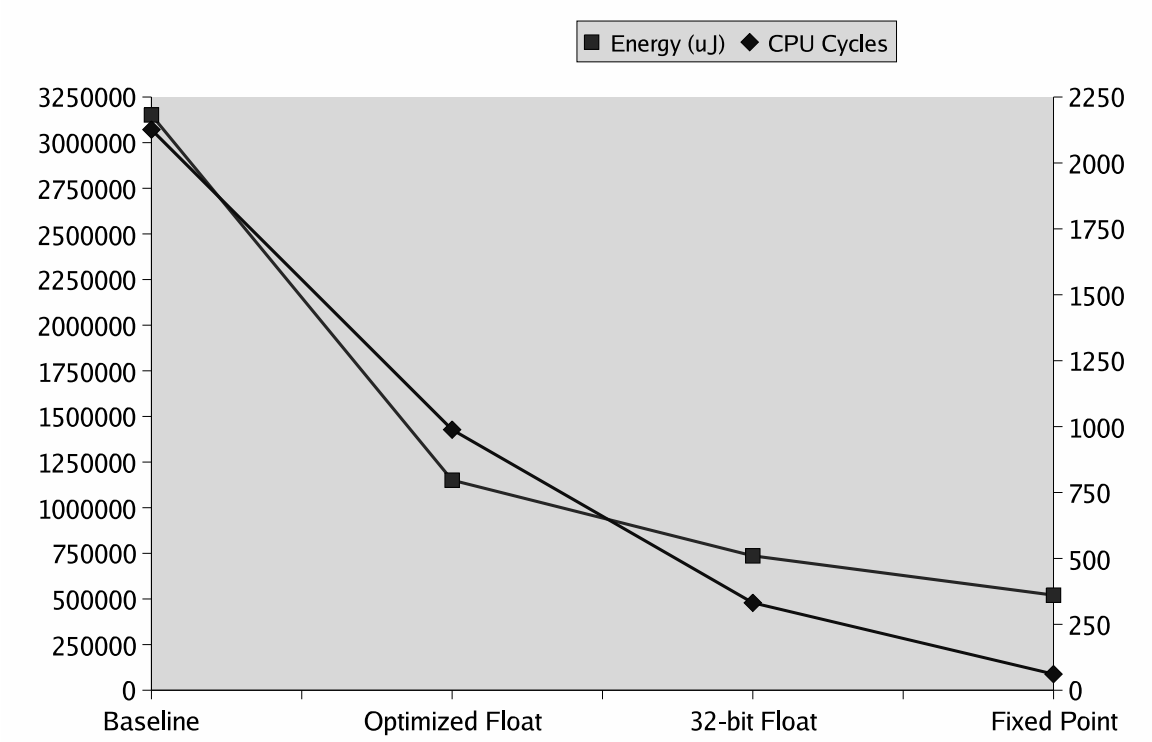


Figure 22: Performance and energy consumption per frame of speech.

speech database, and the results are shown in Table 7. A continuous digit speech recognizer was trained using the TIDIGITS database of 8,623 utterances from both male and female speakers. The original floating point front-end was used to generate

mel-frequency cepstral coefficients for the training set. No secondary features (first and second time derivatives of the mel-frequency cepstrum) were used in the training or test phases. The trained speech models were then used to recognize speech from the TIDIGITS test set of 8,700 utterances. The WER was calculated using the various front-end implementations and is shown in Table 7. There is no loss in accuracy among the three floating-point implementations, but the fixed-point implementation uses some approximate algorithms that can create a slight mismatch between the training and test data. We were able to eliminate the slight 0.1% increase in WER by using the fixed-point front-end during the training phase. In addition, Table 7 shows a minimal increase in look-up table size and code size, so the memory requirements for the fixed-point optimized code are about the same. Another performance metric reported in Table 7 is how long it took for each code implementation to process 1 second of speech at the processor clock speed of 202.4 MHz (Time column). The fixed-point version runs 34 times faster than the baseline system.

Table 7: TIDIGITS test set results.

	Code size (Bytes)	Lookup table (Bytes)	Time (sec)	WER Increase % (absolute)
Baseline	29704	N/A	1.510	0.0%
Optimized Float	31960	88120	0.699	0.0%
32-bit Float	31272	88120	0.235	0.0%
Fixed-Point	33124	88136	0.043	0.1%

Because the fixed-point code runs much faster than real-time at 202.4MHz, it is possible to get further reductions in power usage by using DVS as discussed in section 4.1.1.3. The results from this experiment are shown in Table 8. These power measurements are performed on the Smartbadge IV system running the eCos embedded operating system and using the WaveLAN card to transmit the uncompressed cepstral parameters. The P_{sys} measurement is taken from the main power supply

output. At 59 MHz the algorithm still runs in real-time, and the system uses 34.7% less power than at 206 MHz. Combining the DVS results with the source code optimizations, we calculate the overall reduction in power consumption to be 89.2%.

Table 8: Measured Power Consumption with DVS.

Frequency (MHz)	Voltage (V)	P_{sys} (mW)
59	0.78	1721
74	0.94	1807
89	1.09	1901
103	1.21	2029
118	1.33	2114
132	1.42	2234
147	1.51	2320
162	1.57	2432
176	1.63	2508
191	1.67	2568
206	1.69	2636
221	1.70	2748

4.1.2 Vector Quantization

Finally, we include the fixed-point vector quantization code in our profiling and consider different bit rates and quantization levels. Although some differing techniques have been proposed, the most common technique for compressing MFCCs is some form of vector quantization. In vector quantization, we train a set of codebooks against some speech data. These codebooks contain a set of centroids representing the clusters that occur in the training data. We simply transmit the centroid index for each codebook. Smaller codebooks will result in a noisy representation of the original signal, and speech recognition accuracy will degrade.

For our system, we use an intra-frame product code vector quantization scheme presented in [25]. We use the existing bit allocation in [25] to train a set of codebooks

using a K-means training algorithm with bit rates ranging from 1.2 kbps to 2.0 kbps. We include an additional bit allocation that is similar to the ETSI standard that will operate at 4.2kbps [109]. The actual bit rate needed for a speech recognition task depends on many factors such as acoustic and speaker conditions as well as the vocabulary size and complexity of the acoustic models used. In [25], the range of bit rates was evaluated for a small vocabulary task under ideal acoustic conditions. We can expect the word error rate (WER) to increase under less ideal conditions (i.e. larger vocabulary, more acoustic background noise, etc.). Table 9 shows the resulting bit rates and word error rates from [25] on the rows labelled VQ-XX, where XX is the number of bits per 10 ms speech frame. The WER for full bandwidth speech at 16 kHz and 16 bits per sample (256 kbps) was 6.55%.

Table 9: Word error rate for several bit rates [25].

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

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 as shown in Figure 23. 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.

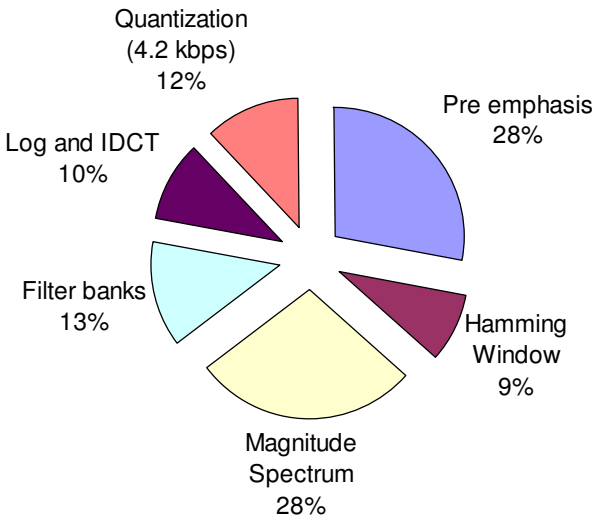


Figure 23: Energy consumption per DSR functional block.

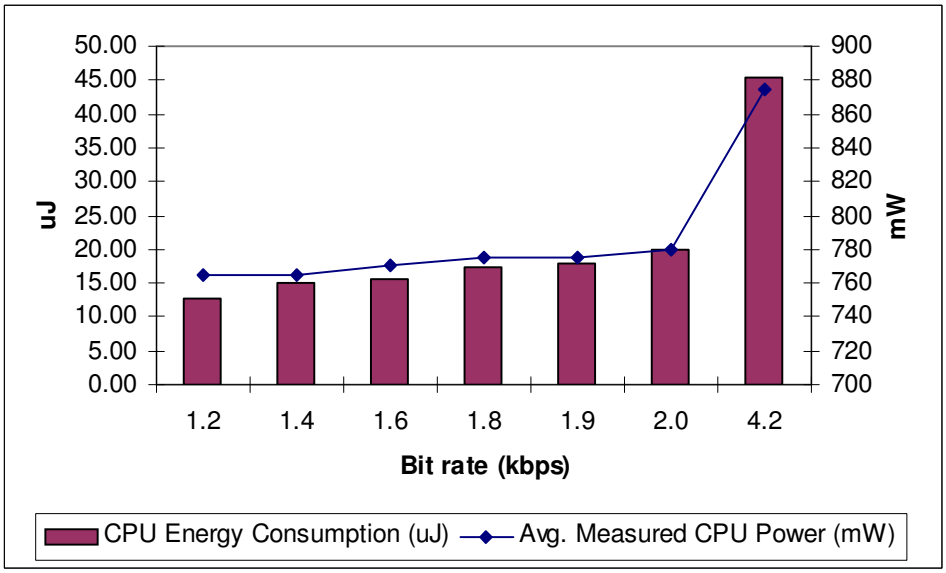


Figure 24: Computational energy usage and measured average power for different quantization bit allocation schemes.

Figure 24 shows a comparison of energy consumption for various vector quantization bit allocation schemes. The bars represent the total energy consumption per

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

4.1.3 Summary

In this section, we have outlined some optimization techniques to reduce the energy consumption of a particular signal processing algorithm. On embedded systems with no floating-point hardware, fixed-point arithmetic is an important step in lowering the power consumption of a program. However, careful attention must be paid to basic math functions (i.e. cosine, log, etc.) and overflow/underflow issues. Approximate algorithms perform well for certain applications and can result in huge savings in both time and power usage. By using software optimizations, we were able to achieve a reduction in energy usage by 83.5% compared to the unoptimized source code. We show that additional power savings are possible by scaling processor frequency and voltage at run time, while still meeting the performance requirements. At the lowest frequency/voltage setting, we calculate an overall reduction in power consumption by 89.2%. With the addition of vector quantization, the total energy required to process one frame of speech data is approximately 380 μJoules.

4.2 Small Vocabulary Connected Word ASR

In this section, we develop a model of arithmetic computation for small vocabulary connected word ASR. As we began to investigate the computation involved in automatic speech recognition (ASR), it became clear that there were many different options and tradeoffs to consider. Although portable hand-held devices are becoming more powerful, they are still limited by memory, computational ability, and battery energy. Complicated ASR tasks along with other user-level applications can easily consume all available resources. We intend to examine some ASR algorithms under a range of tasks to determine which tasks are suitable for local vs. distributed processing. In this section, we will develop a model of the computational complexity of a connected word automatic speech recognizer using Hidden Markov Models (HMM) with continuous Gaussian mixture densities. We will examine the number of arithmetic operations in each block of a simple speech recognition system. We will also estimate the storage required for the acoustic models. We will use these numbers to characterize a continuum in a distributed (client-server) view of automatic speech recognition.

4.2.1 System Overview

Figure 25 shows a block diagram of a typical connected word speech recognizer. This type of speech recognizer is practical for small vocabulary systems. We will consider the complexity of a HMM speech recognizer with continuous Gaussian mixture densities. A more in-depth discussion of speech recognition systems can be found in [41] and [80].

We assume that the speech signal is broken up into overlapping frames (typically 25ms) and each frame is processed by each block in the system. Frames are typically processed at a rate of 100 frames per second. In this paper, we will consider the computational complexity on a per frame basis.

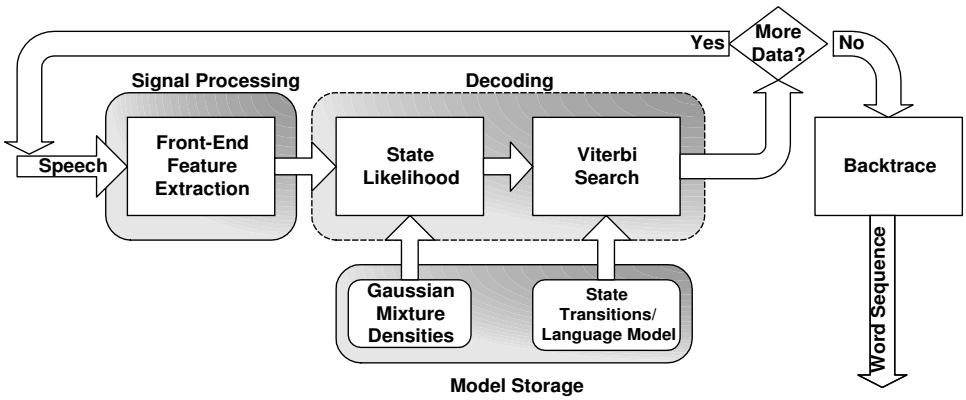


Figure 25: A block diagram of a typical HMM connected word speech recognizer.

4.2.2 Metrics of Complexity

The arithmetic complexity is measured by counting adds, multiplies, and general math functions in critical loops of the algorithms. A general math function is some high-level mathematical function which generally requires some number of additions, multiplies, or table look-ups to be computed (such as logarithmic functions). For simplicity, we are assuming that these high-level math functions are equivalent to adds and multiplies. These counts of arithmetic operations are intended to show a general level of complexity for each module with respect to the various parameters available such as vocabulary size and sampling rate. However, in actual implementation many factors come into play such as register allocation, procedure call overhead, cache misses, and loop overhead. These factors can make a particular algorithm run slower than what would be predicted by these simplified computational models. Finally, each module will use some memory for data storage. We will examine the specific memory requirements for each module.

4.2.2.1 Front-End Feature Extraction

The purpose of the front-end feature extraction step is to get parameters from the speech signal that give information about the speaker's vocal tract. These speech observations must be perceptually meaningful and invariant across different speakers.

Table 10: Table of parameters for front-end feature extraction.

Parameter	Description	Typical Values
f_s	Sampling rate (Hz)	8,11, or 16 kHz
N_{fs}	Frames per second	100
T_{window}	Window length	0.025 seconds
N_{frame}	Samples per frame	$f_s \times T_{window}$
N_{FFT}	FFT size	$2^{\lfloor \log_2(N_{frame}) \rfloor}$
N_{fb}	Number of filter-banks	24
L_{avg}	Average filter-bank length	20
N_{cep}	Number of cepstra	13

A vector of mel-frequency cepstral coefficients is the most common feature set used in automatic speech recognition. Dynamic features consisting of first and second temporal derivatives of the cepstrum are also used. A more in depth discussion of the theory and properties of the cepstrum can be found in [23]. In practice, the mel-frequency cepstral coefficients can be computed using the algorithm in Figure 5. This is a basic algorithm that does not take into account more advanced techniques such as cepstral mean normalization.

Arithmetic Complexity Some parameters used in the front-end feature extraction are listed in Table 10. The third column lists some approximate values for these parameters. Based on the algorithm in Figure 5, we can estimate the number of operations required. The formulas to determine the total number of arithmetic operations at each frame are listed in Table 11. The FFT is the dominant computational block for this module, but since the input is real, an $N/2$-point FFT with some post-processing can be used instead of an N-point FFT. Ignoring this post-processing overhead for simplicity, the FFT size, N_{FFT}, is $2^{\lfloor \log_2(N_{Frame}) \rfloor}$ instead of $2^{\lceil \log_2(N_{Frame}) \rceil}$.

Memory Usage The memory requirements for the signal processing front-end are minimal. Some small buffers will be needed to store speech data as it is processed

Table 11: Number of operations to compute MFCC.

Module	Adds	Multiplies	Log
Pre-emphasis	N_{frame}	N_{frame}	0
Window	0	N_{frame}	0
FFT	$6N_{FFT}\log_2 N_{FFT}$	$4N_{FFT}\log_2 N_{FFT}$	0
Magnitude	N_{FFT}	$2N_{FFT}$	0
Mel-spectrum	$(N_{fb})(L_{avg})$	$(N_{fb})(L_{avg})$	0
Logarithm	0	0	N_{fb}
DCT	$(N_{cep})(N_{fb})$	$(N_{cep})(N_{fb})$	0
Dynamic feat.	$2N_{cep}$	0	0

through the pipeline in Figure 5. The bulk of the data storage will be in the form of pre-computed sine and cosine tables for the FFT and filter-bank coefficients for the mel-filter bank operation. This amount of data can typically be stored in less than 100 kilobytes.

4.2.2.2 State Likelihood Computation

A state-of-the-art speech recognizer uses multi-dimensional Gaussian mixture densities to calculate output probabilities for each HMM state. For small vocabulary systems, each state has its own unique mixture density, but large vocabulary systems tend to share parameters across HMM states that are acoustically similar. Each Gaussian component is evaluated using the input speech vector for the current frame. These results are combined with mixture weights from the individual states to produce output probabilities for each HMM state.

The state likelihood step computes the output probability for state k of HMM n:

$$b_{n,k}(\mathbf{x}_t) = \sum_{i=1}^{M} \lambda_{n,k,i} N(\mathbf{x}_t, \mu_{n,k,i}, \mathbf{\Sigma}_{n,k,i}) \tag{16}$$

where $\mathbf{x}_t$ is the input speech vector at time t, $\lambda_{n,k,i}$ is the ith mixture weight for state k of HMM n, and $\mu_{n,k,i}$ and $\mathbf{\Sigma}_{n,k,i}$ are the means and covariances for the corresponding mixture density.

Table 12: Back-end speech recognition parameters.

Parameter	Description
M	Number of Gaussian mixtures
D	Dimension of feature vector
N_m	Total number of HMM models (or words)
B_f	Branching factor
N_s	Numbers of states per model

The multi-dimensional Gaussian, $N(\mathbf{x}_t, \mu_{n,k,i}, \mathbf{\Sigma}_{n,k,i})$, can be calculated with the following:

$$N(\mathbf{x}_t, \mu_{n,k,i}, \mathbf{\Sigma}_{n,k,i}) = \frac{1}{(2\pi)^{\frac{D}{2}}|\mathbf{\Sigma}_{n,k,i}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\mathbf{x}_t - \mu_{n,k,i})^T \mathbf{\Sigma}_{n,k,i}^{-1}(\mathbf{x}_t - \mu_{n,k,i})\right] \quad (17)$$

In an actual implementation, this calculation takes place in the log domain with a diagonal covariance matrix. It can be written as:

$$\log(N(\mathbf{x}_t, \mu_{n,k,i}, \mathbf{\Sigma}_{n,k,i})) = -\log((2\pi)^{\frac{D}{2}}|\mathbf{\Sigma}_{n,k,i}|^{\frac{1}{2}}) - \frac{1}{2}\sum_{j=1}^{D}\left[(\mathbf{x}_t(j) - \mu_{n,k,i}(j))^2 \mathbf{\Sigma}_{n,k,i}^{-1}(j,j)\right] \quad (18)$$

where D is the dimension of the input feature vector.

Finally, the mixture weights are used to scale each individual Gaussian component and the result is summed to produce the output probability for state k of HMM n. In the log domain, this is computed as:

$$b_{n,k}(\mathbf{x}_t) = log\_sum_{i=1}^{M}(\log(\lambda_{n,k,i}) + \log(N(\mathbf{x}_t, \mu_{n,k,i}, \mathbf{\Sigma}_{n,k,i}))) \quad (19)$$

where *log_sum* is the logarithm accumulator operation or the equivalent summation in the log domain. It can be computed by either transforming the input data between the log and linear domains or through a function that utilizes lookup tables and some known log identities. For the purposes of this work, we will assume that this operation is equivalent to a normal math operation.

Arithmetic Complexity The values of $-\log((2\pi)^{\frac{D}{2}}|\mathbf{\Sigma}_{n,k,i}|^{\frac{1}{2}})$ and $\mathbf{\Sigma}^{-1}_{n,k,i}(j,j)$, the inverse of the variances, are fixed during the training phase and can be pre-computed and stored. Assuming that there is no pruning of the search space, the log probability must be computed for each HMM state k and for each model n. This requires $N_m N_s M(2D+1)$ adds and multiplies each as well as $N_m N_s M$ logarithm domain adds.

Memory Usage The means, covariances, and mixture weights for each state are the bulk of the storage used in this step. We also need to store the pre-computed value $-\log((2\pi)^{\frac{D}{2}}|\mathbf{\Sigma}_{n,k,i}|^{\frac{1}{2}})$ from equation 18. Therefore, we will need to store:

$$Mem_{SL} = N_m N_s M(2D+2) \tag{20}$$

words for the means, inverse variances, mixture weights, and pre-computed values. It was shown in [85] that some of these values can be stored with 8-bits of precision without loss of accuracy.

4.2.2.3 Viterbi Search

The Viterbi search is a well known dynamic programming algorithm used to find the best path through the set of possible HMM states. The general order of complexity is $O(N^2T)$, where N is the total number of states, and T is the total number of frames. There are N^2 possible transitions for an HMM with N states. However, with HMMs used in speech recognition, many state transitions have zero probability. Figure 6 shows a typical left-to-right HMM topology used in speech recognition systems. Such an HMM has only $3(N-1)$ non-zero state transitions.

In a connected word speech recognizer, a composite HMM is constructed from all N_m models. Each state in each model is evaluated, but the total number of allowed arcs is greatly reduced. There are two types of arcs to consider in this composite HMM. There are intra- and inter-word transitions. Intra-word transitions occur within a word and are indicated by solid black lines in Figure 7. The dashed

lines are inter-word transitions and do not consume a frame of input. The language model probabilities (if any) can be used here to weight the cost of transitions between words. There are $N_m \times 3(N_s - 1)$ intra-word transitions. There are N_m^2 inter-word transitions if we allow every word to follow every other word. At each frame, all arcs must be evaluated and partial path scores must computed. A more formal discussion of the Viterbi algorithm can be found in [41].

Arithmetic Complexity For brevity, we have omitted the update equations for the Viterbi algorithm, but we consider both types of transition arcs, word insertion penalties, and language model weights. There are $N_m(9N_s + 2(B_f \times N_m))$ adds and $2N_m(B_f \times N_m)$ multiplies in the basic Viterbi search, where B_f represents the branching factor of the language model. That is, what fraction of vocabulary words can follow one another. In practice, medium to large vocabulary systems use a beam pruning heuristic with the Viterbi algorithm to reduce the number of active states at any given time frame. States whose partial path scores do not fall within some threshold of the lowest cost path are eliminated from consideration. This will reduce the overall size of the Viterbi search but will add some overhead due to the threshold comparisons. We have not considered beam pruning heuristics in this paper.

Memory Usage The storage required in this stage is a combination of the language model and the state transitions for the HMM acoustic models. In medium to large vocabulary systems, the language model can use anywhere from several kilobytes to tens of megabytes of storage depending on the complexity of the task and the vocabulary size. For a simple bigram or finite state grammar that might be used in a small vocabulary application such as this, the storage requirements for the langauge model are usually minimal. The number of state transitions is dependent on the model topology and total number of states available in the acoustic model. If the transition probabilities are stored as 4 byte floats or 4 byte fixed-point numbers, then

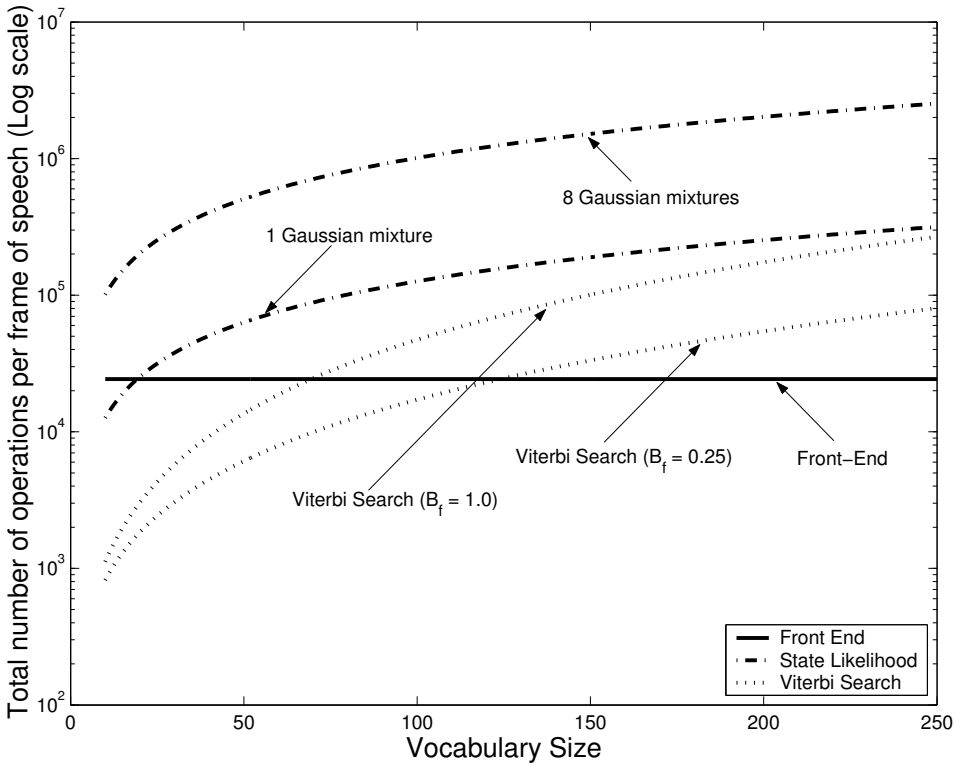


Figure 26: Arithmetic complexity per frame of speech for connected word speech recognition. (Y-axis is logarithmic).

the total amount of memory required to store them is:

$$Mem_{VS} = N_{NZ} \times N_s \times N_m \times 4 \tag{21}$$

where N_{NZ} is the total number of non-zero state transitions allowed by the HMM topology. For the configuration in Figure 6, $N_{NZ} = 3(N_s - 1)$. If we have 8 states per HMM, then we need 672 bytes per model (or word). There will also be some amount of working memory required to store all of the path scores and best paths from the start state until the current state.

4.2.3 Summary

In this section, we have estimated the arithmetic complexity of a small vocabulary connected word speech recognizer. We have assumed a simple language model with a varying branching factor. The equations presented represent the general arithmetic

complexity to process one frame of speech data. Although they ignore much of the overhead involved in implementation on a general purpose processor, they still show how the problem scales with varying parameters. Figure 26 shows the complexity of the modules with respect to vocabulary size with some typical speech recognition parameters. The Y-axis is plotted on a logarithmic scale. The complexity of the front-end is independent of vocabulary size. The Gaussian evaluation involved in the ouptut probability computation (or state likelihood) requires the bulk of the computation. The Viterbi search can scale quadratically with vocabulary size, but the slope is determined by the language model branching factor, B_f, which determines how many inter-word transitions must be considered per frame.

This model suggests that distributing the speech recognition across a network by performing feature extraction/compression on the mobile device and HMM evaluation/Viterbi search on the server is an attractive alternative for resource constrained devices. We examined the energy consumption of the feature extraction algorithm on a StrongARM based platform in [18]. Based on our findings, the energy consumption for the feature extraction calculation plus compression and wireless transmission will likely be less than the energy required for the remaining HMM evaluation and Viterbi search to be compututed locally for all but the smallest vocabulary sizes. That is, the entire system will use less battery energy in a distributed speech recognition configuration than simply performing the entire task locally. This suggests that distributed speech recognition is the best solution when considering battery life as the sole optimization criterion.

4.3 Large Vocabulary ASR

Large vocabulary continuous speech recognition (LVCSR) is a complex task requiring large amounts of both memory and CPU usage. For the time being, high-quality large vocabulary continuous speech recognition is not feasible on a wireless mobile device.

In this section, we discuss some of the requirements for LVCSR and how mobile devices might be able to meet these requirements in the future. In section 4.3.4 we introduce a simple model of energy consumption on StrongARM based systems for client-side ASR of lesser quality.

There have been several studies of the performance of speech recognition systems on modern processors in the literature. There are three main issues discussed in the literature: computation, memory size, cache architecture, and instruction level parallelism. On modern day processors capable of clock speeds in the gigahertz range, computation is rarely an issue, but other factors such as cache architecuture and parallelism are important.

4.3.1 Computation

A study of the Sphinx II ASR system in [3] reported that 100 MIPS is required for real-time performance. However, the Sphinx II ASR system uses less accurate semi-continuous HMMs for it's acoustic modeling. We can expect that systems based on fully continous HMMs would require more computation. In [54] it was shown that the most frequently accessed computational kernels were of the form $((a-b)^2)c$ and $\log(1+\exp(x))$. The former is used in the kernel of the Gaussian probability evaluation, where a is the input vector, b represents the mean, and c is the inverse of the variance. This operation is the opposite of the fast multiply accumulate operation, which is available on many DSP chips and embedded processors including the StrongARM. The latter expression is used in the accumulation of mixture density probabilities, and is typically calculated via a polynomial expansion in the floating point hardware. This operation is very slow on systems without floating point hardware, but it can be implemented via a series of lookup tables and other approximations. While many of the Viterbi search computations can be carried out using fixed point arithmetic, the Gaussian evaluation step requires simultaneous dynamic range and precision that

fixed point arithmetic cannot deliver. In [64], it was shown that light-weight floating point routines requiring only a 12-bit mantissa and 8-bit exponent were sufficient for Gaussian evaluation. (The IEEE 754 format requires a 23-bit mantissa.)

Table 13: Cycle counts for the front-end, Gaussian evaluation, and Viterbi search portions of speech recognition.

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

The breakdown of computation for the Sphinx III ASR system (based on fully continuous HMMs) is shown in Table 13. The results were obtained on a 1.4 GHz Pentium 4 workstation. The total processing for the front-end is less than one percent of the overall computation, with the majority of time being spent in the hidden Markov modeling step. These results are similar to those reported in [54] for the commercial speech recognition system studied. However, depending on the input data and ASR task, the percentage of cycles spent in Gaussian evaluation can be as high as 80%. The search algorithm involves the Viterbi search, language modeling, beam pruning, and lexical tree building, and most often involves a smaller portion of the overall cycle time.

4.3.2 Memory Heirarchy

The total amount of memory used for the acoustic models in the Sphinx III ASR system is approximately 26 Mbytes. This includes the means, variances, and mixture weights of the HMM states as well as the transition probablilties and state tying information. Systems based on semi-continuous HMMs, such as Sphinx II, have few parameters and will typically use less memory for acoustic models. A typical HMM state can use between 2-4 Kbytes for its various parameters. A large vocabulary n-gram language model can easily use 100 Mbytes of memory. For example,

a trigram language model trained on broadcast news transcriptions contains 13 million trigrams, 9 million bigrams, and 64,000 unigram probabilities, and occupies 125 Mbytes. Smaller language models based on bigrams will be less accurate but can be significantly smaller in size. For example, a bigram language model for a Wall Street Journal read text task contains approximately 800,000 bigrams and 5,000 unigrams and occupiess less than 10 Mbytes. The size of the language model is dependent on the complexity of the task, vocabulary size, and amount of training text. It also effects the overall computational effort.

During the decoding phase, a speech LVCSR recognizer has three main memory working sets: the acoustic models, the language models, and the search buffer. The search buffer is reported to occupy tens of Mbytes of memory in [54]. These three working sets often compete for cache space in the memory hierarchy of the computer. A typical modern workstation has a three level memory hierarchy consisting of fast but small, on-chip L1 cache, a larger and slower L2 cache, and main memory, which is the slowest of the three. The hard-disk can also be used as a memory device, but it is too slow to be used frequently. The authors in [54] report a speedup by a factor of 2 when doubling the main memory size from 128 Mbytes to 256 Mbytes, presumably from not having to use the hard disk for memory access. The memory access patterns of the HMM phase have some spatial locality but little temporal locality. That is, memory accesses were frequently sequential across the entire HMM parameter space but rarely used more than once before being flushed from the cache. The search phase of the recognition has a more random data access pattern due to the beam pruning heuristic. A study of the optimum cache architecture was performed in [64], [3], and [54] for various speech recognition systems. The required memory bandwidth was reported to be larger than 150 Mbytes/s for the Sphinx II system and almost 800 Mbytes/s for the Sphinx III system. Larger L2 cache sizes can help reduce this requirement to the slower main memory, but all authors reported that L2 cache sizes

of at least 8 Mbytes were required to reduce the L2 cache miss rate to below 10%. Many embedded devices typically do not even have an L2 cache, and a StrongARM based system typically has peak memory bandwidth of around 64 Mbytes/s. Larger cache line sizes were also beneficial as it allowed an entire HMM parameter set to be loaded in one line.

4.3.3 Parallelism

Data dependencies will reduce the amount of parallelism that can be obtained from a sequence of instructions. The tight coupling between the Viterbi search and HMM evaulation makes it difficult to extract parallelism at a higher level. In [3], thread level parallelism was used to produce a speedup of approximately 3.5. The iteration over individual HMM states was parallelized, which allowed the hardware to effectively use multiple data paths more efficiently. A high percentage of resource related stalls was reported in the back-end search, indicating memory or register dependencies within the algorithm. Larger cache sizes and an increased number of arithmetic units can help reduce these stalls and increase the number of instructions per cycle. A Gaussian evaluation accelerator was introduced in [64], where the Gaussian evalaution can be carried out in parallel for a large number of states. This coprocessor is fed a set of means, variances, and features from memory and can process them in parallel with the Viterbi search from the previous frame.

4.3.4 A Model of Energy Used in Computation for Client-Side ASR

An overview of power consumption for various speech recognition implementations is shown in Figure 27. Application specific integrated circuits (ASIC) can provide the lowest levels of power consumption. Embedded processors and DSP chips use more power, but the complexity of the speech recognition task is generally limited. Desktop machines based on modern superscalar processors typically use several tens of watts.

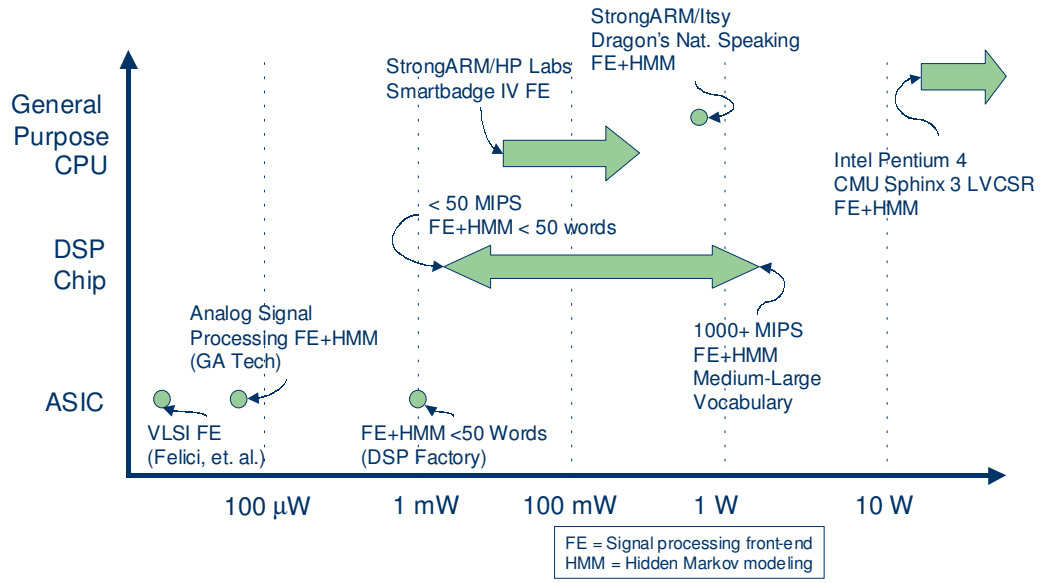


Figure 27: Power consumption figures for various speech recognition hardware/software configurations.

As we have seen, the computation of speech features is a small portion of the overall speech recognition task in both computation and memory usage. Client-side ASR requires more computation and memory bandwidth due to the back-end search algorithm. Porting a full speech recognition sytem to a mobile device requires more optimization than a simple conversion to fixed-point arithmetic. It involves optimization at many levels, from search space reduction to fast arithmetic kernels and techniques to reduce memory bandwidth. For these reasons, we concentrate our software optimization on the signal processing front-end only, and estimate the full client-side ASR energy usage by using some published results [37].

In the absence of a network connection it may be necessary to perform ASR on the mobile device. Speech recognition engines have been optimized for the StrongARM or other mobile processors by many industry players, but it has been shown that they use most available resources and may run several times slower than real-time for many tasks. Power measurements for an embedded dictation ASR system running on a StrongARM based processor are given in [37]. The ASR system ran just over

2.5 times real-time, and the processor was almost never idle during the task.

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

$$E_{local} = (P_{cpu} + P_{mem}) \times R \times \frac{1}{100} \tag{22}$$

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

CHAPTER V

REDUCED ENERGY CONSUMPTION FOR DSR IN WIRELESS NETWORKS

In this chapter we address the issue of energy consumption of the wireless interface in a distributed speech recognition system. As we have discussed earlier, the wireless interface can occupy almost half of the energy budget on many mobile wireless devices. We introduce techniques to minimize the energy consumption required to transmit speech parameters to an ASR server. In Section 5.1, we model the energy consumption of a DSR system using both the IEEE 802.11b and Bluetooth wireless interfaces. By employing synchronous bursty transmission of speech parameters, we can maximize the amount of time spent in a low power state or off state while adding virtually imperceptible delay to the application. Using this technique, we can significantly reduce the energy consumption required for transmission. We explore these tradeoffs with respect to latency, channel conditions, and energy consumption in Section 5.2. These techniques can provide reductions in energy consumption of over 90% compared to a software based client-side ASR system.

The embedded system used in the experiments is the SmartBadge IV developed at the Mobile and Media Systems Lab at HP Labs [62]. The SmartBadge contains a 206 MHz StrongARM-1110 processor, StrongARM-1111 co-processor, Flash, SRAM, PCMCIA interface, and various sensor inputs such as audio, temperature, and accelerometers. It runs the Linux operating system. The SmartBadge has speech/audio driven I/O, so speech recognition can provide some level of user interaction through a voice-user interface. It supports a variety of different networking hardware options

including Bluetooth and 802.11b wireless interfaces. It has high-quality audio input suitable for speech recognition. The StrongARM platform is still used in many high-end PDAs on the market today, such as the HP iPAQ H3800. Table 14 shows the total average power dissipation of the iPAQ with both 802.11b and Bluetooth transmitting data as well as without the network. The SmartBadge IV uses the same memory and

Table 14: Engery Consumption of the HP iPAQ.

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

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.

5.1 Modeling the Energy Used in Communication

The wireless network can use significant amounts of energy on a mobile device. Measurements on the SmartBadge IV hardware show that an 802.11b interface card can use up to 45% of the total power budget. Reducing the energy consumption is an important consideration and has been well studied in general. Section 2.5.1 outlines some of the techniques. We consider both 802.11b and Bluetooth wireless networks in our analysis. We assume single hop communication with a speech recognition server connected to a wired network via a wireless access point. Multi-hop communication has limited utility for this application as it is a client-server scenario over limited range wireless links. Distributing the computation across a set of equally constrained

mobile devices is not considered here.

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

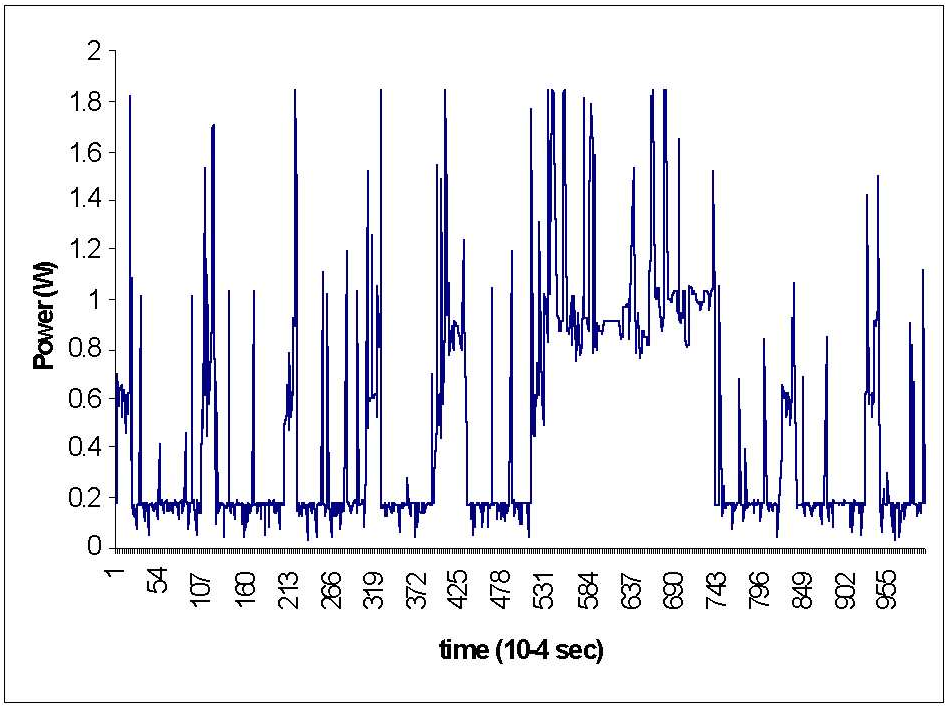
In order to estimate the power consumption for wireless transmission, we directly measure the average current into the network interface. These measurements are performed under ideal conditions with no competing mobile hosts or excessive interference. Using these measurements as a baseline, we are able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

5.1.1 802.11b Wireless Networks

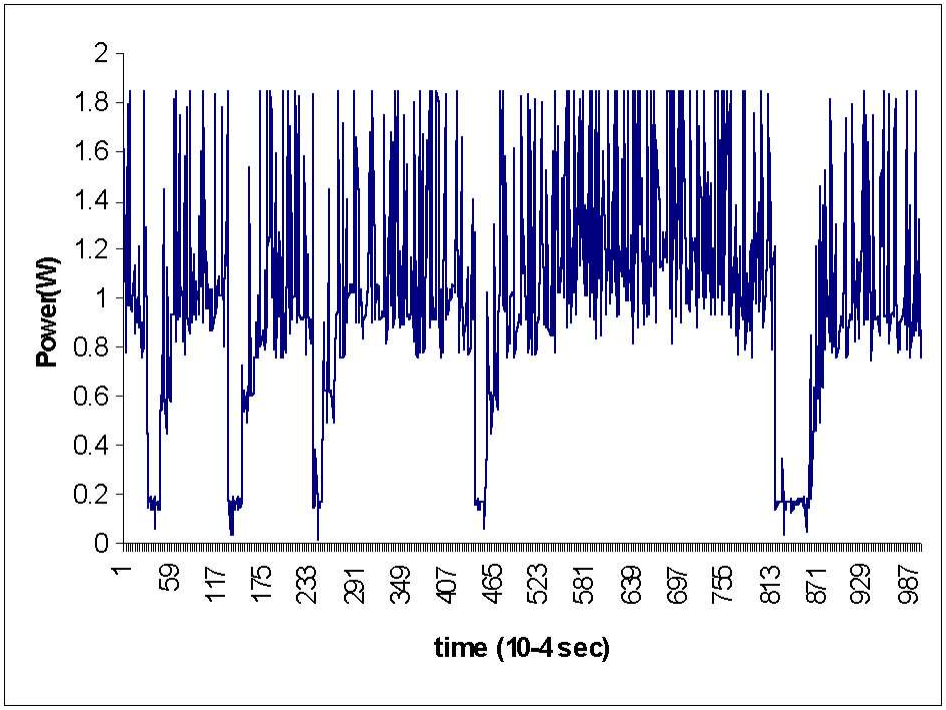
The 802.11b interface operates at a maximum bit rate of 11 Mbps with a maximum range of 100 meters. The MAC protocol is based on a carrier sense multiple access/-collision avoidance scheme, which includes a binary exponential backoff system to

avoid collision. It uses an automatic repeat request (ARQ) system with CRC error detection to maintain data integrity. We used a PCMCIA 802.11b interface card and measured the average current going into the interface to get the power dissipation.

Our measurements indicate there is only a difference of a few mW in power consumption between the highest and lowest bit rates. This is expected since the bit rates are low, and the transmit times are very short. Also, the use of UDP/IP protocol stacks and 802.11b MAC layer protocols both add significant overhead for small packet sizes. The 11 Mbps WLAN interface is 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. This increases the total delay of the system, but less battery energy is used since the various networking overhead is amortized across a larger packet size. However, due to the relatively high data rates provided by 802.11b, the WLAN interface spends most of its time waiting for the next packet to transmit. The 802.11b PM mode can provide some savings in energy consumption but this does not hold under heavy broadcast traffic conditions [2], defined as a higher than average amount of broadcast packets. In addition, the PM mode is not available in the adhoc (as opposed to infrastructured) topography. We present an on/off scheduling algorithm to reduce the total energy consumption of the 802.11b device under these conditions. While operating in the 802.11b power management mode, a WLAN card goes into an idle state. Every 100ms it wakes up and receives a traffic indication map, which is used to indicate when the base station will be transmitting data to this particular mobile host. With heavy broadcast traffic, the WLAN interface will rarely be in the idle state and it will consume power as if it were in the always-on mode. This is because the time required to analyze the broadcast packets is larger than the sleep interveral. This increase in power consumption will happen even if there are no applications running on the mobile host. Figure 28 shows the power consumption of the WLAN card in the 802.11b power management mode



(a) light traffic



(b) heavy traffic

Figure 28: WLAN power consumption in 802.11b PM mode in light and heavy traffic conditions.

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. Measurements in [2] indicate that even in less than average amounts of broadcast traffic, energy is wasted by the extra processing.

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

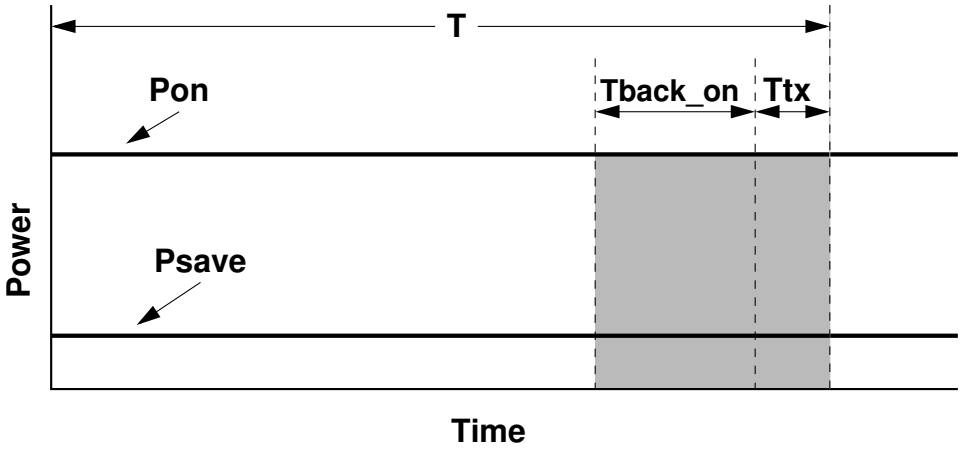


Figure 29: The timing of the 802.11b scheduling algorithm.

Figure 29 shows the timing of this scheduling algorithm. The period, T, is determined by the number of speech frames sent in one packet. The transmission is synchronous such that every T seconds we will send that amount of compressed speech features and stay in the off state for the remainder of the time. With larger values of T we can hope to amortize the cost of turning the WLAN card on and off at the expense of longer delay. Assuming that a speech recognizer server is able to process speech at or near real-time, the user will experience delay near the value

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

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

$$E_{on} = P_{on} \times T \tag{23}$$

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

$$E_{save} = P_{save} \times T \tag{24}$$

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

Using the proposed scheduling algorithm, the WLAN card will be on only during the shaded region in figure 29. The value, T_{back_on}, is the amount of time required to turn the WLAN card back on, during which time it uses power as if it were transmitting. The value T_{tx} is the total amount of time required to transmit the data, which is typically much smaller than T_{back_on} for the low bit rates required for speech traffic. The energy required to transmit under the proposed scheduling

algorithm is:

$$E_{sched} = P_{on} \times (T_{back\_on} + T_{tx}) \tag{25}$$

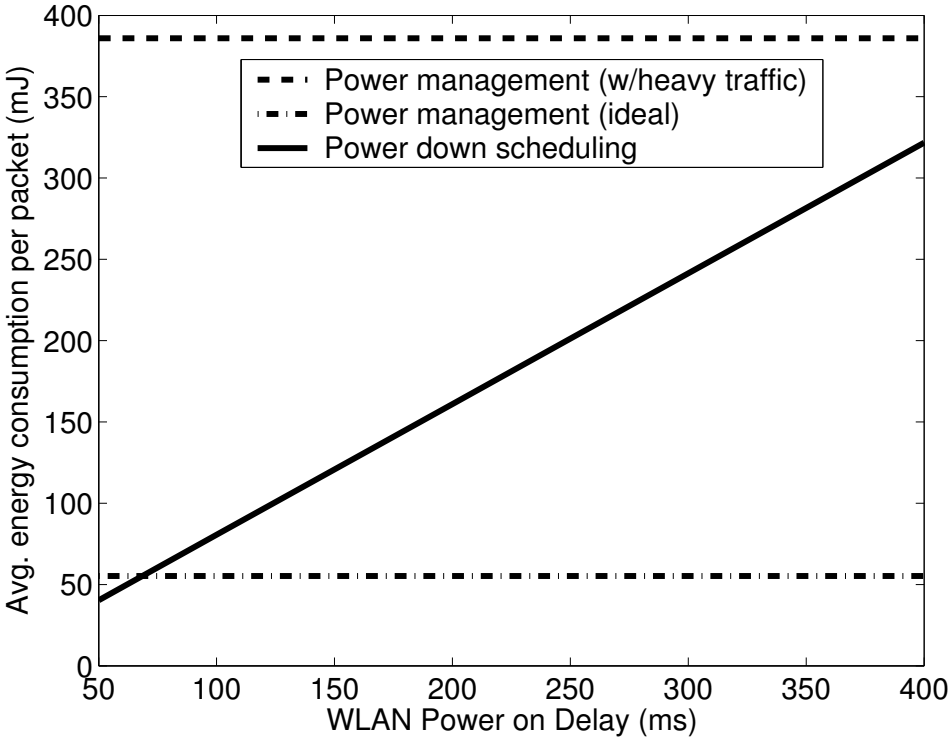


Figure 30: WaveLAN power on delay vs. energy consumption per packet.

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 30 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 30). This is a reduction in energy consumption by about 80%. However, this only holds true for heavy broadcast traffic conditions, so the mobile device will have to monitor the broadcast traffic and decide between the standard 802.11b PM mode or the scheduling algorithm.

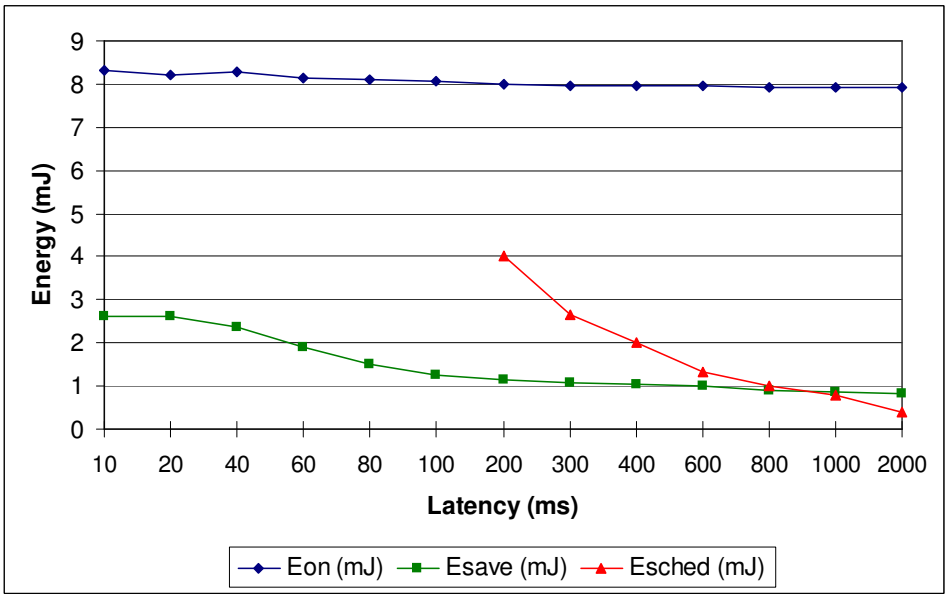


Figure 31: Average energy consumption per 10ms speech frame vs. DSR latency for various 802.11b power save schemes. (WLAN power on delay is fixed at 100ms.)

Finally, we consider increased delay or latency, T, in Figure 31. with T_{back_on} fixed at 100ms. In this plot, the energy cost was determined using measured values of power consumption. The energy cost has been normalized to show the average energy required to transmit one frame of speech data. As the total number of frames approaches 80 ($T = 800ms$), we can see that the scheduling algorithm (E_{sched}) will be able to outperform the PM mode configuration (E_{save}) regardless of traffic conditions. This will result in less than one second of delay for a user interface application with speech recognition. Shorter power on (T_{back_on}) times can help move this crossover point to shorter delays. Longer delays of two seconds or more can further reduce energy consumption and are good candidates for applications requiring lower interactivity such as dictation.

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

$$P_r = 1 - (1 - BER)^L \tag{26}$$

where L is the packet length, and BER is the bit error probability for the current channel conditions. For our analysis, we used the BER probability for 256-QAM modulation in a Rayleigh fading channel to approximate the 802.11b CCK modulation. A Rayleigh fading channel models the effect of time-varying multipath fading of the received signal by accounting for constructive and destructive interference of the scattered carrier signal. The Rayleigh fading channel asssumption is widely used in wireless communications literature as a more realistic alternative to an additive white Gaussian noise channel [78]. The BER expression for the 256-QAM modulation is:

$$BER = \frac{2^{k-1}}{2^k - 1} \sum_{m=1}^{M-1} \frac{(-1)^{m+1} \binom{M-1}{m}}{1 + m + 2m\bar{\gamma_b}} \tag{27}$$

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

Given the probability of retransmission (P_r), the expected number of retransmissions (T_r) is given by [115]:

$$T_r = \frac{1}{1 - P_r} \tag{28}$$

Using these equations, an energy model can be constructed that incorporates the energy used in the MAC overhead as well as the energy required for repeated retransmissions, assuming the average SNR remains the same. Such an energy model is presented in [27] and is summarized here:

$$E_{tx}(BER, L) = E_{aq} + T_{ack} \times P_{rx} + (E_{aq} + T_{tx} \times P_{tx}) \times \frac{1}{(1 - BER)^L} \tag{29}$$

where E_{aq} is the average energy required to acquire the channel, T_{ack} is the time required to receive the ACK packet, and P_{rx} is the receive power for the robust control channel. Given this energy model, we can incorporate it into our scheduling algorithm model in (25) as follows:

$$E_{sched} = E_{tx}(BER, L) + P_{on} \times T_{back\_on} \tag{30}$$

We use this expression in Section 5.2 to quantify the energy consumption of 802.11b vs. channel SNR. In particular, we show how larger packet sizes and lack of error correction techniques force 802.11b to operate in higher channel SNR. However, techniques such as packet fragmentation and error correction can be used to extend the lower SNR range of 802.11b.

5.1.2 Bluetooth Personal Area Network

The Bluetooth personal area network provides a maximum bit rate of 1 Mbps, and a variety of different packet types are available to support different traffic requirements [110]. It supports a range that is considerably less than 802.11b, on the order of 10 meters. Bluetooth supports both data and voice traffic packets as well as a hybrid packet containing both voice and data. Media access is handled via a time-division duplex (TDD) scheme where each time slot lasts 625 μseconds. Voice packets are given priority over data packets in scheduling. In this work, we consider only pure voice or pure data packets. Data packets are available in both high-rate and medium-rate packets. These are DHn or DMn packets for both high and medium data rate respectively, where n depicts the number of TDD slots the packet occupies: 1, 3, or 5. High-rate packets use a stop-and-wait automatic-repeat-request (ARQ) protocol with CRC error detection within the packet. Medium-rate packets use a 2/3 rate (15,10) shortened Hamming code in addition to the ARQ protocol. Voice packets, due to their time-sensitive nature, do not use an ARQ protocol. Voice packets are

available in HV1, HV2, or HV3 types, where the number denotes the amount of error correction rather than slot length. All voice packets occupy one TDD slot with varying data payloads. HV3 packets use no error correction. HV2 packets use the (15,10) Hamming code, and HV1 packets use a 1/3 rate repetition code. Given the soft time deadlines with speech data intended for a machine listener, we can easily use either data packets or voice packets without consideration of packet jitter or delay characteristics.

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

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

$$E_{HVn} = P_{tx} \times T_{tx} = P_{tx} \times 625\mu s \tag{31}$$

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

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

$$E_{Dxn} = (P_{tx} \times 625\mu s \times n) \tag{32}$$

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

Using power measurements of a USB Bluetooth device attached to the SmartBadge IV, we are able to estimate the energy usage for our system. Figure 32 shows the energy required to transmit one frame of speech data at various DSR compression rates over a Bluetooth link. We consider the use of both high-speed and medium-speed data packets. We assume an error-free channel with no retransmissions. We can see in figure 32 that there is a higher energy cost for medium-rate packets due to the FEC overhead. However, these packets will be a better choice for lower SNR conditions. Energy consumption approximately doubles between the 1.2 kbps and 4.2 kbps bit rates. However, these estimates do not consider idle time between packets that will consume energy as well.

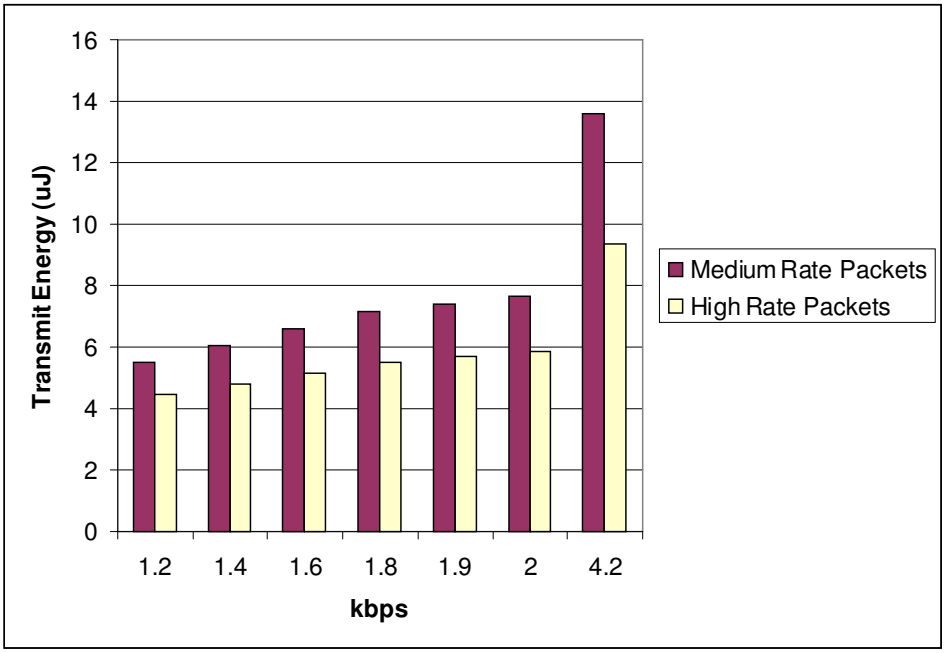


Figure 32: Energy used to transmit one frame of speech with varying compression rates for Bluetooth radio.

We can incorporate the Bluetooth power saving modes into our model to account for the idle time in between packets. A node within a Bluetooth piconet can operate in a variety of different power management modes [110]. In the default *active* mode, the slave node listens to every master-slave slot to see if the packet is addressed

to it. In the *sniff* mode, the node only listens to slots at specified intervals. In *hold* mode, the node goes into a low-power state until some specified interval, after which it powers up to transmit. In *park* mode the Bluetooth node temporarily gives up its membership to the piconet to join a list of parked nodes. The node's only activity in parked mode is to periodically listen for synchronization and broadcast packets. Figure 33 shows the state transition times and power measurements for the Bluetooth energy saving modes. The transition times are shown on the arcs, and the power measurements are inside the states. During a transition, the Bluetooth device consumes power according to the state it is leaving.

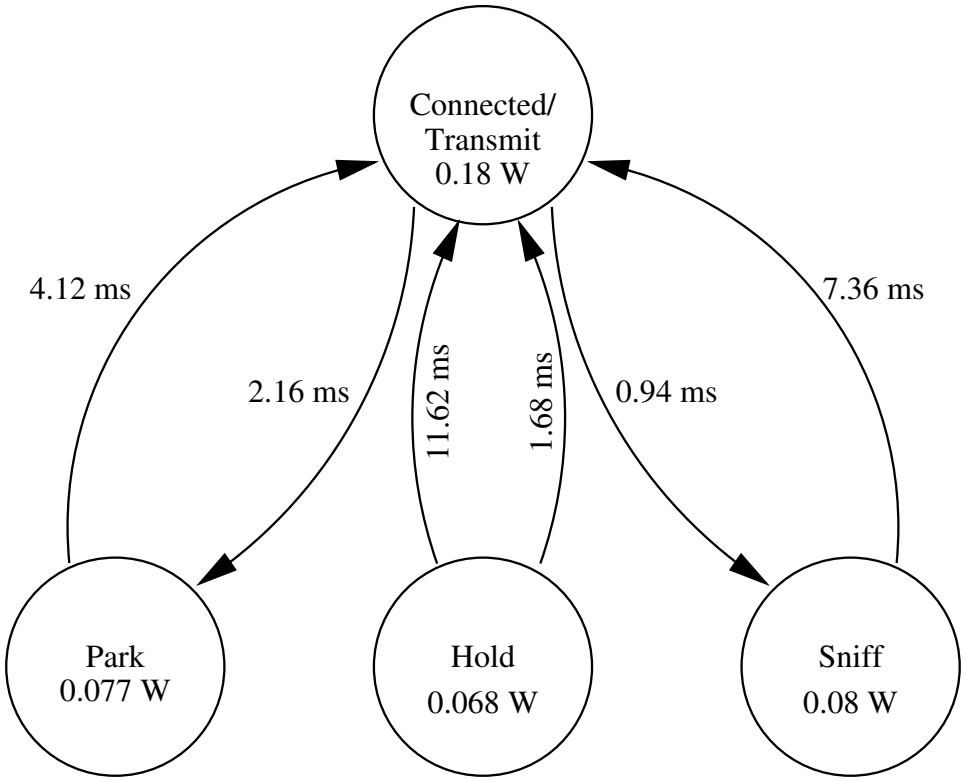


Figure 33: Transition times and power measurements for Bluetooth energy saving modes.

In the case of an asymmetric client-server network application, the Bluetooth network was shown to be most energy efficient when the node sending the most data is configured as the slave [121]. This is due to the asymmetric behavior of the ARQ

scheme in Bluetooth. In this configuration, the slave node never retransmits packets to the other host. In our case, the slave node will be the mobile device, which is sending audio data to an automatic speech recognizer. This configuration also allows the mobile device to enter a low-power state. For our analysis, we will use the *park*, since it provides competitive transition times as well as the ability for a master node performing multiple speech recognition requests to support more nodes.

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

$$E = P_{tx} \times T_{tx} + E_{transition} + P_{park} \times T_{park} \tag{33}$$

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

By varying the amount of data transmitted at once, we can increase the amount of time spent in the park state. Figure 34 shows the tradeoff between speech recognition latency and energy consumption per frame of speech for both 802.11b and Bluetooth. Once again, we assume a perfect channel with no bit errors. For smaller values of T, Bluetooth can offer better performance than 802.11b, but as T approaches 1.3 seconds, 802.11b will use less energy. This is because the 100 ms startup cost of 802.11b is amortized across a larger number of frames, while the Bluetooth node remains in the park state and still consumes power. Powering off a Bluetooth node between packet transmissions is not an option since the paging/inquiry actions required to join a piconet can easily take in excess of 10 seconds. However, since the transition time from park to active and back is small, we see an initial drop in energy consumption

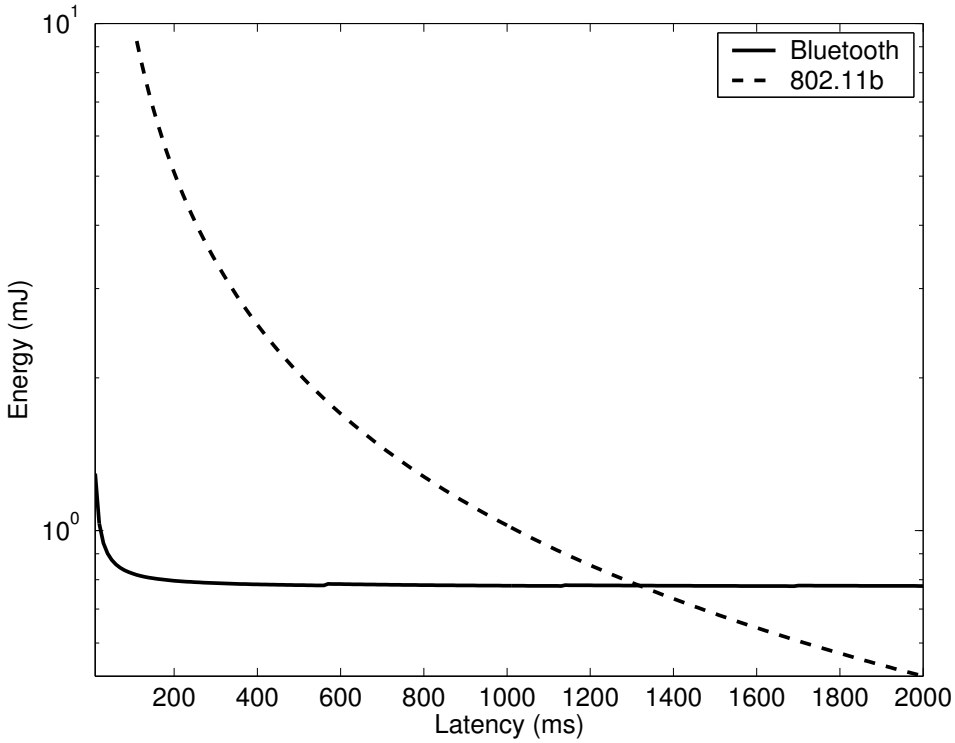


Figure 34: Energy per frame of speech vs. DSR latency for a Bluetooth and 802.11b.

with respect to increased latency in Bluetooth. However, with increased delay the energy spent in park mode becomes the dominant factor.

Next, we investigate how the presence of bit errors on the wireless channel will affect both the energy consumption and, in the case of voice packets, speech recognition accuracy. We use this data to identify which types of packets can be used effectively in various channel conditions. The main difference between the two types of packets is that voice packets rely only on FEC and no ARQ, while data packets can use *both* FEC and ARQ. The energy consumption of Bluetooth voice packets is independent of channel conditions. Therefore, we can estimate the energy consumption using (32) and (33). The main difference in energy consumption per frame of speech will come from the reduced user payload due to FEC bits.

In the presence of bit errors, data packets will continue to be retransmitted until they are received correctly or a timeout occurs. For the purposes of this analysis, we

assume BFSK modulation with coherent detection under a Rayleigh fading channel. We also assume that the average SNR remains constant throughout the transmission. The BER expression used is as follows:

$$BER = \frac{1}{2}\left(1 - \sqrt{\frac{\bar{\gamma_b}}{2 + \bar{\gamma_b}}}\right) \tag{34}$$

where $\bar{\gamma_b}$ is the average SNR per bit. The total energy used is also a function of the probability of a packet retransmission. The expression is based on the probability of packet synchronization failure, header failure, payload error, and both synchronization and header failure in the ACK packet. Each of these items is a function of the bit error rate (BER), which is, in turn, a function of the channel signal to noise ratio (SNR). An expression for this probability (P_r) is derived in [104] and will be summarized here. Packet retransmission will occur when any of the following events occurs:

A: Synchronization failure of outbound packet.

B: Header corruption on outbound packet.

C: Payload error on outbound packet.

D: Synchronization failure on ACK packet.

E: Header corruption on ACK packet.

Therefore, the probability of a packet retransmission is as follows:

$$P_r(\gamma) = 1 - P[\bar{A}]P[\bar{B}]P[\bar{C}]P[\bar{D}]P[\bar{E}] \tag{35}$$

where γ is the instantaneous signal-to-noise ratio, $\bar{A}$ is the complement of event A, and $P[A]$ is the probability of event A. Expressions for the individual probabilities of events A through E follow. The probability of successful synchronization, events A and D, are identical and defined by:

$$P[\bar{A}] = P[\bar{D}] = \sum_{k=0}^{72-T} \binom{72}{k} \epsilon(\gamma)^k (1 - \epsilon(\gamma))^{72-k} \tag{36}$$

where T is the threshold of bits that must be received correctly for successful synchronization and $\epsilon(\gamma)$ is the probability of bit error for channel SNR, γ. The probability of the header success, events B and E is:

$$P[\bar{B}] = P[\bar{E}] = (3\epsilon(\gamma)(1-\epsilon(\gamma))^2 + (1-\epsilon(\gamma))^3)^{18} \tag{37}$$

Finally, the event of a payload success for high rate packets it given by:

$$P[\bar{C}] = (1-\epsilon(\gamma))^m \tag{38}$$

where m is the payload size in bits which varies depending on the packet size (DH1, DH3, or DH5). For medium rate packets, the 2/3 Hamming code is capable of correcting one error per 15 bit block. The probability of successful transmission for medium rate packets is:

$$P[\bar{C}] = (15\epsilon(\gamma)(1-\epsilon(\gamma))^{14} + (1-\epsilon(\gamma))^{15})^{\frac{m}{15}} \tag{39}$$

If we ignore the overhead for receiving an ACK packet, an estimate of the energy consumption for Bluetooth data packets in the presence of bit errors is:

$$E_{Dxn} = P_{tx} \times 625\mu s \times n \times \frac{1}{1-P_r} \tag{40}$$

where P_r is the probability of a retransmit for the appropriate packet type. By dividing the energy by the number of frames in a packet, which varies with packet length and coding technique, we can get the energy required to send one frame of speech.

5.2 Summary of DSR Tradeoffs

We have provided energy models for both 802.11b and Bluetooth wireless networks for distributed speech recognition traffic. The two main variables of interest are the total delay, T, and the average channel SNR. First, we analyze Bluetooth and 802.11b energy consumption with respect to these variables individually, and then we compare their performance with the addition of front-end processing and full client side ASR.

5.2.1 Bluetooth Data Packets and IEEE 802.11b

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 15. For the remainder of this analysis, we consider transmission at the highest available bit rate, which offers the best WER.

Table 15: 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

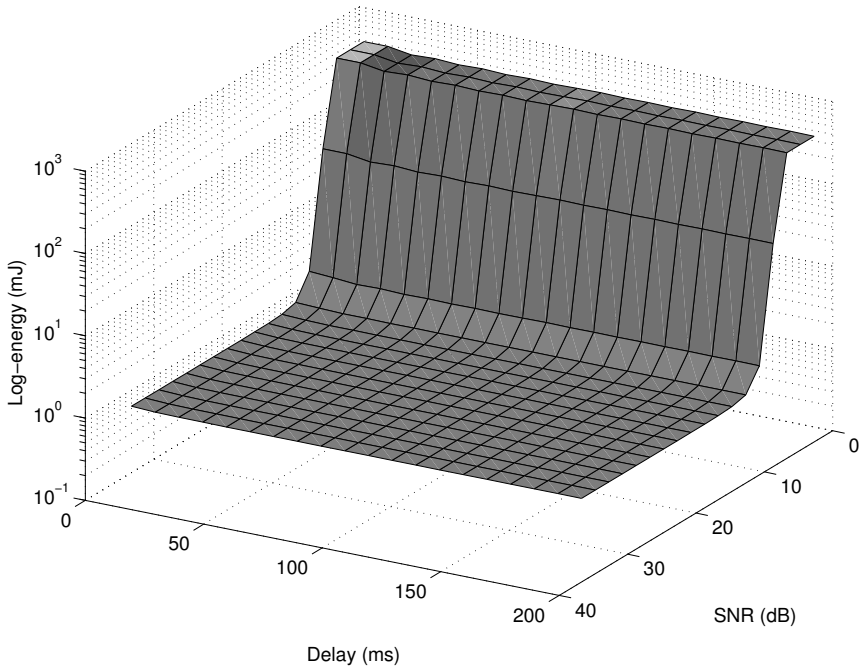
In Figure 35 and Figure 36 we show the energy consumption of two kinds of Bluetooth data packets, DM1 packets and DH5 packets respectively. In plot (a), we show the energy consumption with respect to SNR and delay on the X and Y axes, while the energy consumption is plotted on the Z axis. DM1 packets have the smallest payload and are the most robust packet type due to the error correction coding. Because each packet has such a small payload, the energy consumption is almost entirely independent of the system delay. In plot (b), we show the contour of the surface projected on to the delay-SNR axis. Darker lines represent areas of lower energy consumption. In Figure 35a, the contour lines are nearly vertical. In Figure 36a, we can see a slight effect of the total delay, T. The stair-stepping effect

is due to the DH5 packets, which have the largest usable payload, not being entirely utilized. Except for small delays, which result in under-utilized packets, the energy consumption per frame of DH5 packets is also independent of delay, but DH5 packets require higher SNR due to the lack of error correction coding and the longer packet size, which increases probability of a packet error.

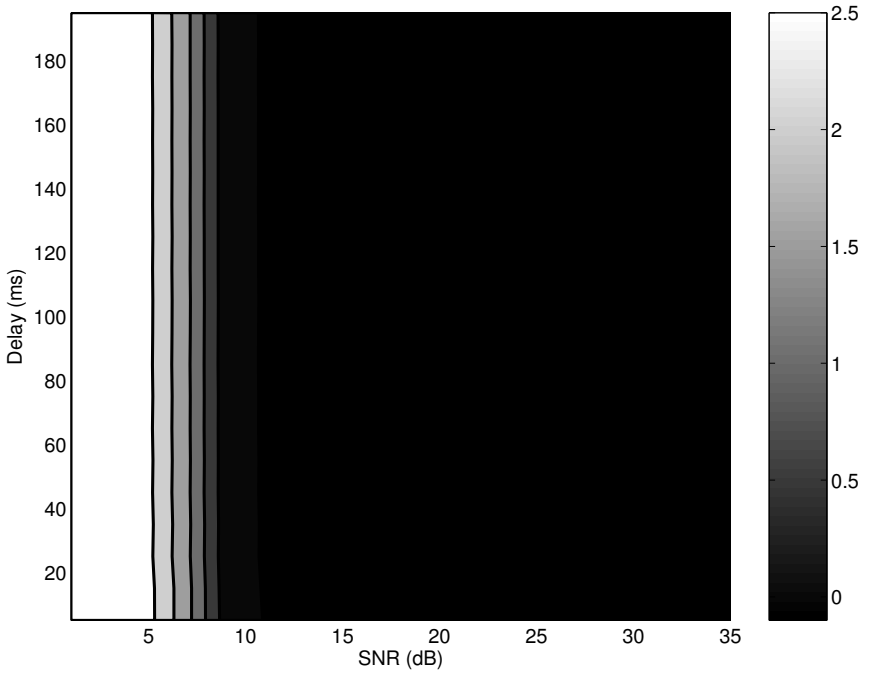
In Figure 37 we show the same delay vs. SNR tradeoff with 802.11b packets. 802.11b packets are variable length, up to a maximum size. At 4.8 kbps, even 2 seconds of speech data can fit within one packet. However, this larger packet size increases the chances of packet error. Therefore, the energy consumption per frame of speech with 802.11b packets is dependent on both delay and SNR. Figure 37b shows this relationship. Assuming a fixed delay (and packet size), a decrease in SNR will cause an increase in expected energy consumption due to increased chances of packet retransmission. Likewise, for a given average SNR, a decrease in delay will generally lower the average energy cost per frame. However, this does not hold true when the delay is too small and the 802.11b overhead dominates the energy consumption. The other bit rates supported by 802.11b use a more robust modulation scheme and will operate down to a lower SNR with some slight increases in energy consumption. However, these additional modulation schemes are not represented on the graph in Figure 38.

5.2.2 Comparison of Bluetooth, IEEE 802.11b, and Client-Side ASR

In Figure 38, 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

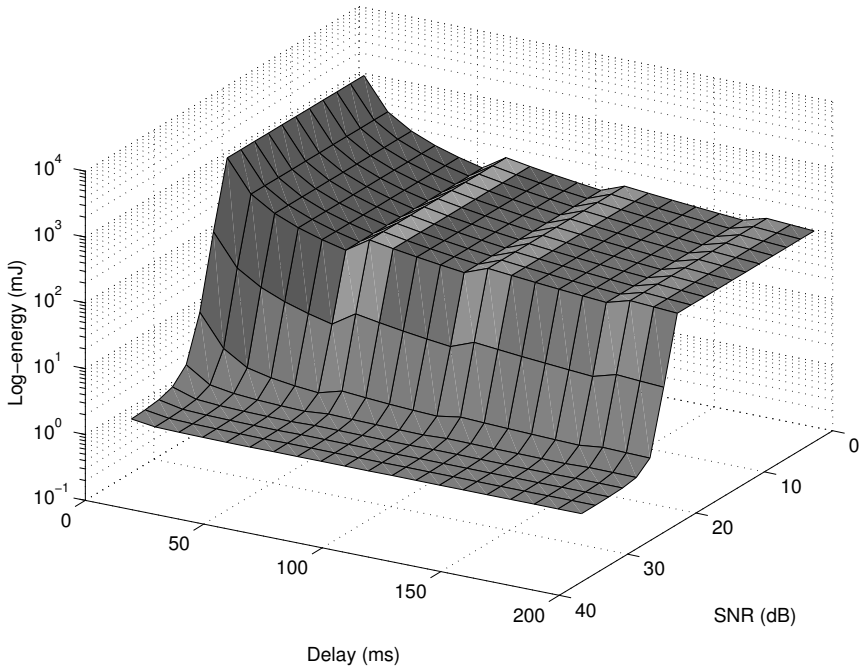


(a) Energy per frame of speech with respect to SNR and delay

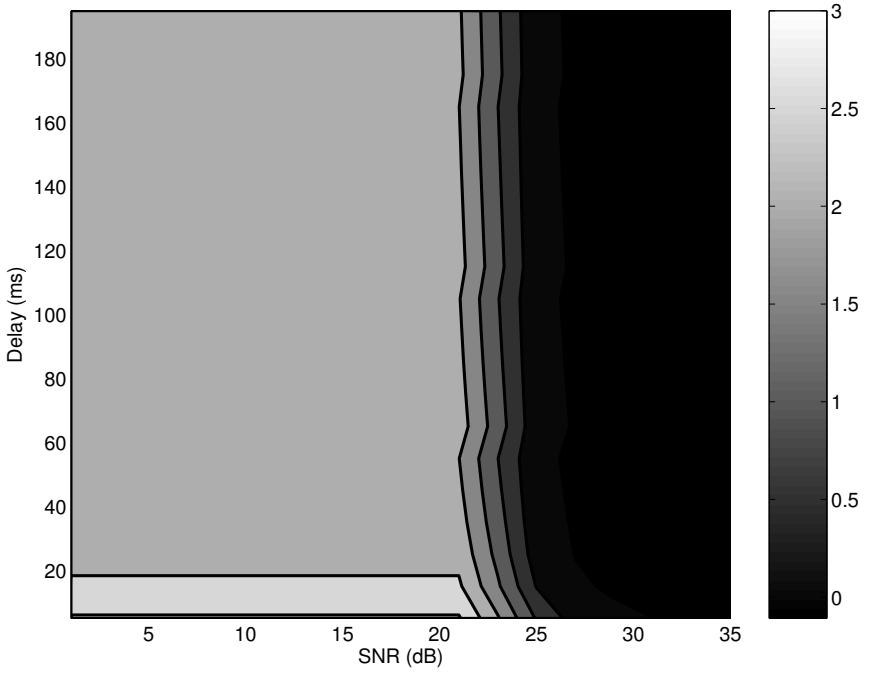


(b) SNR vs. delay contour

Figure 35: Delay, SNR, and energy tradeoffs with Bluetooth DM1 packets.

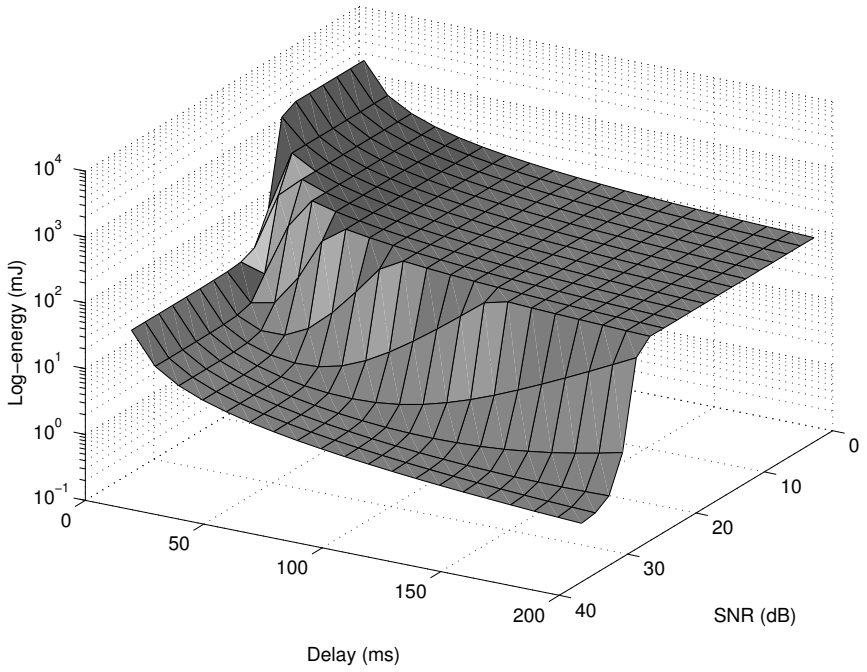


(a) Energy per frame of speech with respect to SNR and delay

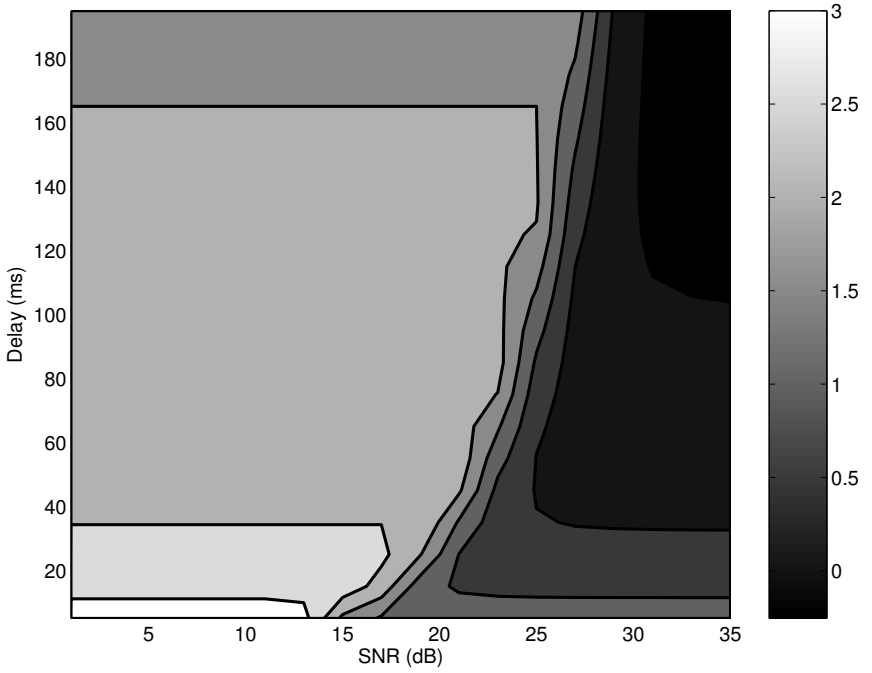


(b) SNR vs. delay contour

Figure 36: Delay, SNR, and energy tradeoffs with Bluetooth DH5 packets.



(a) Energy per frame of speech with respect to SNR and delay



(b) SNR vs. delay contour

Figure 37: Delay, SNR, and energy tradeoffs with 802.11b data packets.

show the energy consumption for both medium- and high-rate data packets as well as the three types of voice packets with latency of 480ms. To the right of the Y-axis we have the approximate energy savings over client-side ASR operating 2.5 times slower than real-time. We can expect a scaled down speech recognition task (i.e. simpler acoustic and language models or smaller vocabulary) running at real-time to give 60% energy savings. However, this will come at a cost of reduced functionality for the user, perhaps going to a more constrained vocabulary and speaking style. We have not quantified the cost of reduced utility for the user in this work. However, for the various DSR scenarios in Figure 38 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 16 shows the percentages of computation and communication energy for a few different configurations as well as the expected battery lifetime with a 1400mAh/3.6V lithium-ion cell. The 802.11b interface with long delays gives the lowest overal energy consumption and an almost even division between energy spent in computation and communication. DSR with Bluetooth uses a higher percentage of communication energy, and this amount does not decrease significantly with increased delay due to the overhead of the park mode. Expected battery lifetimes exceed that of typical cellular telephones as we do not require real-time communication. Even modest delays of less than half a second can yield significant battery lifetime with constant streaming of DSR data.

Table 16: Summary of energy consumption for ASR and DSR with high channel SNR.

Type	Comp. (%)	Comm. (%)	Total/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

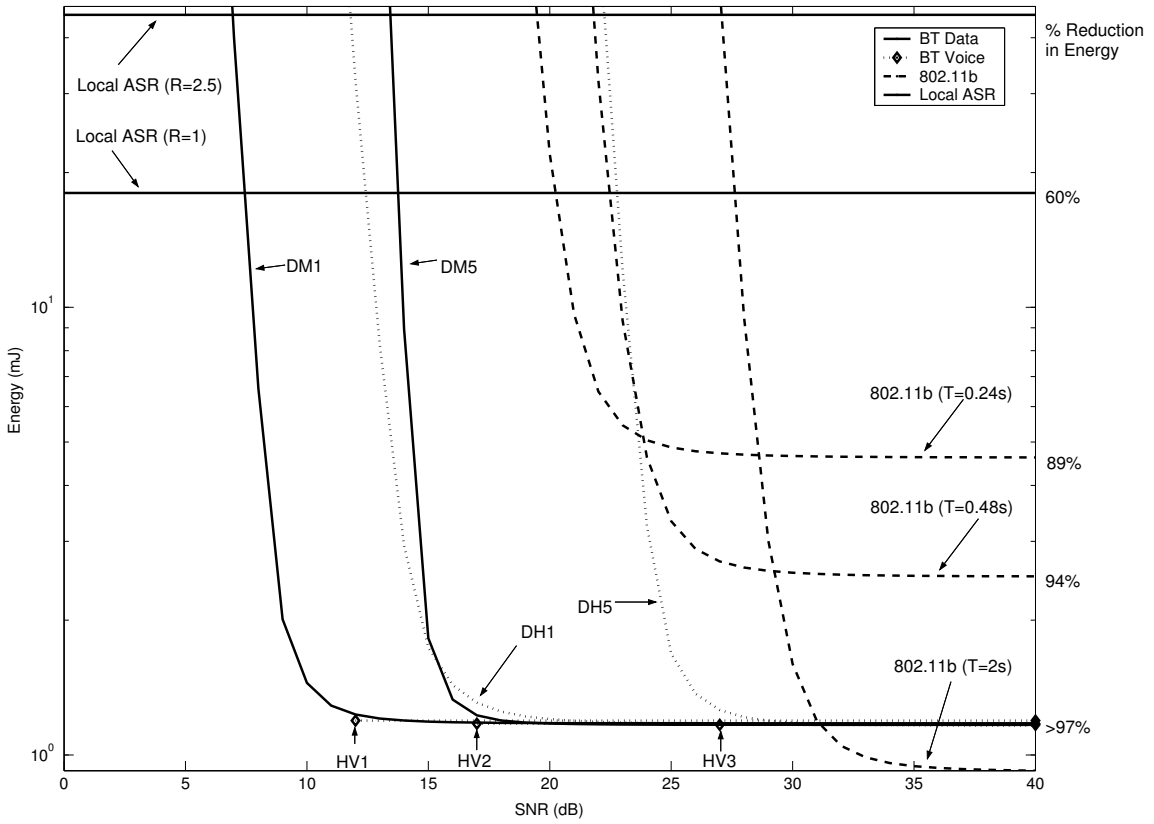


Figure 38: The energy consumption of client-side ASR and DSR under Bluetooth and 802.11b vs. SNR.

In a good channel with high SNR, Bluetooth allows systemwide energy savings of over 95% compared with full client-side ASR. DH5 packets offer the lowest overhead and best energy savings, while DM1 packets offer the most robust operation down to around 10 dB with some minimal energy cost. The ARQ retransmission protocol causes rapid increases in energy consumption after some SNR threshold is reached. It is possible to operate in lower SNR through packet fragmentation, which will lower the probability of a packet being received in error. This is evident in Figure 38 by comparing DH1 and DH5 data packets. The longer packet length in DH5 packets causes a sharp increase in retransmits and energy consumption at around 25 dB, whereas DH1 packets can operate down 15 dB before the number of retransmits becomes excessive. In addition, FEC bits can be used to lower the probability of a packet retransmit. The Hamming code in DM1 and DM5 packets allows operation

down to around 10 and 16 dB respectively.

IEEE 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 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.

5.3 Conclusion

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

In our analysis of DSR, we have considered both 802.11b and Bluetooth wireless networks. Given the relatively high bit rates these standards provide with respect to DSR traffic, we investigated the use of synchronous bursty transmission of the data to maximize the amount of time spent in a low-power or off state. While this adds a small delay to the end-user, the energy savings can be significant. With 802.11b, we can reduce the energy consumption of the wireless interface by around 80% with modest application delays of just under half a second. Bluetooth offers lower energy consumption for smaller values of delay, T, but as delay increases, the Bluetooth

energy consumption is dominated by the time spent in park mode. The 802.11b interface with on/off scheduling can operate with a lower energy consumption than Bluetooth when T exceeds 1.3 seconds.

CHAPTER VI

IMPROVED SPEECH RECOGNITION ACCURACY IN BURST ERROR CHANNELS

The presence of bit errors in the quantized speech feature stream can cause a significant decrease in accuracy. It is essential that bit errors be detected and concealed when possible. In [5], the relationship between packet erasure and packet errors with respect to accuracy was demonstrated. A DSR system is more robust to channel erasures than errors. Channel errors can cause significant changes in the most likely state sequence through an a HMM, while channel erasures did not have such a significant effect. This emphasizes the need for effective error detection, correction, or concealment techniques. In this chapter, we attempt to alleviate the effects of bit errors on the DSR bitstream through the use of interleaving, concealment through interpolation, and a novel use of the stochastic weighted Viterbi algorithm.

In Section 6.1, we discuss some related work. The channel model used to simulate burst-like bit errors is presented in Section 6.2. We overview the interpolation and interleaving methods in Section 6.4, including a characterization of the interpolation error and a sub-frame interleaving technique. This interpolation error is fed into a stochastic weighted Viterbi algorithm that is explained in Section 6.5. Recognition results are presented in Section 6.6, including a comparison to existing techniques. Finally, we analyze possible energy savings using Bluetooth voice packets in section 6.7. Conclusions and observations are presented in Section 6.8.

6.1 Related Work

In [79], the performance of DSR over IP networks is investigated. Packet losses, containing a single frame of speech, are simulated using random losses, a Gilbert-Elliot model, and a network bottleneck simulation. Repetition based error concealment is shown to be adequate for random isolated losses, but it breaks down under lengthy burst-like packet loss. An interleaving technique is introduced in [65] that distributes the speech features across multiple packets. This reduces the probability of the loss of a complete frame at the cost of increased delay. Interleaving, coupled with linear interpolation, minimized reductions in accuracy in moderate packet loss conditions. In [68] an interpolation technique in the log-filterbank domain is shown to be more robust to consecutive frame losses since log filterbank features exhibit a much higher temporal correlation. Errors were concealed by repetition, and the HMM output probability was weighted exponentially by the square root of the auto-correlation lag of the repeated feature in [5]. Secondary features were weighted according to a binary value (0 or 1) based on the length of the burst. The result was that longer bursts of repeated features counted less toward the Viterbi search update. A stochastic version of the weighted Viterbi algorithm was presented in [120] in the context of speaker verification in noise. It is based on the expected value of the HMM output probability for noisy speech vectors. The ETSI DSR standard provides a framework for MFCC calculation, quantization, framing and error protection for DSR applications [109]. We use the split vector quantization codebooks supplied with the ETSI DSR system for our simulations.

6.2 Gilbert-Elliot Model

The Gilbert-Elliot model is a two state Markov chain where the channel is either in a good or bad state [33]. Figure 39 shows this two state model. The transition probabilities, p and q, can be used to calculate the steady state probabilities of being

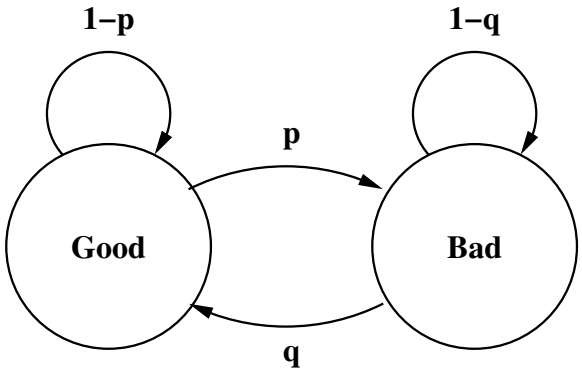


Figure 39: A two state Gilbert-Elliot channel model.

in the good or bad state as well as the mean length of a burst and mean time between bursts. This model can be used to simulate burst packet losses on packet-based networks or, in our case, burst-like bit errors over general wireless communication channels. In a typical packet level simulation, the packets are always received in the good state and never received in the bad state. In a typical bit level simulation, each state has an associated probability of error, with the probability of error in the good state being small or zero and the probability of a bit error in the bad state being larger (i.e. $E_g << E_b$). By using the state transition probabilities and state error probabilities, we can calculate the average bit error rate of the channel.

In our implementation, the codebook indices from the ETSI DSR system are packed into 6 bytes, including 4-bit CRC, for each frame. A 4-byte header is added to each 24 frame packet of speech data, for an overall bit rate of 5.0 kbps. After CRC error detection, our algorithm sees bit errors as missing frames in the MFCC feature stream.

6.3 Evaulation of the ETSI DSR Standard Under Burst-Like Error Conditions

First, we evaluate the performance of the ETSI Aurora DSR standard under various burst-like error channel conditions. The ETSI standard uses CRC error detection on consecutive frame pairs to determine if there is a bit error [109]. A consistency check

is performed by comparing the difference of the two frames against a threshold, where the threshold is determined emperically by studying clean speech. This exploits the correlation between successive frames of speech, and if the algorithm finds that they are highly uncorrelated, it assumes a bit error has occured. The consistency check helps prevent the case when an error pattern is missed by the 4-bit CRC. Errors in the quantized speech vectors are concealed simply by repeating previous or subsequent speech vectors to fill in the gap.

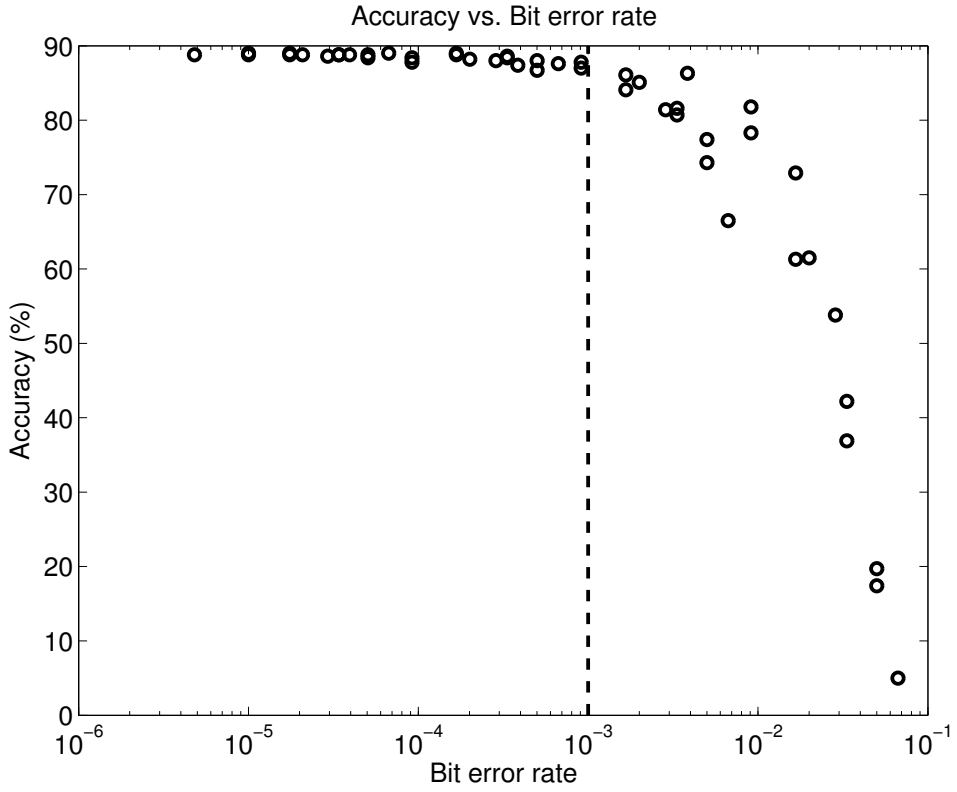


Figure 40: Average bit error rate vs. speech recognition accuracy using the ETSI DSR standard and a 5,000 word speech recognition task.

In networks that support multimedia traffic, such as Bluetooth, packets can be delivered even if they contain errors. Concealment techniques can then be used to minimize the effect of the errors at the speech recognizer (or human listener). We simulated bursty bit errors using the Gilbert-Elliot channel model and used the error

mitigation technique supplied with the ETSI standard described above. We considered a range of burst lengths, time between bursts, and error probabilities in the bad state. For each channel model, we simulated the errors on a test data set and performed error concealment on the corrupted bitstream. For each channel model considered, we calculate the average bit error probability according to the following equation:

$$BER = \frac{q}{p+q}E_g + \frac{p}{q+p}E_b \tag{41}$$

In Figure 40, we plot the average BER vs. accuracy on a 5,000-word vocabulary Wall Street Journal speech recognition task. The accuracy of the system without bit errors was 87.2%, and we used a set of 30 test utterances for each data point. We can see that the system starts to lose accuracy significantly after an average bit error probability of 10^{-3}.

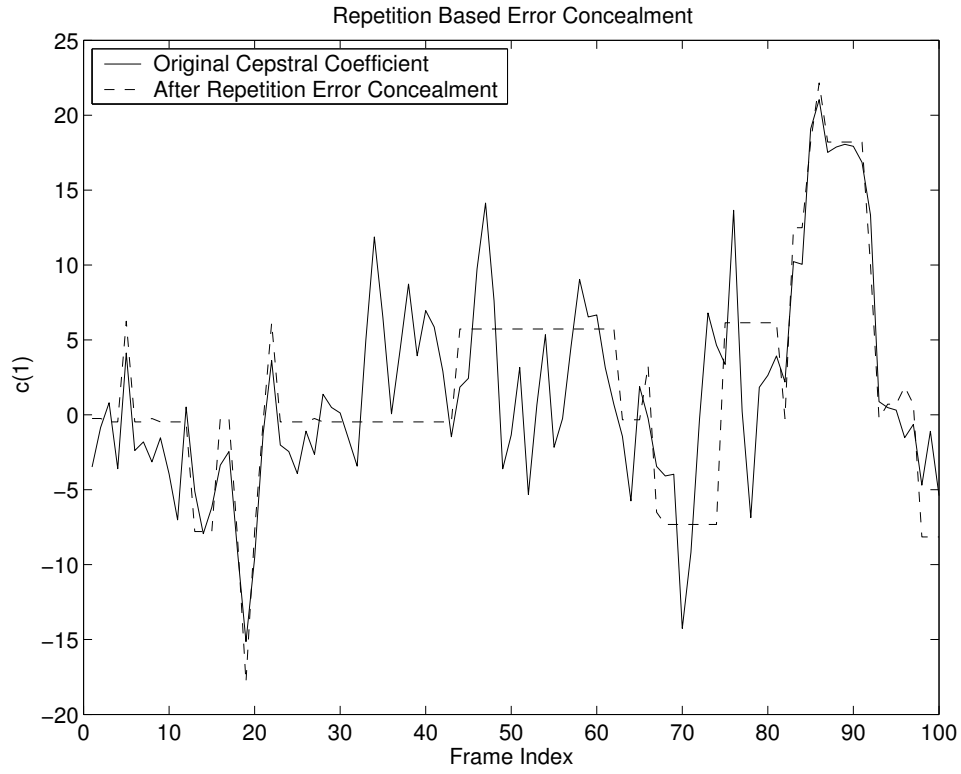


Figure 41: Repetition based error concealment in the ETSI DSR system.

One potential problem with the ETSI DSR system is the use of CRC error detection bits on consecutive frame pairs. It was shown in [102] that this use of error protection coupled with repetition can cause a single feature vector to be used for many consecutive frames in the presence of bit errors. Since the delta and delta-delta features are computed after error concealment, this could potentially result in many zeros for these secondary features, depending on the window size used to calculate them. For the remainder of this work, we will consider CRC protection bits applied only to individual frames. This adds 2 bits to each frame of speech but helps to increase robustness significantly.

6.4 Interleaving and Interpolation

Interpolation is used to conceal errors in the output feature vectors by filling the gap with data based on the good frames at either side of the gap. A frame error is defined as a failure of the 4-bit CRC in the received 48 bit frame. All codebook indices are considered corrupted and concealment is required. In [68], it was shown that interpolation in the quefrency domain is not optimal as there is little temporal correlation in the cepstrum. This is particularly true for the cepstral parameters with higher quefrency indices as they represent more quickly varying features. Instead, it was proposed that interpolation take place in the log-filterbank domain, where there is higher correlation in time among filterbank amplitudes. A block diagram of this technique is shown in Figure 42. This exploits the fact that the log filberbank amplitudes have much higher correlation in time. Figure 43 shows the cepstral coefficients $c(1)$ through $c(12)$ vs. time. The jagged, rough, surface in the plot indicates that interpolation would be difficult to perform accurately. However, in the log-filterbank domain (Figure 44), we see a much smoother surface. This technique requires some zero-padding, followed by an inverse DCT, interpolation, and finally a DCT to return to the cepstral domain. We performed cubic spline interpolation in the log-filterbank

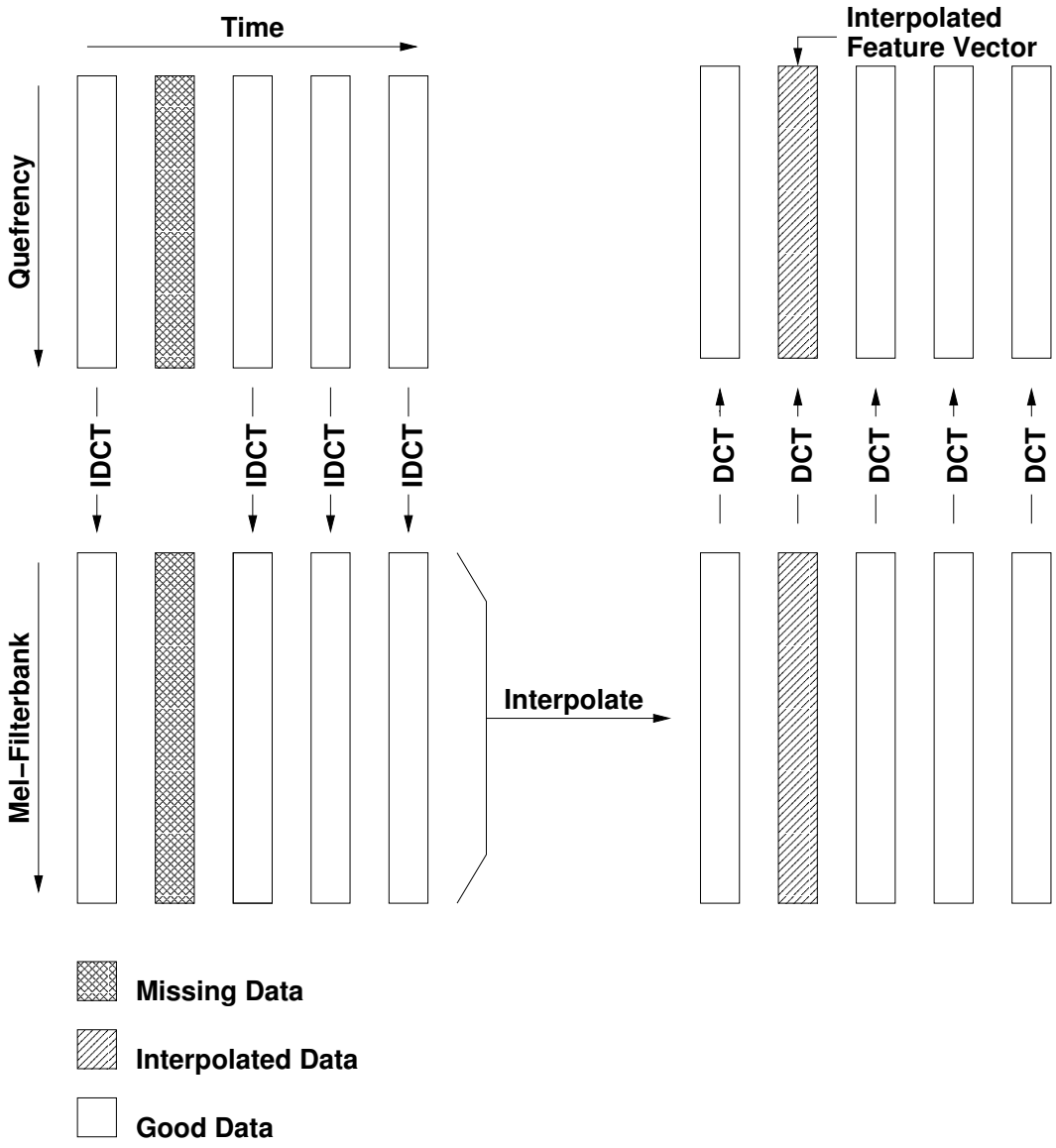


Figure 42: Interpolation of speech features in the log-filterbank domain.

domain, which offers a better performance than linear interpolation. Linear interpolation fails to take into account the curvature of the log-filterbank amplitudes vs. time. The result is that linear interpolation is not smooth in the first derivative and not continuous in the second derivative. Cubic spline interpolation can result in a much better representation of the missing data by requiring that the first and second derivatives be smooth and continuous, respectively [77].

In the presence of burst-like errors, many consecutive frames of speech can be

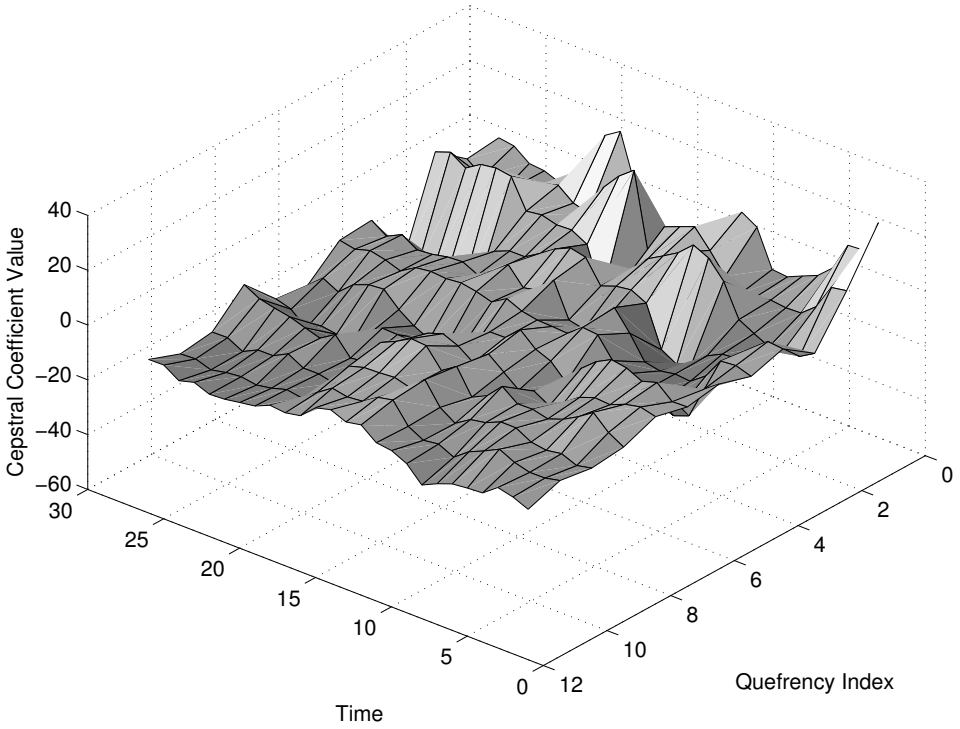


Figure 43: The cepstrum of a speech segment vs. time.

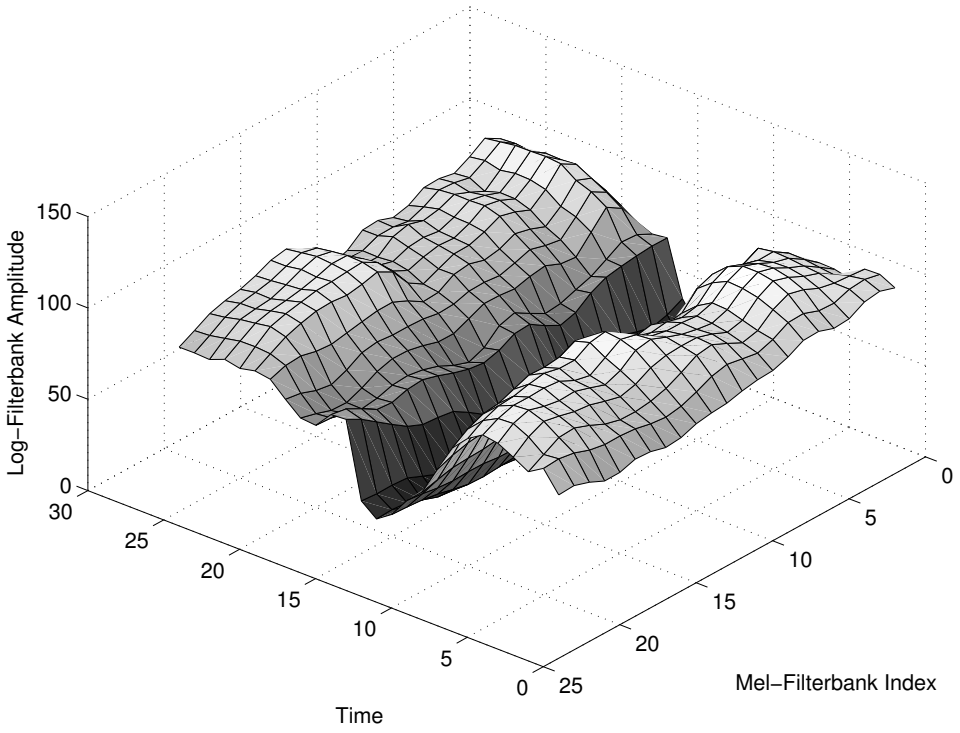


Figure 44: The log mel-filterbank amplitude of a speech segment vs. time.

corrupted. Interpolation techniques break down over longer burst lengths and can produce significant errors in the output. Interleaving is a common technique to reduce the chances of consecutive missing frames of data. By scrambling the order of the dataframes, a burst-like error pattern will spread the loss across non-consecutive packets, allowing interpolation to be performed across a smaller gap. An $M \times N$ block interleaver re-arranges the order of data frames for $M \cdot N$ total frames. The de-interleaving operation is the inverse of the interleaver, and the input frames are restored to their original order.

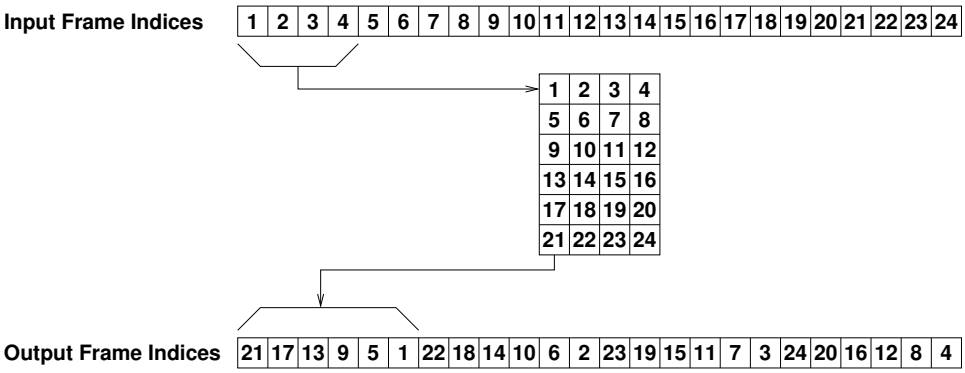


Figure 45: A 6×4 block interleaver. Input frames are numbered sequentially.

For most of the simulations in this work, we use a 6×4 interleaver, which preserves the 24 frame block size in the ETSI standard. This interleaver is shown in Figure 45. The input sequence consists of sequentially numbered frames. The output sequence is obtained by reading up each column. This interleaver separates sequential output frames by 4 (i.e. an error burst covering two frames will be separated by 4 frames in the de-interleaved sequence.) An error burst covering more than 6 output frames will result in a loss of two sequential frames in the de-interleaved output.

6.4.1 Sub-frame Interleaving

We can further separate error bursts in the output feature stream by using sub-frame interleaving. In sub-frame interleaving, we exploit the use of split vector quantization

in most DSR systems, including the ETSI system. In split-VQ, the cepstral coefficients are quantized using smaller dimensional codebooks within a frame. Typically, pairs of sequentially occuring cepstral coefficients are quantized using 2-dimensional codebooks of varying size. In the ETSI DSR system, the codebook arrangement is shown in Table 17. There are seven codebooks, each consisting of two consecutive cepstral coefficients. In the proposed sub-frame interleaving scheme, we consider each individual codebook index as a separate data element for input to the interleaver, as opposed to an entire frame block. This allows the input data sequence to be interleaved both across and within speech frames.

Table 17: Split vector quantization codebook arrangement for the ETSI DSR System [109], including group allocation for sub-frame interleaving.

Codebook	Size	Element 1	Element 2	Group
$Q^{0,1}$	64	$c(1)$	$c(2)$	1
$Q^{2,3}$	64	$c(3)$	$c(4)$	2
$Q^{4,5}$	64	$c(5)$	$c(6)$	3
$Q^{6,7}$	64	$c(7)$	$c(8)$	4
$Q^{8,9}$	64	$c(9)$	$c(10)$	5
$Q^{10,11}$	32	$c(11)$	$c(12)$	6
$Q^{12,13}$	256	$c(0)$	$\log(E)$	7

If we choose the interleaver dimensions properly, we can ensure that each group of seven codebook indices contains one index from each codebook. In this way, the bit allocation and CRC error protection can be applied to each group of seven codebook indices as in the frame-based method. However, unlike the frame-based interleaving technique, each CRC protected block contains codebook indices from seven different frames of speech. This minimizes the chances of an entire speech frame being corrupt by spreading the error across multiple frames.

A codebook index stream of 168 elements is fed into a 14×12 interleaver. This configuration maintains a delay of 24 frames as in the frame-based interleaver. In general, $M \neq N$ and M must be a multiple of the number of codebooks. Both

conditions ensure that each consecutive group of seven indices forms a full speech frame. The output of this interleaver is shown in Figure 46. The input frame index is denoted by the color of each square, where black is the first input frame and white is the last input frame. The X-axis denotes the output frame index, and the Y-axis is the output feature group. Each output frame contains MFCC coefficients from seven different speech frames. For example, the first column represents the first output frame after interleaving. It contains features from input frames 12, 17, 22, 15, 20, 13, and 24. A burst length of 13 CRC protected blocks is required to lose an entire frame of speech. The risk with this technique is that a single bit error is spread across many frames of speech rather than isolated within a single frame. However, the use of weighted Viterbi recognition can minimize this risk by indentifying which portions of the speech vector may contain errors and giving those less weight in the overall output computation.

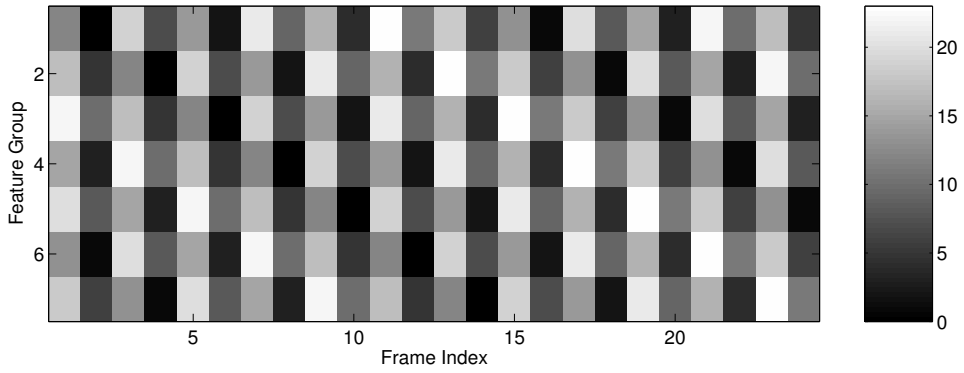


Figure 46: The output of a sub-frame interleaver with delay of 24 frames. Input frame indices are indicated by the color of each square, where black is the first input frame and white is the last input frame.

Given this sub-frame interleaver structure, we still wish to exploit the use of interpolation in the log-filterbank domain. However, given that we have the possibility of incomplete speech vectors, the interpolation will require some extra work. We can use the linearity and invertability properties of the discrete cosine transform to

accomplish this. In general,

$$DCT\left[IDCT(\mathbf{x}) + IDCT(\mathbf{y})\right] = DCT\left[IDCT(\mathbf{x}+\mathbf{y})\right] \tag{42}$$

where $\mathbf{x}$ and $\mathbf{y}$ represent arbitrary vectors and DCT and IDCT represent the forward and inverse discrete cosine transforms, respectively. We separate the feature vectors into seven separate groups according to the split-VQ scheme, and insert zeros for values not belonging to the correpsonding feature group. After zero padding the input and performing the IDCT, we have the log-filterbank representation of the respective feature group. Given the time indices of missing or bad data, we can perform interpolation on the log-filterbank values on each feature group separately. The final vectors for each feature group can be added and transformed back into the quefrency domain via a DCT followed by truncation. The entire process is outlined in Figure 47. For the purposes of the illustration, only two feature groups are shown.

6.4.2 Characterizing the Interpolation Error

In this section, we model the error due to interpolation of bad or missing feature vectors. While the interpolation is reasonably accurate for gaps of one or two frames, it breaks down with longer burst length. We consider various burst lengths in our analysis.

The interpolation error for a given feature will be modeled as zero-mean white Gaussian noise. For a given feature, n, at time, t, we have:

$$\hat{O}_{t,n} = O_{t,n} + \mathbf{v}_{t,n} \tag{43}$$

where $\hat{O}_{t,n}$ is the interpolated feature, $O_{t,n}$ is the original speech feature (unknown at the receiver), and $\mathbf{v}_{t,n}$ is the white Gaussian noise process. In order to estimate the mean and variance of this Gaussian process, we use a series of speech utterances from a test data set. Feature vectors consisting of the first 13 cepstral coefficients and the log energy are calculated for each speech utterance. Single frame errors are

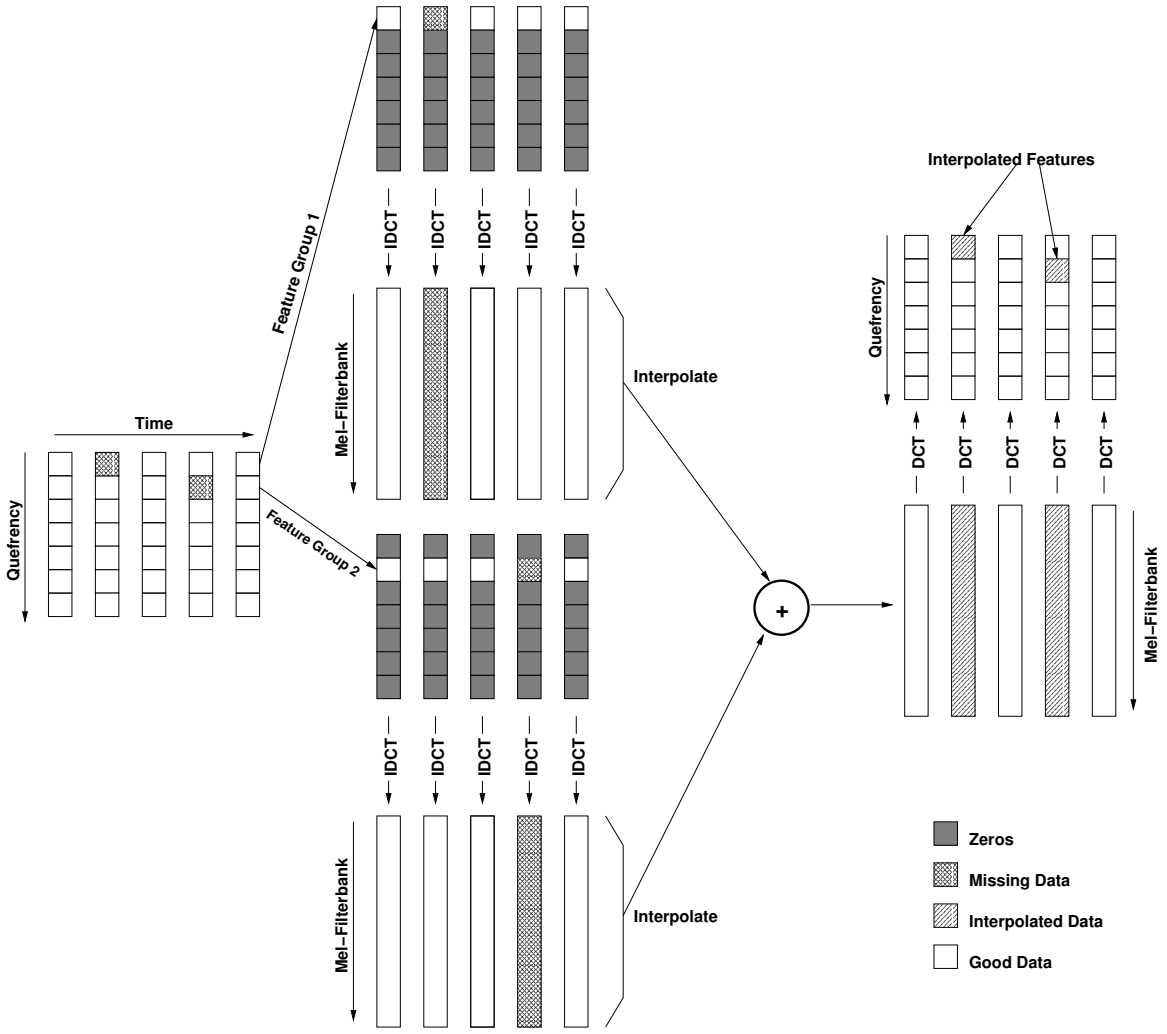


Figure 47: Interpolation in the log-filterbank domain with de-interleaved output.

periodically inserted into the speech vectors and interpolation is performed. The delta and delta-delta features are then calculated based on the interpolated and original speech vectors. Given both the interpolated, $\hat{O}_{t,n}$, and original, $O_{t,n}$, speech features, the mean and variance of the interpolation error can be estimated from the sample set, provided the test data set is large enough. The burst length is then increased by one, and the process is repeated.

The result is a set of mean and variance vectors for each feature at varying burst lengths. For bursts longer than 2 frames, the variance was significantly larger for those frames in the middle of the burst. The mean was found to be very close to zero

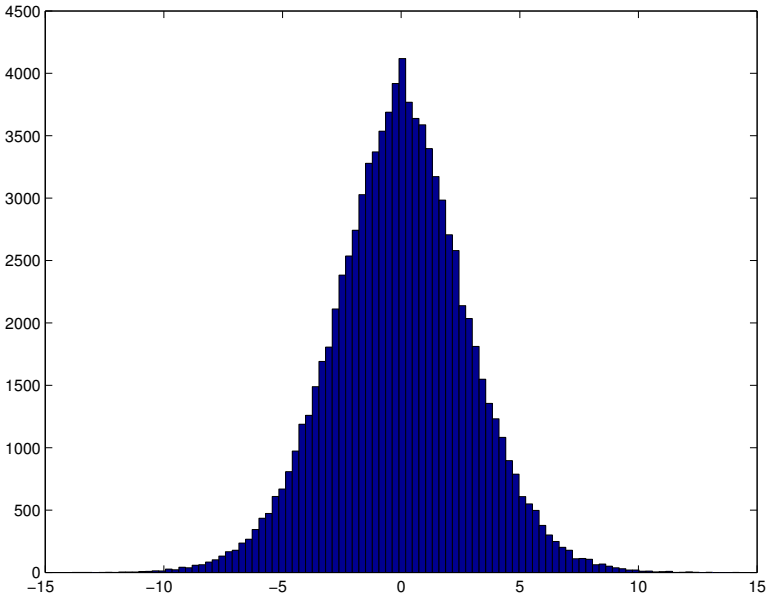


Figure 48: The distribution of the interpolation error for $c(11)$ with a burst length of one frame.

for all features and for all burst lengths tested. Figure 48 shows the distribution of the interpolation error for the 11th MFCC coefficient during a burst of length one. We will continue with the assumption that the noise process is Gaussian. However, we did find that the distribution was not always a pure Gaussian, particularly for low-time coefficients during short burst lengths. In these cases, the distribution exhibited a Gaussian shape except for a spike very close to zero. However, our algorithm still shows improvement even with the Gaussian assumption.

6.5 Weighted Viterbi Recognition Algorithm

Given a model of the additive noise for interpolated speech vectors, we can pass this uncertainty information to the Gaussian mixture density calculation. Weighted Viterbi recognition has been used to increase robustness in noise in [16], [118], and [117]. In each of these papers, the output probability was weighted exponentially according to a confidence measure of the input feature vector, so the Viterbi search update

equation looks like [16]:

$$\phi_j(t) = \max_i\{\phi_i(t-1)a_{ij}\}[b_j(\mathbf{O}_t)]^{\gamma_t} \tag{44}$$

where a_{ij} is the state transition probability, $b_j(\mathbf{O}_t)$ is the output probability of state j, $\phi_j(t)$ is the best path score into state j at time t, and γ_t is the time-varying exponential weighting between 0 and 1. This exponential weighted Viterbi algorithm was first used in DSR in [5]:

$$\phi_j(t) = \max_i\{\phi_i(t-1)a_{ij}\}\prod_{k=1}^{N}[b_j(\mathbf{O}_{k,t})]^{\gamma_{k,t}} \tag{45}$$

where the output probability of each individual feature $b_j(\mathbf{O}_{k,t})$ is weighted according to the square root of the time-autocorrelation of the feature. Therfore $\gamma_{k,t} = \sqrt{\rho_k(t-t_c)}$, where $\rho_k(t-t_c)$ is the autocorrelation of feature k at lag index t_c. The loss concealment method used in [5] was repetition, and secondary features were given zero weight when the burst length was longer than two frames. While this weighting scheme seems to work for repetition based concealment, it does not capture the statistics of log-filterbank interpolated features. Our preliminary simulations using this weighting scheme with log-filterbank interpolation showed little to no improvement, and often the system performed worse than without the weighting.

A stochastic version of the weighted Viterbi recognition (WVR) is introduced in [120] in the context of speaker indentification in noise. In [119], it is used to combat additive noise and distortion due to coding. The stochastic weighted Viterbi algorithm replaces the output probability calculation with its expected value. The value $b(\mathbf{O}_t)$ is often referred to in the literature as the output probability, when in reality it is a weighted distance measure between the input vector and the model parameters. In the normal HMM computation, the output probability (or distance) for state k is calculated as follows:

$$b(\mathbf{O}_t) = \sum_{m=0}^{M-1} C_m \frac{1}{(2\pi)^{\frac{N}{2}}|\mathbf{\Sigma}_{m,k}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\mathbf{O}_t - \mu_{m,k})^T \mathbf{\Sigma}_{m,k}^{-1}(\mathbf{O}_t - \mu_{m,k})\right] \tag{46}$$

where M is the number of Gaussian mixture densities and $\mu_{m,k}$ and $\mathbf{\Sigma}_{m,k}$ are the means and covariance matrix for state k, mixture m. Assuming diagonal covariance matrices, this probability can be expressed at the product of the probabilities of each individual feature:

$$b(\mathbf{O}_t) = \sum_{m=0}^{M-1} C_m \prod_{n=1}^{N-1} \frac{1}{\sqrt{(2\pi)\sigma_{m,k,n}^2}} \exp\left[-\frac{1}{2}\frac{(O_{t,n} - \mu_{m,k,n})^2}{\sigma_{m,k,n}^2}\right] \tag{47}$$

where $\mu_{m,k,n}$ and $\sigma_{m,k,n}^2$ are the means and variances of the mixture m, state k, and feature n. N is the total number of features. Finally, if we consider the input vector $\hat{\mathbf{O}}_t$ as a random variable, the expected value of the distance for the interpolated feature vector can be written as [120]:

$$E[b(\hat{\mathbf{O}}_t)] = \sum_{m=0}^{M-1} C_m \prod_{n=1}^{N-1} \frac{1}{\sqrt{(2\pi)\hat{\sigma}_{m,k,n}^2}} \exp\left[-\frac{1}{2}\frac{(E[\hat{\mathbf{O}}_{\mathbf{t,n}}] - \mu_{m,k,n})^2}{\hat{\sigma}_{m,k,n}^2}\right] \tag{48}$$

where $\hat{\sigma}_{m,k,n}^2$ and $E[\hat{O}_{t,n}]$ are the total variance and expected value of the corrupted feature, $\hat{O}_{t,n}$. Under the assumption of zero mean independent Gaussian noise, the variance, $\hat{\sigma}_{m,k,n}^2$, is:

$$\hat{\sigma}_{m,k,n}^2 = \sigma_{m,k,n}^2 + \tilde{\sigma}_{n,l}^2 \tag{49}$$

where $\tilde{\sigma}_{n,l}^2$ is the interpolation error variance for feature n, at burst position l that was found in Section 6.4.2. The secondary features consisting of the first and second temporal derivatives are often esimatated as a linear combination of the neighboring speech frames. Given that the errors are independent of one another, the values of $\tilde{\sigma}_{n,l}^2$ for the secondary features can be determined as follows:

$$\tilde{\sigma}_{n+13,l}^2 = \sum_{k=-L_w/2}^{L_w/2} \alpha_k^2 \cdot \tilde{\sigma}_{n,l}^2 \tag{50}$$

$$\tilde{\sigma}_{n+26,l}^2 = \sum_{k=-L_w/2}^{L_w/2} \beta_k^2 \cdot \tilde{\sigma}_{n,l}^2 \tag{51}$$

where L_w is the window length used for the derivative calculation and α_k and β_k are the coefficients used to estimate the first and second derivates of the cepstral

coefficients. Finally, the value of $E[\hat{\mathbf{O}}_{t,n}]$ is:

$$E[\hat{\mathbf{O}}_{\mathbf{t,n}}] = E[O_{t,n}] + E[\mathbf{v}_{t,n}] = O_{t,n} \tag{52}$$

since $E[\mathbf{v}_{t,n}] = 0$ and $O_{t,n}$ is a constant for time instant t.

The expression in (48) can be derived in an alternate way as follows: Given that the input vector, $\hat{\mathbf{O}}_{t,n}$, is a random variable, the distance measure can be expressed as a function of this random variable. Consider a random variable, $\mathbf{X}$, representing the distance measure for the HMM parameters in state k, mixture m. The additive noise component, $\mathbf{Y}$, represents the noise in this measurement due to interpolation. We wish to find the actual distance, $\mathbf{Z}$, with respect to the received input vector $\hat{\mathbf{O}}_{t,n}$. A random variable representing this distance after interpolation can be written as:

$$\mathbf{Z} = \mathbf{X} - \mathbf{Y} \tag{53}$$

since we wish to compensate for the increased distance from the noisy input vector by subtracting the amount from the noise estimate.

Both $\mathbf{X}$ and $\mathbf{Y}$ are characterized by normally distributed probability density functions, with $\mathbf{X}$ having mean and variance according to the HMM state and $\mathbf{Y}$ having mean and variance according to the interpolation error found in Section 6.4.2.

In general, the pdf of the difference of two normally distributed random variables with means and variances (μ_x, σ_x^2) and (μ_y, σ_y^2) can be written as [113]:

$$P_{\mathbf{X}-\mathbf{Y}}(u) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{x^2}{2\sigma_x^2}} \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{y^2}{2\sigma_y^2}} \delta((x-y)-u)dxdx \tag{54}$$

$$= \frac{1}{\sqrt{2\pi(\sigma_x^2+\sigma_y^2)}} e^{-\frac{(u-(\mu_x-\mu_y))^2}{2(\sigma_x^2+\sigma_y^2)}} \tag{55}$$

Substituting the appropriate variables for the means and variances, we have:

$$P_{\mathbf{Z}}(\hat{\mathbf{O}}_{t,n}) = P_{\mathbf{X}-\mathbf{Y}}(O_{t,n}) = \mathcal{N}(O_{t,n}, \mu_{m,k,n} - E[\mathbf{v}_{t,n}], \sigma_{m,k,n}^2 + \tilde{\sigma}_{n,l}^2) \tag{56}$$

where $\mathcal{N}(x, \mu, \sigma^2)$ is the normal distribution with mean μ and variance σ^2 for variable x. And since $E[\hat{\mathbf{O}}_{t,n}] = E[O_{t,n}] + E[\mathbf{v}_{t,n}]$, this is equivallent to:

$$P_{\mathbf{Z}}(\hat{\mathbf{O}}_{t,n}) = \frac{1}{\sqrt{(2\pi)\hat{\sigma}^2_{m,k,n}}} \exp\left[-\frac{1}{2}\frac{(E[\hat{\mathbf{O}}_{t,n}] - \mu_{m,k,n})^2}{\hat{\sigma}^2_{m,k,n}}\right] \tag{57}$$

where $\hat{\sigma}^2_{m,k,n} = \sigma^2_{m,k,n} + \tilde{\sigma}^2_{n,l}$. Substituting equation (57) back into the multidimensional Gaussian mixture model calculation, we have:

$$P_{\mathbf{Z}}(\hat{\mathbf{O}}_{t,n}) = \sum_{m=0}^{M-1} C_m \prod_{n=1}^{N-1} \frac{1}{\sqrt{(2\pi)\hat{\sigma}^2_{m,k,n}}} \exp\left[-\frac{1}{2}\frac{(E[\hat{\mathbf{O}}_{t,n}] - \mu_{m,k,n})^2}{\hat{\sigma}^2_{m,k,n}}\right] \tag{58}$$

which is the same as equation (48).

The algorithm works as follows. In the absence of any bit errors, no interpolation is performed, and the interpolation error variance, $\tilde{\sigma}^2_{n,l}$, is set to zero. In this case, the output probability is identical to (46). In the presence of bit errors, the burst length, in frames, is determined by delaying until an un-errored frame is received. Interpolation is performed, and the appropriate set of variances, $\tilde{\sigma}^2_{n,l}$, for the burst length and position within the burst for each feature are passed to the Gaussian evaluation. When the variance of the interpolation error is high, the probability densities of all states and all mixtures tend toward a uniform distribution, so the output probability is given less weight in the overall Viterbi search update and the classification ability of the model is decreased. This has the effect of minimizing both beam pruning and the use of $b(\mathbf{O}_t)$ in the Viterbi state update equations for noisy features. In the case of shorter error bursts, the lower time cepstral coefficients and their derivatives will be given more weight as they have more correlation in quefrency and can be interpolated over greater distances with less error.

There are several key differences between the use of the proposed stochastic weighted Viterbi recognition and the exponential version discussed in [5]. The first is that the amount of adaptation in the stochastic version is related to the HMM variances. Distributions with large variances in relation to the error variance will

have little changes in their output probabilities, while distributions with small variances in relation to the error will have large changes in their output probablilities due to the adaptation. In the exponential version, the weighting is the same regardless of the model variance. In the presence of long error bursts, the probabilities in the exponential version tend toward one, while in the stochastic version, they tend toward zero. This may impact the use of language modeling as the language model weight is tuned to compensate for the under-estimation of the acoustic model probablities. In the exponential version, the acoustic model scores are over-estimated in the presence of errors. In the stochastic version, the acoustic model scores are further under-estimated in the presence of errors, which allows the language model to take on a greater role during periods of errors.

6.6 Results

We tested various error concealment schemes, including the weighted Viterbi recognition, using the TIDIGITS speech corpus. Word models with 16 Gaussian mixture densities per state were trained using 941 utterances whose 39-dimensional MFCC vectors were calculated using the ETSI DSR front-end. The test set consisted of 336 digit utterances of varying length and across different speakers. In addition to the TIDIGITS test set, we also evaluated the performance on the Wall Street Journal (WSJ) database under the same channel error conditions. Triphone acoustic models with 16 Gaussian mixture densities were trained using 1,792 clean speech utterances. The test set consisted of 166 utterances using a bigram language model with a 5,000 word vocabulary.

The burst-like error conditions were simulated using the two state Gilbert-Elliot model discussed in Section 6.2. The bit error patterns for a particular speech utterance were computed in advance and stored on disk. In this way, each concealment method is tested against the same bit errors for a given set of channel model parameters. We

varied both the mean burst length in bits, $\overline{T_b}$, and the mean time between bursts, $\overline{T_g}$. We also show the average bit error rate (BER) for the burst channel conditions. The probability of bit error in the good state is fixed at 10^{-6}, and the probability of bit error in the bad state is fixed at 10^{-1}.

Table 18: A summary of DSR systems tested.

System Parameter	ETSI	INT-LFB	EXP-WVR	STO-WVR	SF-WVR
CRC Protection					
Frame Pair	√				
Single Frame		√	√	√	√
Concealment					
Repetition	√		√		
Interpolation		√		√	√
Interleaving					
Frame Based		√	√	√	
Sub-frame					√
Adaptation					
Exponential WVR			√		
Stochastic WVR				√	√

Table 18 summarizes the various DSR configurations that we tested. Each system adds an increasing amount of error detection and concealment. The system labelled ETSI is the ETSI DSR system with no modifications. The bit rate of ETSI is 4.8 kbps, while the bit rate of INT-LFB, EXP-WVR, STO-WVR, and SF-WVR is 5.0 kbps due to the single frame based CRC error detection. Each system uses the same VQ codebooks provided with the ETSI DSR standard. System INT-LFB uses a 6×4 block interleaver with interpolation in the log-filterbank domain. System EXP-WVR uses the method described in [5] with exponential WVR and repetition based concealment. (In order to provide a fair comparison, we have also added frame-based interleaving to this system.) Finally, systems STO-WVR and SF-WVR both use stochastic WVR and log-filterbank interpolation with frame-based and sub-frame interleaving, respectively.

During the preliminary simulations, the STO-WVR system did not always offer improvement in the more difficult WSJ task. This is likely due to the decrease in the discrimative ability of the acoustic models in the presence of the weighting, coupled with the larger number of parameters in the triphone acoustic models. In addition, the variance may be overestimated due to the Gaussian assumption for the interpolation noise. A scaling factor, α, was introduced to reduce the effect of the adaptation.

$$\hat{\sigma}^2_{m,k,n} = \sigma^2_{m,k,n} + \alpha \cdot \tilde{\sigma}^2_{n,l} \tag{59}$$

where $0 \leq \alpha \leq 1$. Recognition results were then obtained using a small test set, and the results are shown in Figure 49. There seems to be a peak in recognition accuracy around $\alpha = 0.3$. We use this scaling factor of 0.3 on the STO-WVR and SF-WVR systems for the WSJ task.

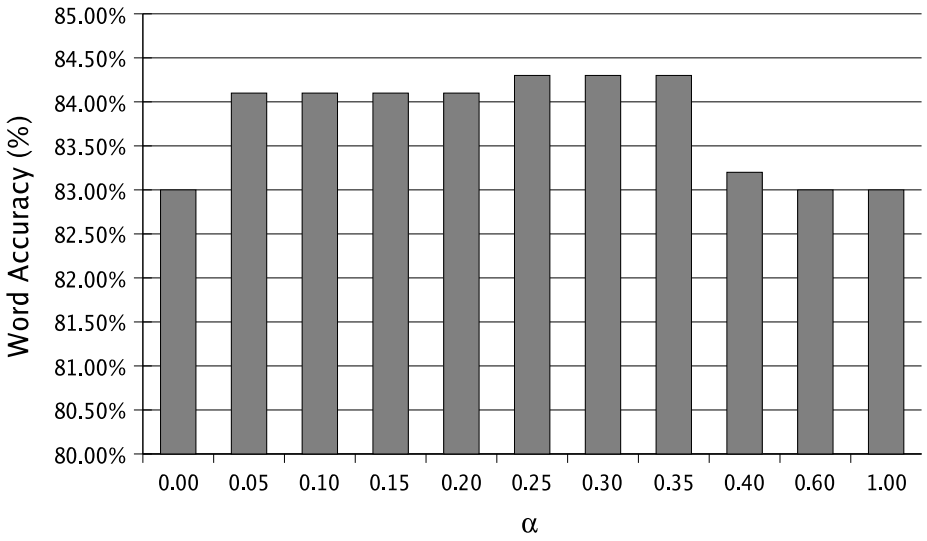


Figure 49: Selection of α for the WSJ recognition task.

The results for a variety of error conditions are shown in Table 19 and Figures 50 and 51. Word accuracy, including substitutions, deletions, and insertions, is reported. The baseline accuracy without bit errors is 99.5% for the TIDIGITS task and 85.7% for the WSJ task. From the table, we can see that the ETSI system does not provide

Table 19: Results of DSR simulations for TIDIGITS and WSJ Tasks. Reported WER reduction is relative to the INT-LFB system.

TIDIGITS Task						
$\overline{T_g}/\overline{T_b}$	Avg. BER	ETSI	INT-LFB	EXP-WVR	STO-WVR	SF-WVR
500/200	2.86×10^{-2}	91.00	98.30	98.90	99.20	98.90
200/100	3.33×10^{-2}	89.70	99.00	98.40	98.80	98.60
200/200	5.00×10^{-2}	72.50	95.10	95.90	96.80	97.60
500/500	5.00×10^{-2}	71.00	91.90	94.20	94.60	95.20
200/500	7.14×10^{-2}	45.30	80.20	83.90	86.00	86.60
Average Accuracy		73.90	92.90	94.26	95.08	95.38
WER reduction		–	–	19.15	30.70	34.93
WSJ Task						
$\overline{T_g}/\overline{T_b}$	Avg. BER	ETSI	INT-LFB	EXP-WVR	STO-WVR	SF-WVR
500/200	2.86×10^{-2}	50.50	81.90	81.20	82.70	82.90
200/100	3.33×10^{-2}	41.10	80.90	78.90	82.20	82.90
200/200	5.00×10^{-2}	16.80	69.40	68.00	70.30	73.00
500/500	5.00×10^{-2}	18.20	64.30	66.50	68.10	68.00
200/500	7.14×10^{-2}	5.10	38.30	41.50	41.30	45.20
Average Accuracy		26.34	66.96	67.22	68.92	70.40
WER reduction		–	–	0.79	5.93	10.41

adequate performance in the more severe error conditions that we have simulated. Accuracy quickly drops below 90% in the digits task and *starts* at around 50% in the WSJ task. The average accuracy for each system under all error conditions is also listed in Table 19.

The simulataneous introduction of interleaving, log-filterbank interpolation, and single frame based CRC error detection in system INT-LFB offer a large decrease in WER over the ETSI system. The relative reductions in WER for the TIDIGITS and WSJ task are 72.8% and 55.2%, respectively. The last line of each table shows the decrease in average WER relative to the INT-LFB system. The EXP-WVR system from [5] exhibits a 19.15% relative reduction in average WER for the TIDIGITS task but only a 0.79% relative reduction in WER for the WSJ task. The proposed stochastic WVR systems are able to perform significantly better in both the TIDIGITS and

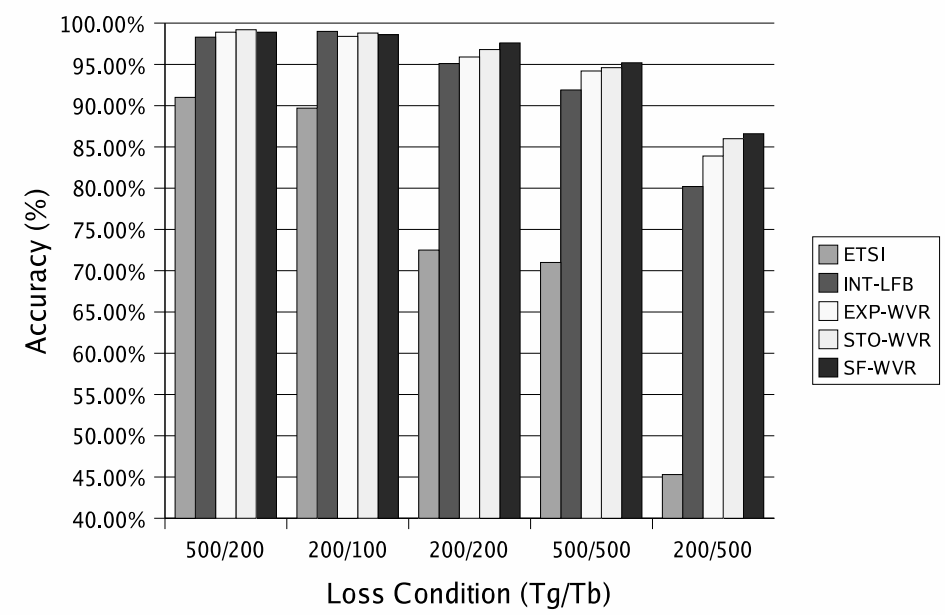


Figure 50: Results of DSR simulations for TIDIGITS task.

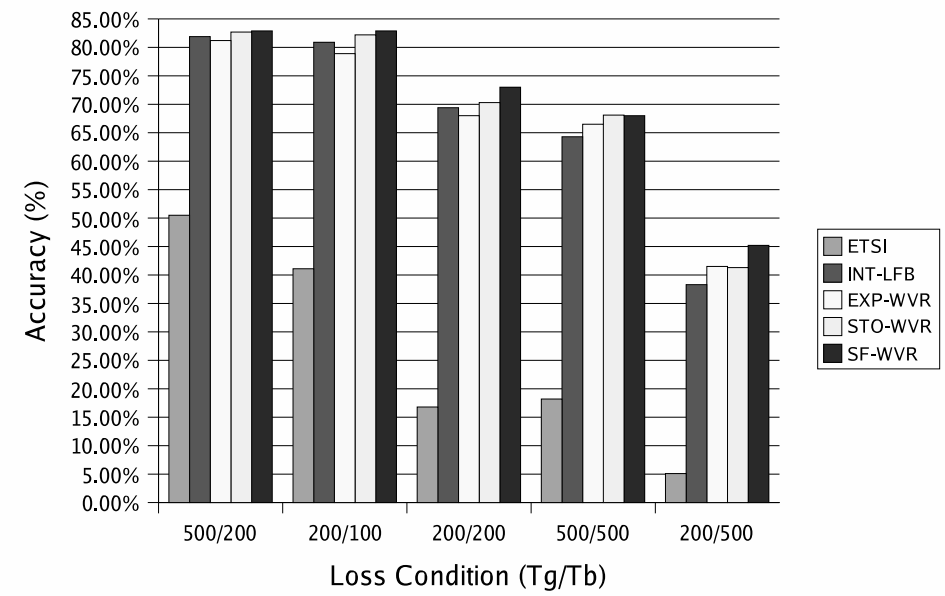


Figure 51: Results of DSR simulations for WSJ task.

WSJ tasks. Using frame-based interleaving, the STO-WVR system offers a 30.7% relative reduction in WER over INT-LFB on the TIDIGITS task and a 5.93% reduction on the WSJ task. Only a small amount of further improvement, 34.93%, is seen with the introduction of sub-frame interleaving (SF-WVR) on the TIDIGITS task.

However, on the WSJ task, the use of sub-frame interleaving and stochastic WVR can reduce the average WER by 10.41%. The SF-WVR system is able to provide accuracy up to 97.6% and 73.0% in burst-like error conditions with an average BER as high as 5.00×10^{-2} for the TIDIGITS and WSJ task. This is 1 in every 20 bits in error.

6.7 Reduced Energy Consumption with Bluetooth Voice Packets

Bluetooth voice packets have energy consumption that is independent of SNR since no ARQ protocol is used. We assume that the synchronization and header portions of the transmission occur without error. The short header is protected by a 1/3 rate repetition code, and the payload is sent using the error correction code appropriate to the packet type. HV3 packets contain no error protection and have the lowest overhead and, therefore, the lowest energy consumption per frame of speech. The payload of HV2 packets is protected by a (15,10) Hamming code, and HV1 packets are protected by a 1/3 rate repetition code. The packet payload is delivered even if it contains bit errors.

By using increased delay, as with the data packets, we can minimize the energy consumption by increasing the amount of time spent in the low-power park state. However, since ARQ is not used, bit errors can have an impact on speech recognition accuracy. In section 6.3, we investigated the performance of the ETSI DSR standard in the presence of bit errors and introduced sub-frame interleaving and the use of stochastic weighted Viterbi recognition to minimize the effect of bursty bit errors on ASR accuracy. Using the combined interleaving and weighted Viterbi recognition approach, we can show how increased interleaver delay can improve the ASR accuracy. In Figure 52, we plot the accuracy vs. interleaver delay for two severe burst error conditions with the same bit error rate, 5×10^{-2}, on a subset (166 utterances) of the

WSJ task. The 200/200 error condition represents a faster fading situation where the state transition probabilities, p and q, between the good and bad states are larger, and the mean time spent in both the good and bad states last for 200 bits. The fast fading error condition has better performance with moderate interleaver delay. In general, the interleaver delay increases quadratically with the separation between frames at the interleaver output. The result is that the improvements from interleaving begin to level off after 64 frames (0.64 seconds) of delay for both error conditions.

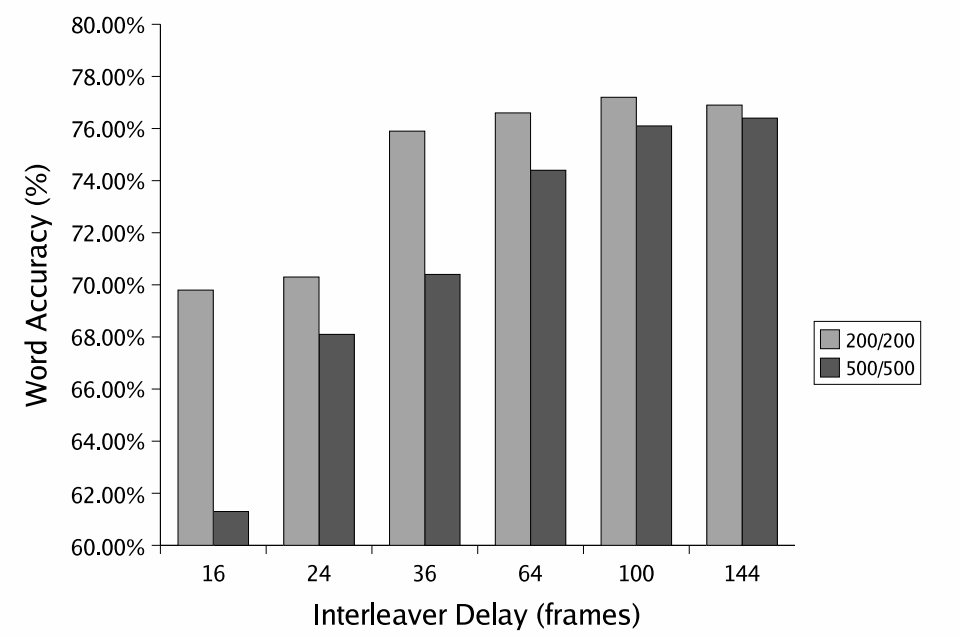


Figure 52: Accuracy vs. interleaver delay for the SF-WVR system on the WSJ task with bit error probability of 5×10^{-2}.

In Figure 53, the energy consumption per frame of speech is plotted vs. the interleaver delay. This energy consumption includes the time between transmissions, including the low-power park state. For a given packet type the reduction in energy consumption with respect to increased delay levels off after 64 frames. This coincides with the accuracy results, suggesting that delays beyond 0.64 seconds have little benefit in terms of improved accuracy and decreased energy consumption. This is due in part to the energy overhead of the park state (see Figure 34), where the amount of

time spent in the park state makes it a dominant portion of the energy consumption. However, some modest energy savings are possible, coupled with the possibility of improved accuracy in burst error conditions. In addition, several vendors now provide an additional *deep sleep* mode with power consumption around 270 μWatts. There is a longer delay (typically 250ms) to switch into this mode, but it can help reduce the energy consumption for longer delays.

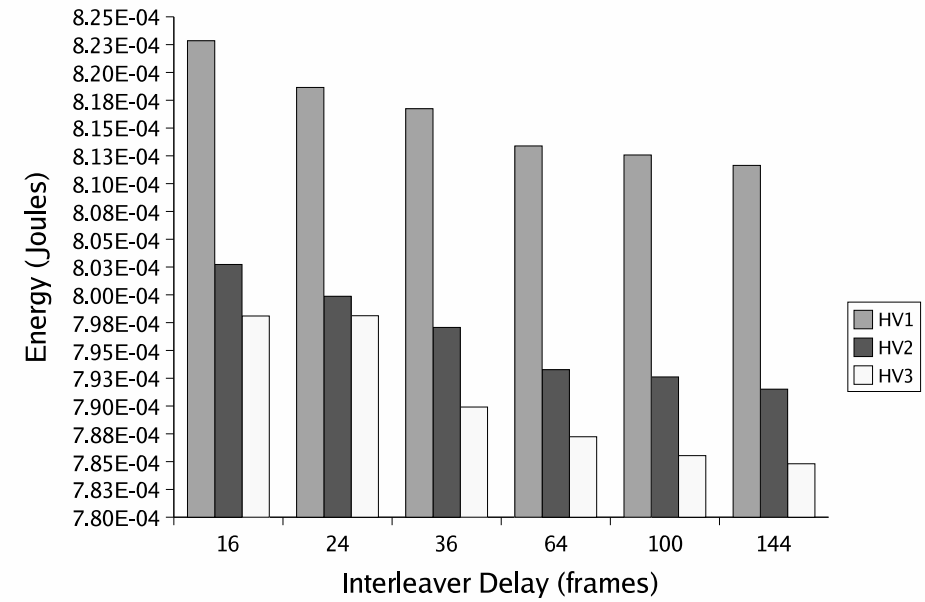


Figure 53: Energy vs. interleaver delay for Bluetooth voice packet types.

When discussing the use of data packets, we assumed a minimal quality of service (ASR accuracy) for the user, maintained through the use of error protection and/or retransmission of corrupt packets. While this offered a maximum of ASR accuracy, the plots in Figure 38 show how the energy consumption can increase dramatically for a given packet type/size when SNR drops below some threshold. A hard decision must be made either between packet types/sizes or client-side vs. distributed ASR. The use of Bluetooth voice packets (or multimedia packets in general) allow for constant energy consumption with variable ASR accuracy, subject to channel conditions

and error protection. In practice, a threshold on the bit error rate or SNR can be determined where the accuracy will drop below usable levels. In Chapter 6 we showed how the performance of the ETSI DSR standard degrades rapidly when the error rate exceeds 10^{-3} (see Figure 40). Through the use of sub-frame interleaving and stochastic weighted Viterbi recognition (SF-WVR), we showed how the accuracy can be improved in burst error channels with bit rates exceeding 10^{-2} (see Table 19 and Figures 50 and 51). Given this data, we can infer that adequate quality of service can be obtained with an upper bound on the BER, after error correction, of about 10^{-2}. Figure 54 shows the error correction performance for various types of Bluetooth voice packets. The upper bounds on bit error probability for both the ETSI DSR system and the SF-WVR are plotted as horizontal lines, 10^{-3} and 10^{-2} respectively.

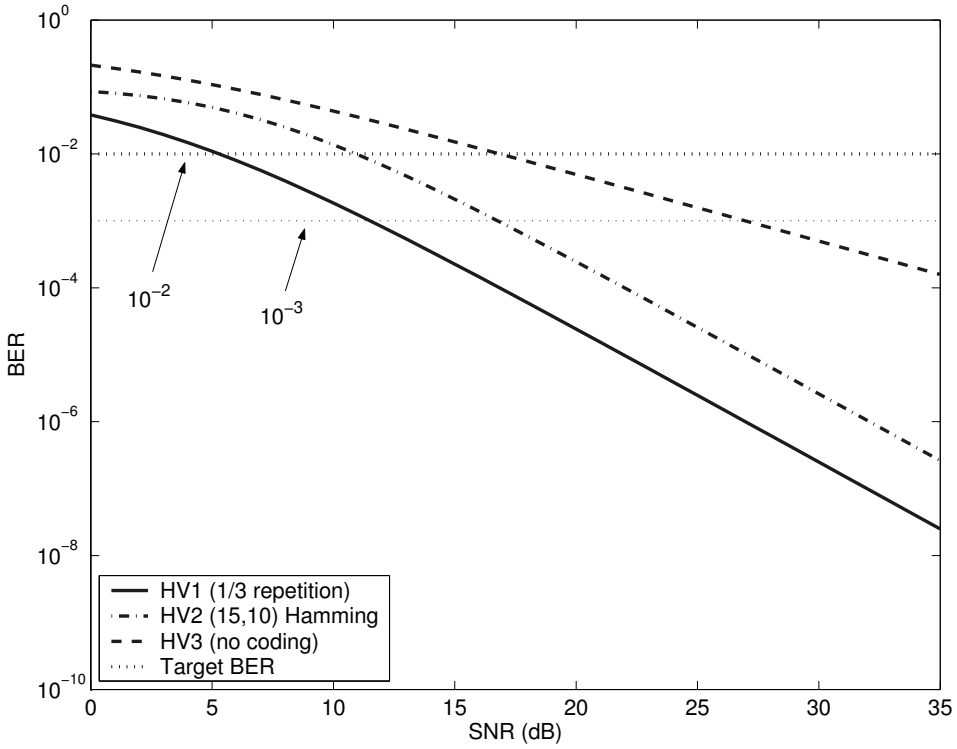


Figure 54: The error correction performance of Bluetooth voice packet types. The x-axis is signal to noise ratio per bit, and the y-axis is the bit error probability after error correction.

The approximate lower bounds for SNR based on these target bit error rates are

Table 20: Lower SNR bound for ETSI and SF-WVR and energy consumption with Bluetooth voice packets.

	SNR Lower Bound (dB)		
Packet Type	ETSI	SF-WVR	Energy (μJ)
HV3	27	17	23.4
HV2	17	10.5	37.5
HV1	12	5	70.3

given in Table 20 for various Bluetooth voice packet types. When using the ETSI standard, HV3 packets (which contain no error correction) will only operate down to an SNR of 27 dB. By increasing the allowed upper bound on BER through the SF-WVR system, a gain of 10 dB is possible. Gains of about 6.5 to 7 dB of SNR are possible for HV2 and HV3 packets, respectively. This gain can be translated to an energy savings either through a decreased reliance on error correction or through a reduction in transmit power. A 6 dB gain in performance would allow the node to transmit at 25% of it's original power output and still maintain adequate performance. However, the Bluetooth standard does not currently support variable transmit power. The energy required to transmit one frame of speech with Bluetooth voice packet types (not including park mode overhead) is listed in the final column of Table 20. By using SF-WVR, the following energy savings are possible: Between 27 and 17 dB a 37% reduction in transmit energy is possible since HV3 packets can be used instead of HV2 packets. A 46% reduction in transmit energy between 17 and 10.5 dB since HV2 packets can be used instead of HV1 packets. DSR can still be used down to 5 dB SNR, so the much more expensive client-side ASR does not need to be used. As shown in Figure 38, the ability to use DSR with adequate performance can provide reductions in system wide energy consumption by over 90%.

6.8 Conclusion

In this chapter, we have investigated the effects of several well-known error concealment algorithms for DSR traffic over a burst error channel. We present a novel use of the stochastic weighted Viterbi algorithm to further conceal the effects of interpolated features as well as a new sub-frame interleaving technique. The systems were compared under several simulated burst error conditions. The combined stochastic WVR and sub-frame interleaving techniques can provide accuracy as high as 97.6% in burst error conditions with an average BER of $\frac{1}{20}$ on a digit recognition task. Under the same error condition, a WSJ task maintained an accuracy of 73%. The improvement in accuracy is shown to be better than existing techniques, particularly in a large vocabulary task with an N-gram language model. We believe further improvement may be possible through a more accurate modeling of the interpolation noise, perhaps through a mixture of Gaussians or a Laplacian distribution.

In the presence of bit errors, we can estimate the energy consumption with respect to SNR and identify the appropriate operating ranges for the various packet types. For Bluetooth voice packets, we recommend a minimum bit error rate after error correction of 10^{-3} with the ETSI standard and 10^{-2} after using interleaving and loss protection techniques introduced in this chapter. Using these techniques, Bluetooth voice packets can operate down to 5 dB SNR with small only reductions reduction in accuracy. In addition, the decreased reliance on error correction code overhead can yield energy savings in packet transmission of up to 46%.

CHAPTER VII

DISCUSSION AND CONCLUSION

This thesis presents an in-depth study of a pervasive distributed speech recognition multimedia client. Optimization at many layers is considered from software to hardware and networking layers. Quality of service metrics such as delay and accuracy are incorporated into our analysis where applicable and minimization of energy consumption is studied.

In Chapter 3, a voice user interface architecture was implemented based on an open standard markup language for speech applications. Standard off-the-shelf speech technology was used, and the system was integrated with a software radio testbed. In addition, a working prototype on a Compaq iPAQ PDA with IEEE 802.11b network access was demonstrated.

In Chapter 4, a software-based, low-power, fixed-point front-end is developed for the HP Labs Smartbadge IV. Hand optimization of software for embedded devices can greatly reduce the energy consumption. Through the use of architectural and algorithmic enhancements, the energy consumption of the algorithm is greatly reduced. The use of approximate algorithms such as fast complex magnitude calculation and a logarithm through simple bit manipulation provide sufficient accuracy for the MFCC front-end while reducing the energy consumption. In section 4.2 a simple model for computation of a small vocabulary connected word speech recognizer is developed. In section 4.3 a literature survey of the performance of large vocabulary speech recognition systems on modern computer hardware is presented. LVCSR requires a large amount of memory bandwidth to operate in real-time. Large L1 and L2 cache sizes can help to minimize this bandwidth to main memory, but most current workstation

class computers do not contain such a large cache. In addition, the Gaussian evaluation step requires floating point arithmetic to compensate for the large dynamic range. A model of energy consumption for client-side ASR is presented in section 4.3.4. This would likely be a system running slower than real-time with reduced accuracy over a server based system.

Chapter 5 examines the energy consumption of wireless transmission of speech recognition data across local area data networks, including IEEE 802.11b and Bluetooth. The bit-rates required for DSR traffic are generally an order of magnitude or more less than the maximum throughput of the wireless network. The communication is therefore characterized by short bursts of transmission followed by idle periods until the next amount of speech data is ready for transmission. Significant savings can be obtained by increasing the delay to exploit energy saving modes of the wireless interface. We report battery lifetimes in excess of 40 hours using our distributed speech recognition enhancements, while local ASR can only provide just over 1 hour of operation.

In Chapter 6, an algorithm to improve the accuracy of distributed speech recognition in error prone channels is developed. Bit errors are simulated using a Gilbert-Elliot channel model, and frame errors are then seen as a failure of the CRC protection scheme. Interleaving and interpolation in the log-filterbank domain is used to decrease the effect of missing frames on the back-end ASR search. A novel use of stochastic weighted Viterbi recognition is presented to minimize the effect of interpolation on missing speech frames. By identifying those frames and features that may have large error during interpolation, we can improve the accuracy of the speech recognition system. In addition, the use of sub-frame interleaving can provide further improvement in recognition accuracy by providing separation of errors without additional delay. The weighted Viterbi algorithm can provide relative reductions of 34% on a digit recognition task and 10% on a more complex large vocabulary task when compared

to only interleaving and interpolation. This technique is able to outperform previously proposed techniques, especially in the presence of N-gram language modeling. This technique, including sub-frame interleaving and weighted Viterbi recognition, allows further reductions in energy consumption by allowing Bluetooth voice packets to operate in a lower SNR range. In particular, the usable lower bound with HV2 packets is decreased by almost 7 dB, which allows a savings in transmit energy of 46% over the more expensive HV1 packets.

Wireless networking has enabled the application of many different kinds of multimedia technology on mobile devices through distributed computing. Applications such as streaming audio and video, real-time voice and video communication, and media recognition are now possible on mobile devices. Even with distributed computing the battery lifetimes of these small devices are limited. Optimization is important at all levels of the application. At the software level, energy efficient source code can be written that outperforms available compiler optimizations. Scalability can be addressed through flexible algorithms that allow various quality of service points (i.e. varying quantization levels, frame rates, video quality, etc.) By modifying the computational ability of the hardware through techniques such as dynamic frequency and voltage scaling and scalable memory heirarchies, the energy consumption can be further reduced. The wireless optimization problem can also be addressed at many layers. The application can tailor quality of service, which may map to a particular bit-rate, to vary the application performance and possibly reduce activity on the wireless interface. Efficient protocols can be used to minimize retransmissions and packet errors. Scalable error correction can be used to add redunancy only when the channel conditions require it. At the physical layer, the transmit power can be adapted to the noise and interference conditions. Simply powering off extraneous peripherals and I/O devices when not needed can provide large reductions in energy consumption. Each of these optimizations must in turn coordinate with the user or

operating system to provide a set of *knobs* or tuning parameters such that the desired quality of service can be acheived while simultaneously increasing battery lifetime. As particular applications have differing quality requirements, the energy optimization techniques suitable for the application may differ as well.

7.1 Future Research

This thesis considers a distributed speech recognition application on a PDA-like device with either 802.11b or Bluetooth wireless networking. This narrowed scope enabled a detailed analysis across a smaller subset of available hardware and standards. Future research directions may include a broader look at hardware, software, networking options, and applications. The following topics are suggested:

- A more detailed analysis of the energy consumption profile of large vocabulary continuous speech recognition search algorithms on future embedded processors. This may include incorporation of energy models from specialized ASIC media processing components ranging from analog signal processing to Gaussian evaluation coprocessors.

- Application of energy aware distributed speech recognition to other wireless networks, such as CDMA 2000 or UMTS.

- Analysis of other multimedia applications such as real-time video communication or streaming video or audio.

- A more detailed modeling of the log-filterbank interpolation error (i.e. Gaussian mixtures) for the weighted Viterbi algorithm.

- Combined acoustic and channel noise robustness using weighted Viterbi recognition.

REFERENCES

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

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

[3] Agaram, K., Keckler, S., and Burger, D., "A characterization of speech recognition on modern computer systems," in *IEEE International Workshop on Workload Characteriztion*, 2001.

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

[5] Bernard, A. and Alwan, A., "Low-bitrate distributed speech recognition for packet-based wireless communication," *IEEE Transactions on Speech and Audio Processing*, vol. 10, pp. 570–579, November 2002.

[6] Borgatti, M., Felici, M., Ferrari, A., and Guerrieri, R., "A low–power integrated circuit for remote speech recogntion," *IEEE Journal of Solid State Circuits*, pp. 1082–1089, July 1998.

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

[8] Brodd, R. J., "Into the crystal ball - dimly: Battery technology developments," in *WESCON '94: Idea Microelectronics*, 1994.

[9] Chang, C.-T., Chang, C.-T., Yang, H.-L., and Chang, H.-T., "Real–time implementation of speech recognition using risc processor core," in *Proceedings of the 9th Annual IEEE International ASIC Conference and Exhibit*, pp. 231–234, 1996.

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

[11] Chowdhury, D. and Chia, S., "Distributed processing in the home using a pc with a wireless speech interface," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, pp. 2363–2366, 1999.

[12] Comerford, R., "Handhelds duke it out for the internet," *IEEE Spectrum*, pp. 35–41, August 2000.

[13] Consortium, W. W. W., "Voice browser activity." Work in Progress. `http://www.w3.org/Voice`.

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

[15] C.T. Hemphill, P.R. Thrift, J. L., "Speech–aware multimedia," *IEEE Multimedia*, vol. 3, pp. 74–78, 1996.

[16] Cui, X., Bernard, A., and Alwan, A., "A noise robust asr back-end technique based on weighted viterbi recognition," in *EuroSpeech 2003*, 2003.

[17] Danielsen, P. J., "The promise of a voice–enabled web," *Computer*, vol. 33, pp. 104–106, August 2000.

[18] Delaney, B., Jayant, N., Hans, M., Simunic, T., and Acquaviva, A., "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)*, vol. 1, pp. 793–796, 2002.

[19] Delaney, B., Jayant, N., Hans, M., Simunic, T., and Acquaviva, A., "A low-power fixed-point front-end feature extraction for a distributed speech recognition system," Tech. Rep. HPL-2001-252, Hewlett-Packard Laboratories, 2002.

[20] Delaney, B., Jayant, N., and Simunic, T., "A wlan scheduling algorithm to reduce the energy consumption of a distributed speech recognition system," in *Proceedings of the First Workshop on Embedded Systems for Real-Time Multimedia (ESTIMedia 2003)*, 2003.

[21] Delaney, B., Simunic, T., and Jayant, N., "Energy aware distributed speech recognition for mobile wireless devices," *IEEE Design and Test of Computers: Special Issue on Embedded Systems for Real-Time Multimedia*, September 2004.

[22] Delaney, B., Simunic, T., and Jayant, N., "Energy aware distributed speech recognition for wireless mobile devices," tech. rep., Hewlett-Packard Laboratories, 2004.

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

[24] Deshmikh, N., Ganapathiraju, A., and Picone, J., "Hierarchical search for large vocabulary conversational speech recognition," *IEEE Signal Processing Magazine*, pp. 84–105, September 1999.

[25] Digilakis, V., Neumeyer, L., and Perakakis, M., "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.

[26] Ditlea, S., "The pc goes ready to wear," *IEEE Spectrum*, pp. 35–39, October 2000.

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

[28] Fainberg, M., "A performance analysis of the ieee 802.11b local area network in the presence of bluetooth personal area network," Master's thesis, Polytechnic University, 2001.

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

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

[31] Ganapathiraju, A., Goel, V., Corrada, J. P. A., Doddington, G., Kirchoff, K., Ordowski, M., and Wheatley, B., "Syllable – a promising recognition unit for lvcsr," in *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing*, vol. 1, pp. 421–424, 1997.

[32] Geppert, L. and Perry, T. S., "Transmeta's magic show," *IEEE Spectrum*, pp. 26–33, May 2000.

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

[34] Green, K. and Wilson, J. C., "Future power sources for mobile communications," *Electronics and Communication Engineering Journal*, 2001.

[35] Greene, S. R. and Fiske, C. F., "Wearable computers open a new era in support resource management," *AUTOTESTCON: IEEE Systems Readiness Technology Conference*, pp. 99–101, 1999.

[36] Gunawan, W. and Hasegawa-Johnson, M., "Plp coefficients can be quantized at 400 bps," in *ICASSP 2001*, 2001.

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

[38] Hans, M., "Smartbadge/badgepad version 4." `http://www.hpl.hp.com/personal/Mat_Hans/research/badge4/index.htm`, January 2004. HP Labs.

[39] Hong, I., Kirovski, D., Qu, G., Potkonjak, M., and Srivastava, M., "Power optimization of variable voltage-core based systems," in *Design Automation Conference*, 1998.

[40] Hong, I., Potkonjak, M., and Srivastava, M., "On-line scheduling of hard real-time tasks on variable voltage processor," in *International Conference on Computer-Aided Design*, 1998.

[41] Huang, X., Acero, A., and Hon, H.-W., *Spoken Language Processing*. Upper Saddle River, New Jersey: Prentice Hall, 2001.

[42] Ishihara, T. and Yasuura, H., "Voltage scheduling problem for dynamically variable voltage processors," in *IEEE International Symposium on Low Power Electronics and Design*, 1998.

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

[44] Juang, B. H. and Rabiner, L. R., "Hidden markov models for speech recognition," *Technometrics*, vol. 33, pp. 251–272, August 1991.

[45] Kannan, A., "Robust estimation of stochastic segment models for word recogntion," Master's thesis, Boston University, 1992.

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

[47] Karray, L. and Mauuary, L., "Improving speech detection robustness for wireless speech recognition," in *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pp. 428–435, 1997.

[48] Kidd, C. D., O'Connell, T., Nagel, K., Patil, S., and Abowd, G. D., "Building a better intercom: Context-mediated communication within the home." Submitted for review to CHI 2001, 2000.

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

[50] Kim, H. K. and Cox, R. V., "Feature enhancement for a bitstream-based front-end in wireless speech recognition," in *ICASSP 2001*, 2001.

[51] Kim, H. K., Cox, R. V., and Rose, R. C., "Performance improvement of a bitstream-based front-end for wireless speech recognition in adverse environments," *IEEE Transactions on Speech and Audio Processing*, 2002.

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

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

[54] Lai, C., Lu, S.-L., and Zhao, Q., "Performance analysis of speech recognition software," in *Eighth International Symposium on High Performance Computer Architecture*, 2002.

[55] Lee, C.-H., Carpenter, B., Chou, W., Chu-Carroll, J., Reichli, W., Saad, A., and Zhou, Q., "On natural language call routing," *Speech Communications*, vol. 31, pp. 309–320, 2000.

[56] Lee, S. and Sakurai, T., "Run-time voltage hopping for low-power real-time systems," in *IEEE International Symposium on Low Power Electronics and Design*, 2000.

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

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

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

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

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

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

[63] Martin, J. D., Rodriguez, J. G., Zapata, J., and Vilda, P., "Robust voice recognition as a distributed service," in *ETFA 2001: 8th International Conference on Emerging Technologies and Factory Automation*, vol. 2, pp. 571–575, October 2001.

[64] Mathew, B., Davis, A., and Fang, Z., "A low-power accelerator for the sphinx 3 speech recognition system," in *CASES*, 2003.

[65] Mayorga, P., Lamy, R., and Besacier, L., "Recovering of packet loss for distributed speech recognition," in *EUSIPCO 2002*, 2002.

[66] McKenzie-Mills, K. and Alty, J. L., "Investigating the role of reduncancy in multimodal input systems," 1998.

[67] Microsystems, S., "The source for java technology." `http://java.sun.com`, 2000.

[68] Milner, B., "Robust speech recognition in burst-like packet loss," in *ICASSP 2001*, 2001.

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

[70] Min, R., *Energy and Quality Scalable Wireless Communicaiton*. PhD thesis, M.I.T., June 2003.

[71] Muthusamy, Y., Agarwal, R., Gong, Y., and Viswanathan, V., "Speech–enabled information retrieval in the automobile environment," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, pp. 2256–2262, 1999.

[72] Olive, J. P., "The voice user interface," *Global Telecom Conference*, vol. 4, pp. 2051–2055, 1999.

[73] Pering, T., Burd, T., and Brodersen, R., "Voltage scheduling in the iparm microprocessor system," in *IEEE International Symposium on Low Power Electronics and Design*, 2000.

[74] Picone, J., "Signal modelling techniques in speech recogntion," *Proceedings of the IEEE*, vol. 81, pp. 1215–1247, September 1993.

[75] Poritz, A. B., "Hidden markov models: A guided tour," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7–12, 1988.

[76] Potamianos, A. and Weerackody, V., "Soft-feature decoding for speech recognition over wireless channels," in *ICASSP 2001*, 2001.

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

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

[79] Quercia, D., Docio-Fernandez, L., Garcio-Mateo, C., Farinetti, L., and Martin, J. D., "Performance analysis of distributed speech recognition over ip networks on the aurora database," in *ICASSP 2002*, 2002.

[80] Rabiner, L. and Juang, B. H., *Fundamentals of Speech Recognition*. Englewood Cliffs, New Jersey: Prentice Hall, 1993.

[81] Rabiner, L. R., "Applications of voice processing to telecommunications," *Proceedings of the IEEE*, vol. 82, pp. 199–228, Feb 1994.

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

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

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

[85] Ravishankar, M. K., *Efficient Algorithms for Speech Recognition*. PhD thesis, Carnegie Mellon University, 1996.

[86] Rudnicky, A. I., Reed, S. D., and Thayer, E. H., "Speechwear: A mobile speech system," in *Proceedings of the 4th International Conference on Spoken Language*, vol. 1, pp. 538–541, 1996.

[87] Scholey, N., "Rechargeable batteries for mobile communications," in *IEE Colloquium on Radio Frequency Design in Mobile Radio Transceivers*, 1994.

[88] Shan, M., Yanlei, Z., and Youwei, Z., "Corba based distributed computing model for multimodal speech recognition," in *The International Symposium on Intelligent Multimedia, Video and Speech Processing, 2001*, pp. 417–420, 2001.

[89] Sharman, R., "Markets and prospects for speech and language," *IEEE Colloquium on State of the Art Speech and Language Engineering*, pp. 12/1–12/3, November 1998.

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

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

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

[93] Shin, Y. and Choi, K., "Power conscious fixed priority scheduling for hard real-time systems," in *Design Automation Conference*, 1999.

[94] Simunic, T., Benini, L., Acquaviva, A., Glynn, P., and Micheli, G. D., "Dynamic voltage scaling and power management for portable systems," in *DAC*, 2001.

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

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

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

[98] Smailagic, A., "Isaac: A voice activated speech response system for wearable computers," *First International Symposium on Wearable Computers*, pp. 183–184, 1997.

[99] Smith, M. T. and Jr., G. Q. M., "Smartbadge/badgepad version 4." `http://www.it.kth.se/~maguire/badge4.html`, January 2004. HP Labs and Royal Institute of Technology (KTH).

[100] Srinivasamurthy, N., Ortega, A., Zhu, Q., and Alwan, A., "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.

[101] Takahashi, E., "Application aware scheduling for power management on ieee 802.11," in *IPCCC*, 2000.

[102] Tan, Z.-H. and Dalsgaard, P., "Channel error protection scheme for distributed speech recognition," in *ICLSP '02*, 2002.

[103] Tanaka, K., "Next major application systems and key techniques in speech recognition technology," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, pp. 1057–1060, 1998.

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

[105] Various, "The salt form." `http://www.saltforum.org`.

[106] Various, "Sphinx-2 open-source speech recognizer (version 0.3)." `http://www.speech.cs.cmu.edu/speech/`.

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

[108] Various, "Open services gateway initiative." `http://www.osgi.org`, 2000.

[109] Various, "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.

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

[111] The VoiceXML Forum, *The Voice eXtensible Markup Language*, March 2000. http://www.voicexml.org.

[112] Weerackody, V., Reichl, W., and Potamianos, A., "An error-protected speech recognition system for wireless communications," *IEEE Transactions on Wireless Communications*, vol. 1, pp. 282–291, 2002.

[113] Weisstein, E. W., "Normal difference distribution." MathWorld - A Wolfram Web Resource: http://mathworld.wolfram.com/NormalDifferenceDistribution.html.

[114] White, J., "Voice browsing," *IEEE Internet Computing*, vol. 4, pp. 55–56, Febuary 2000.

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

[116] Yao, F., Demers, A., and Shenker, S., "A scheduling model for reduced cpu energy," in *IEEE Annual foundations of computer science*, 1995.

[117] Yoma, N., McInnes, F., and Jack, M., "Improving performance of spectral subtraction in speech recognition using a model for additive noise," *IEEE Transactions on Speech and Audio Processing*, vol. 6, pp. 579–582, November 1998.

[118] Yoma, N., McInnes, F., and Jack, M., "Weighted viterbi algorithm and state duration modeling for speech recognition in noise," in *ICASSP 1998*, pp. 709–712, 1998.

[119] Yoma, N., Silva, J., Busso, C., and Brito, I., "Compensating additive noise and cs-celp distortion in speech recognition using stochastic weighted viterbi recognition," *Electronics Letters*, vol. 39, pp. 409–411, February 2003.

[120] Yoma, N. and Villar, M., "Speaker verification in noise using a stochastic version of the weighted viterbi algorithm," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 3, pp. 158–166, 2002.

[121] Zanella, A., Miorandi, D., and Pupolin, S., "Mathematical analysis of bluetooth energy efficiency," in *Proceedings of WPMC03*, October 2003.

[122] Zhang, W., He, L., Chow, Y.-L., Yang, R., and Su, Y., "The study on distributed speech recognition system," in *ICASSP 2000*, pp. 1431–1434, 2000.

[123] Zhu, Q. and Alwan, A., "An efficient and scalable 2d dct-based feature coding scheme for remote speech recognition," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2001.