# A VOICE OVER IP FRAMEWORK AND SIMULATION FOR LOW RATE SPEECH AND THE FUTURE NARROWBAND DIGITAL TERMINAL

By

EDWARD J. G. DANIEL

Bachelor of Science in Electrical Engineering

Oklahoma State University

Stillwater, Oklahoma

1997

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2000

A VOICE OVER IP FRAMEWORK AND SIMULATION

FOR LOW RATE SPEECH AND THE FUTURE

NARROWBAND DIGITAL TERMINAL

Thesis Approved:

Thesis Advisor

Dean of the Graduate College

## ACKNOWLEDGEMENTS

This thesis would not have been possible without the support and guidance of many people. I would like to take this opportunity to acknowledge and thank some of those people.

First, I would like to thank my Lord and Savior, Jesus Christ, who is the author and finisher of my faith. Thank you for blessing the work of my hands, and giving me all that I needed to do this and every good work. For it was your commitment to me that enabled me finish this project. I pray that this thesis will be a testament to your faithfulness, love, and glory.

Second, I want to thank my advisor, Dr. Keith A. Teague, AKA "Zorro", for his technical expertise, wisdom and support that was instrumental in the completion of this thesis. Thank you for allowing me to put work, including this thesis, on hold, so that I could enjoy the first precious moments of my new baby girl's life. I am forever grateful for your friendship and support. Additionally, I wish to thank the other members of my advisory committee, Dr. Rao Yarlagadda, and Dr. Rama Ramakumar, as well as the School of Electrical and Computer Engineering for their support throughout the years. I also want to thank the Department of Defense for funding this research especially Rich Dean and Lynn Supplee.

Next, I would especially like to thank my wife, Cherdena Daniel. You have truly been by inspiration. Thank you for your patience, support, sacrifice, and encouragement

throughout the years, especially on this project. I could not have completed this thesis without you by my side. Thank you.

Finally, I want to thank my mother Shirley Wallace and my four sisters, Kim, Lynn, Darlene, and Donna for the love, support, prayers, and encouragement they have given me throughout the years. I also would like to thank my brother-in-law Randy for his encouragement and particularly for introducing me to the field of engineering. My other family members and friends has also prayed for and encouraged me, and I appreciate them. None of this would have ever been possible without their support.

## TABLE OF CONTENTS

Chapter Page

## LIST OF TABLES

## LIST OF FIGURES

## NOMENCLATURE

| | |
|---|---|
| A/D | Analog-to-Digital Converter |
| ACK | Acknowledgement |
| ACM | Audio Compression Manager |
| ADPCM | Adaptive Differential Pulse Code Modulation |
| ATM | Asynchronous Transfer Mode |
| C | C programming language |
| CELP | Code Excited Linear Prediction |
| D/A | Digital-to-Analog Converter |
| DDVPC | Defense Digital Voice Processor Consortium |
| EMBE | Enhanced Multi-Band Excitation |
| EOM | End of Message |
| FDDI | Fiber Distributed Data Interface |
| FEC | Forward Error Correction |
| FIFO | First In First Out |
| FNBDT | Future NarrowBand Digital Terminal |
| $F_s$ | Sampling Frequency |
| I/O | Input / Output |
| IMBE | Improved Multi-Band Excitation |
| IP | Internet Protocol |

| | |
|---|---|
| ISDN | Integrated Services Digital Network |
| ISO | International Organization for Standardization |
| ITU | International Telecommunications Union |
| LAN | Local Area Network |
| LMR | Land Mobile Radio |
| LPC | Linear Predictive Coder |
| LSF | Line Spectral Frequencies |
| Mb | Mega bit |
| MBE | Multi-Band Excitation |
| MELP | Mixed-Excitation Linear Prediction |
| MoIP | MELP-over-IP |
| ms | millisecond |
| MS Windows NT | Microsoft Windows NT |
| NLMS | Normalized Least-Mean-Square |
| OS | Operating System |
| OSI | Open System Interconnect |
| OSU | Oklahoma State University (header) |
| PBX | Private Branch Exchange |
| PC | Personal Computer |
| PCM | Pulse Code Modulation |
| POTS | Plain Old Telephone System |
| PPP | Point-to-Point Protocol |
| PSTN | Public Switched Telephone Network |

| | |
|---|---|
| QoS | Quality of Service |
| RTP | Real-Time Protocol |
| SNR | Signal to Noise Ratio |
| SOM | Start of Message |
| STE | Secure Terminal Equipment |
| TCP | Transmission Control Protocol |
| TE | Terminal Equipment |
| TI | Texas Instruments |
| TransPK | Transmit Packet |
| UDP | User Datagram Protocol |
| VoIP | Voice-over-IP |
| WAN | Wide Area Network |

# CHAPTER I

# INTRODUCTION

## Purpose

This study describes the design and implementation of a voice over packet-switched network, Voice over Internet Protocol (VoIP), system using the new proposed federal standard Mixed-Excitation Linear Prediction (MELP) 2400 bits per second voice coder developed by Texas Instruments. The implementation simulates one application of the Future NarrowBand Digital Terminal (FNBDT). FNBDT is being developed by the government to provide secure digital multimedia communication, including voice, data, image, video and others, between users over a concatenation of possibly dissimilar communication networks (Internet, cellular, wireless, satellite, public switched telephone network, etc.).

MELP is an important component of the government's future low rate voice communication systems, of which FNBDT represents one aspect. A primary application of FNBDT will be to provide secure communications for narrowband mobile users in tactical military environments [40], and thus represents one way that MELP will be deployed. FNBDT will also provide secure communication for both narrowband mobile users and broadband wireline (land) users, with the eventual goal of providing an interoperable system that will enable secure communication for mobile and wireline users over mixed networks.

The work described in this thesis deals with the design and development of a MELP over IP implementation that can be used to study the behavior of FNBDT with MELP and a PC-to-PC based network connection. Of interest are Internet, digital cellular

(wireless) and combinations of such networks. This design and implementation examines the various components that comprise the end-to-end VoIP system and illustrates Quality of Service (QoS) issues associated with transmission of low rate voice over the Internet using the FNBDT operational modes. This system facilitates the investigation of various design trade-offs, including packet length and number of voice frames per packet, loss recovery schemes, and other system constraints that are characteristic of voice over IP systems. This implementation simulates several FNBDT operational modes including blank and burst, burst without blank, and clear MELP. It specifically does not simulate FNBDT signaling and control, or data security (encryption).

**Thesis Outline**

The remainder of this thesis details the development, design and implementation of a VoIP system using the new proposed federal standard MELP 2400bps voice coder. A breakdown of the rest of this thesis is presented in the following paragraphs.

Chapter 2 provides the reader with a brief introduction of voice over IP and proceeds with an identification of the 2 major components that make up a VoIP system, namely the audio codec and the network. A more thorough discussion of the audio codec (MELP), IP network characteristics, as well as background information of FNBDT, is also presented.

Chapter 3 examines the Quality of Service (QoS) challenges associated with a VoIP system. The discussion focuses on various QoS trade-offs as well as the identification of problems and potential solutions to QoS issues.

Chapter 4 discusses the overall VoIP system design while addressing the various QoS issues discussed in Chapter 3. It also provides an end-to-end system delay budget based on design decisions discussed in this chapter.

Chapter 5 presents the implementation of the VoIP system using MELP as the audio codec and the Internet as the IP network. It focuses on each component that makes up the end-to-end implementation as well as the interaction between components. It incorporates the design decisions of Chapter 4 into the implementation.

Chapter 6 summarizes the system design and the considerations involved in the design. Additional applications for this system are discussed. Finally, suggestions for future research are outlined.

# CHAPTER II

# BACKGROUND

Prior to 1994 the use of the Internet for voice communication was unheard of. With the Internet being used strictly as a "best effort" data service, there was little confidence in reliably transmitting real-time voice data. But, in mid 1994, a few companies introduced services offering Internet voice communication. These systems were simple PC-to-PC connections, meaning that each "end user" was required to have the same program running on each end while being connected to the same network. During these early stages the quality of communication was lacking. As techniques and new classes of products were introduced, voice communication over the Internet has been significantly enhanced since 1994. With the enhanced quality of communication along with the explosive growth of the Internet, there came the excitement of the possibility of commercially transmitting high quality voice over the Internet. This excitement sparked numerous efforts in the development of Internet Protocol (IP) Telephony leading to the development of services not only for voice telephony, but also for fax, voicemail, desktop videoconferencing, application sharing and document sharing.

IP Telephony is still in its infancy but has the promise of developing into a bridge between the Public Switched Telephone Network (PSTN) and the Internet, which will allow phone-to-phone, PC-to-phone, phone-to-PC, and fax-to-fax services over IP. Current IP Telephony applications allow users to make a call via their standard telephone and a centralized Internet connection, with the Internet substituting for the PSTN, as illustrated in Figure 2.1. The centralized Internet connection is accomplished via an Internet gateway allowing any user to that Private Branch Exchange (PBX) to make calls

over the Internet, thus eliminating the need for each user to have an independent Internet connection.

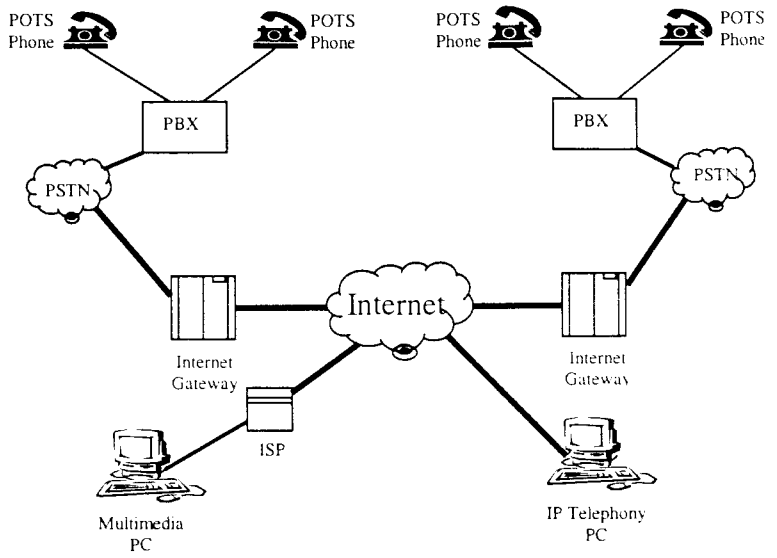


**Figure 2.1. IP Telephony Application**

As a subsystem of IP Telephony, Voice over IP (VoIP) is a system that transmits voice data over the packet switched Internet network running IP. A VoIP implementation is the IP Telephony system without the end telephone signaling and gateway that interfaces with the PSTN. A VoIP implementation is compatible with any IP Telephony scheme; in fact VoIP is encapsulated in the IP Telephony implementation. VoIP concentrates on the aspects of sending voice data over IP, and ignores the signaling and PSTN gateway interface. Used alone, the VoIP implementation could form a PC-to-PC application, or even a PC to "smart" phone application, as shown in Figure 2.2, where a "smart" phone would have circuitry to handle an Internet connection and the tools for processing voice data.

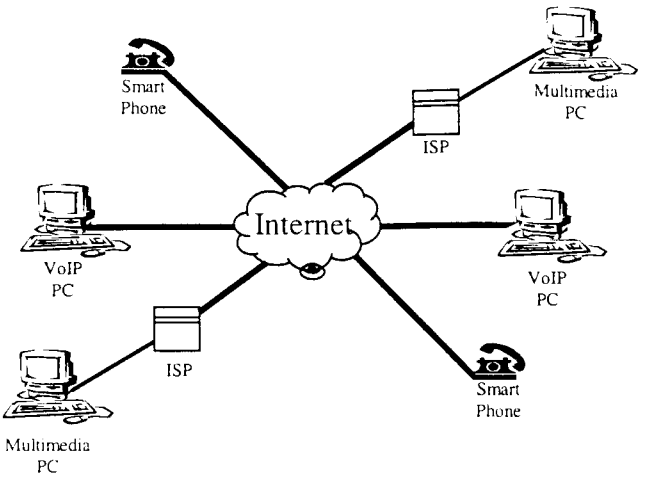


**Figure 2.2. VoIP Application**

## Voice Over IP

Voice is traditionally transmitted over the PSTN. The telephone network is a circuit switched network that allocates a channel for each two-way conversation. When a call is initiated, a circuit is assigned for the duration of the call, giving the talkers a reliable, low latency connection. Other communication channels that carry voice traffic include radio, wireless, and cellular. Similar to the PSTN these communications channels provide a circuit, in some cases a virtual circuit, to the end users. These circuits may be multiplexed upon transmission with other circuits, but nevertheless a dedicated circuit is maintained between each two-way connection for the duration of the connection. Voice over IP involves the transmission of voice over a packet switched network. Packet switched networks were not designed for voice traffic; they were specifically designed for data traffic where each channel is shared by all users. Data traffic is bursty, sent in short spurts; therefore sharing a channel with multiple data connections provides efficient utilization of the channel, but also provides for bandwidth limitations. Moreover,

information is transmitted as packets in contrast to a continuous data stream as with the other communications channels.

Transmission of voice data over an IP network is possible primarily because of the "good fit" between the low bandwidth audio codecs (audio coder/decoder) now available and the characteristics of a packet network. In order to account for the non-stationary nature of speech, audio codecs generally operate on short segments of speech called frames. The speech within each frame is usually assumed to be stationary, and the concatenation of speech frames is used to represent the time varying nature of the speech being coded. The audio codec produces a coded frame, usually greatly compressed as the result of applying a parametric model, for each input speech frame. The frame output of the audio codec provides a good match to the packet format of the IP network. The high compression ratio achievable by many low rate codecs makes low bandwidth transmission across packet networks possible.

As the previous discussion suggests, a VoIP implementation is comprised of two basic parts, the audio codec and the IP network. The primary goal of the VoIP system, as with any communication system, is to maintain a continuous audio stream between end users (or the connection). In order to provide this continuous stream of voice data at both ends, a VoIP service must be able to handle the transition at the transmitter from a continuous speech data stream to an intermediate compressed and packetized format suitable for transmission and back to a continuous speech data stream at the receiver. Again, the mechanism that provides that bridge between the continuous data stream and the packetized, and usually compressed, format is the audio codec as shown in Figure 2.3. The following illustrates the stages the continuous data stream must undergo to complete

an end-to-end connection between users with VoIP connection, as suggested in Figure 2.3:

- Analog speech *sampled*
  - Collected into *frames*
    - *Compressed*
      - *Packetized*
        - *Transmitted* across the network
      - *Unpacketized*
    - *Uncompressed*
- Audio *played out* continuously

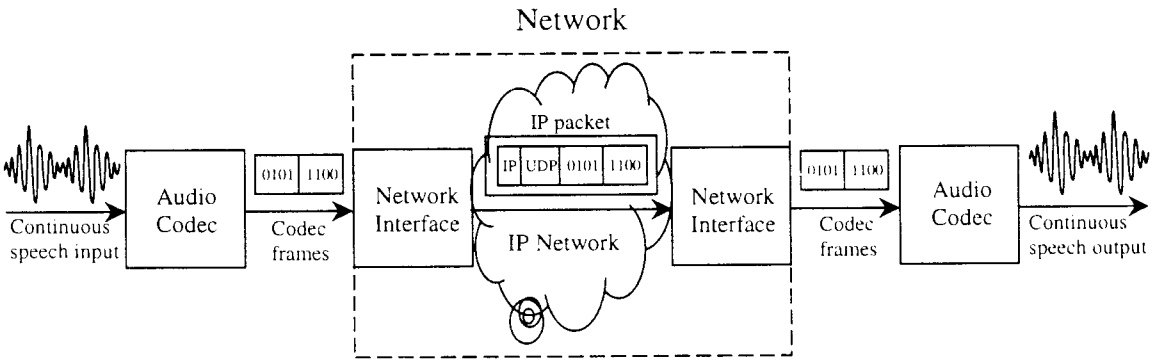


**Figure 2.3. Audio Codec and Network**. Illustration of the stages of an end-to-end IP network voice communication stream using an audio codec.

In the following sections we will discuss additional important aspects of an IP network, including the internet protocol and the audio codec, as well as the interactions that occur between these components. The codec that will be considered is the Mixed Excitation Linear Prediction (MELP) codec developed by Texas Instruments for high

quality low-rate voice communication. We will also briefly discuss the FNBDT protocol and how it relates to the MELP implementation over IP.

**Low Rate Voice Coding**

Speech coding entails the conversion of analog speech into an appropriate digital representation that preserves important characteristics of the speech signal, such as intelligibility and naturalness, while producing a representation that is suitable for transmission or storage. The primary goal of speech coding is usually to achieve a level of compression for use in low bandwidth (or bandwidth constrained) communication and storage applications, which is the goal for VoIP applications.

Speech coders can be classified into two broad categories based on their method of operation: waveform coders and vocoders. Waveform coders such as Pulse Code Modulation (PCM) [25] and Adaptive Differential Pulse Code Modulation (ADPCM) [25] attempt to preserve the original shape of the speech waveform by encoding the signal on a sample-by-sample basis, using temporal and/or spectral techniques [26]. Waveform coders operate by attempting to remove predictable (correlated) portions of the signal or its spectrum. Signal to noise ratio (SNR), based on the closeness of the coded and the original speech signals, is the most often applied measure of performance.

Vocoders such as Mixed Excitation Linear Prediction (MELP) ([4] [5] [49] [50]), Multi-Band Excitation (MBE) ([2] [27]), Code Excited Linear Prediction (CELP) [28], Improved Multi-Band Excitation (IMBE) [29], and Enhanced Multi-Band Excitation (EMBE) 2.4kbps [1], estimate a set of parameters based on the vocal tract model. Unlike waveform coders, vocoders do not attempt to preserve the speech waveform, but attempt

to preserve the perceptual qualities of the speech signal through modeling and parameter estimation techniques. Once the parameters are estimated the original speech waveform (signal) is discarded and a synthetic speech signal is generated using the estimated parameters. The fidelity of a vocoder is based on observed naturalness and intelligibility – both subjective criteria that are in stark contrast to SNR.

Both vocoders and waveform coders have limitations. The performance of waveform coders drops off dramatically at rates below about 16kbps and is thus not appropriate for applications where significant compression is desired. Vocoders are better suited to lower rates (typically 16kbps and below), offering opportunities for substantial compression with a corresponding tradeoff in intelligibility and naturalness.

**Basic Vocoder Operation**

A speech coder, or vocoder, consists of two operational parts: an analyzer (encoder) and a synthesizer (decoder), Figure 2.4. The encoder is responsible for determining, on a short time basis, certain characteristics of the input speech signal, given a specific speech model.

In most cases pitch (fundamental frequency), a single voicing decision, gain, and vocal tract spectrum representation comprise the desired parameters to be estimated from the input speech signal [1]. The estimation techniques as well as the actual parameters vary from vocoder to vocoder.

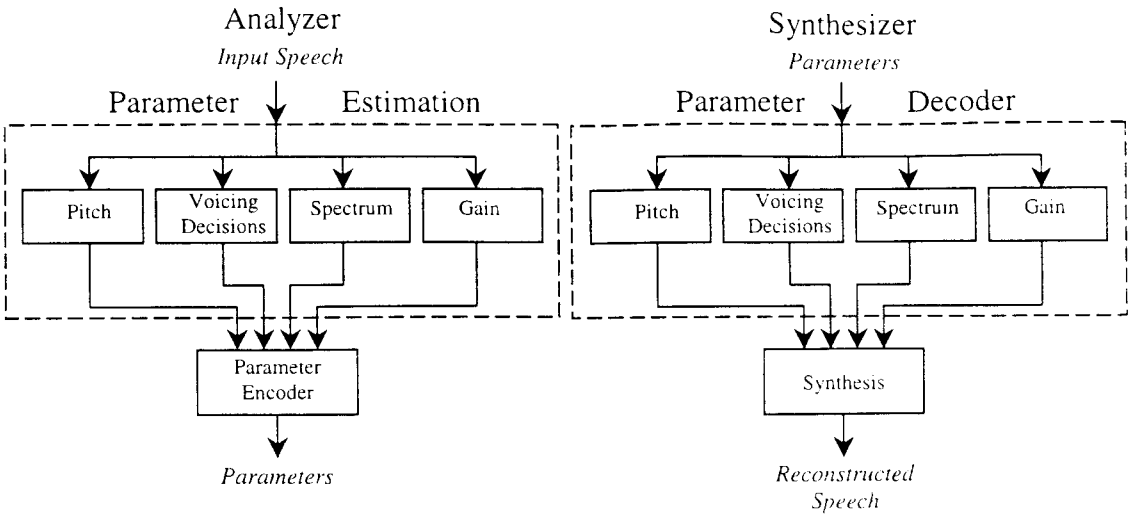


**Figure 2.4. Typical Vocoder**

Once estimated, the parameter vectors are properly coded and sent to the decoder for reconstruction. In the decoder, the analysis model is applied in reverse. The goal of the decoder is to produce the best-sounding speech signal without regard for how the sample-to-sample values correspond to the actual input waveform [2].

### Audio Codec

Codec stands for Coder / Decoder, and is typically used to describe the vocoder algorithm implementation. Audio codecs are in use in most low bandwidth speech systems including cellular telephones, digital voice answering machines, secure telephones, and various PC software applications [3]. The audio codec remains the enabling technology for a VoIP system. In order to transmit voice data effectively across a network such as the Internet using the IP protocol, the audio signal must be significantly compressed because of the Internet's bandwidth constraints. The majority of the audio codecs today offer high quality speech with the compression needed to make this system a reality. Typical codecs used in low rate applications provide compression ratios from 8:1 to 50:1 for a 64kbps-sampled signal. Another advantage to using an audio codec is that the compressed speech is usually output in a framed (packet-like) format, which makes it a good fit for the many packet switched networks that run IP. The codecs' small output frames are ideal for the bursty traffic of a packet network. A few standard voice coders that are applicable for a VoIP implementation are illustrated in Table 1.

**Table 2.1. Standard audio codecs with frame rates [3]**

| Standard | Bit Rate | Frame Length / Lookahead |
|---|---|---|
| G.726 ADPCM | 16, 24, 32, 40 Kbps | .125ms / 20ms |
| G.728 – LD-CELP | 16 Kbps | 2.5ms / 0 |
| G.729 – CS-ACELP | 8Kbps | 10ms / 5ms |
| G.723.1 – Multi-Rate Coder | 5.3, 6.3 Kbps | 30ms / 7.5ms |
| MELP | 2.4Kbps | 22.5ms / 23ms |
| FS-1015 LPC | 2.4Kbps | 22.5ms / 90ms |
| FS-1016 CELP | 4.8Kbps | 30ms / 7.5ms |

For the purpose of this paper we will discuss a VoIP implementation using the new proposed 2400 bps voice standard, Mixed Excitation Linear Prediction (MELP) voice coder. The output format and properties given in the table above is of most importance for the integration of the MELP codec into a VoIP system. Therefore, in the following section we will discuss the MELP coder in detail, giving more emphasis to the estimated parameter output format than the specifics of the coder operation, namely the parameter estimation techniques.

**MELP**

MELP was specifically developed by Texas Instruments under the Defense Digital Voice Processor Consortium (DDVPC) to become a new standard speech coder at 2,400 bps. MELP is currently proposed to become both the new military and federal standards for 2,400 bps high quality speech, replacing Federal Standard FS-1015 (LPC-10) [4], which, by modern standards, produces very low quality speech. MELP at 2,400 bps performs as well as or better than Federal Standard FS-1016 (CELP) at 4,800 bps [5], which is the current benchmark system for low rate speech, making MELP an excellent candidate for most low rate secure voice applications in the government and military.

MELP is a linear predictive coding vocoder; specifically, MELP is a mixed excitation LPC vocoder. MELP differs from LPC [20] in that it has some added features that improve its performance (allowing the synthesizer to better mimic characteristics of natural speech). These enhancements include: mixed voiced and noisy excitation, periodic and aperiodic pulses, adaptive spectral enhancements, pulse dispersion, and Fourier magnitude modeling [5]. Figure 2.5 shows the MELP synthesizer.

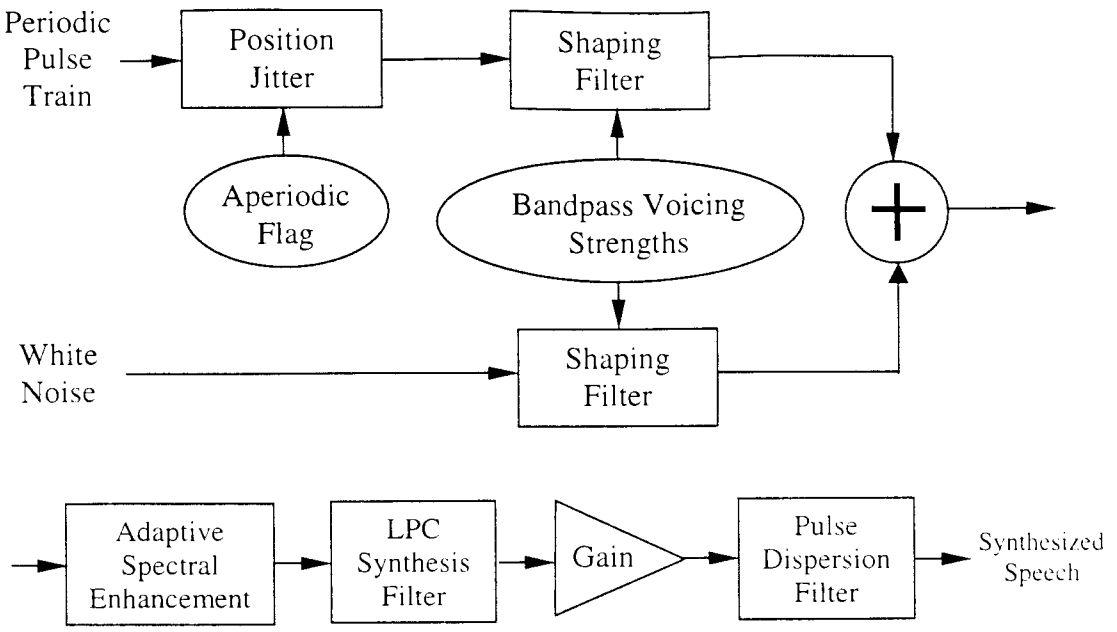


**Figure 2.5. MELP Synthesizer**

MELP produces the following parameters to represent speech frames: Line Spectral Frequencies (LSF), Fourier magnitudes, gain (2 per frame), pitch, overall voicing, bandpass voicing, aperiodic flag, error protection, and sync bit [4]. MELP's frame length is 22.5ms, and to achieve 2400 bps the parameters mentioned above are packed into 54 bits per frame. The bit allocation is illustrated in Table 2 [4].

These parameter bits are scrambled (reordered) to accommodate limited channel bit errors. The parameters are scrambled in a way that enables recovery of the important parameters in case of localized bit errors in the bit stream. The bit ordering matches the LPC-10 Federal Standard [4]. MELP also includes forward error correction (FEC) of unvoiced frames. FEC bits are sent in place of parameters not sent in unvoiced frames, which include: Fourier magnitudes, bandpass voicing, and an aperiodic flag.

**Table 2.2. MELP Bit Allocation**

| PARAMETERS | VOICED | UNVOICED |
|---|---|---|
| LSF parameters | 25 | 25 |
| Fourier magnitudes | 8 | - |
| Gain (2 per frame) | 8 | 8 |
| Pitch, overall voicing | 7 | 7 |
| Bandpass voicing | 4 | - |
| Aperiodic Flag | 1 | - |
| Error protection | - | 13 |
| Sync bit | 1 | 1 |
| Total Bits / 22.5 ms frame | 54 | 54 |

## IP Network Background

### OSI Model for Network Applications

For a better understanding of a VoIP system implementation, we will take a brief look at the Open System Interconnect (OSI) model. The International Organization for Standardization (ISO) proposed the OSI architecture model in the late 1970's to promote compatibility of network designs [6]. The OSI is a seven-layered architecture that is widely used and is a useful framework for network applications such as VoIP. The bottom three layers, respectively, deal with bit transmission, packet transmission on one link, and end-to-end packet transmission; the top layers construct communication services for user applications. Figure 2.6 shows the OSI 7-layer model.

Layer 1, the Physical layer, is responsible for the transmission of bits. This layer actually converts each bit into an electrical (or optical) signal for transmission over the physical link to the destination where the signal is converted back into bits. Layer 2, the Data Link layer, represents the transmission of packets on one given link. For example, in a network configuration where two PC's are separated by two routers, the data link layer is responsible for transmission of packets from PC to router, router to router, and router to PC, each individually.

Layer 3, the Network layer represents the end-to-end transmission of packets. This layer is responsible for allowing packets to find their way through the network. It keeps track of network congestion and uses this information for determining routing information for packets. Layer 3 is where the Internet protocol (IP) resides. Layer 4, the Transport layer, is responsible for the delivery of messages to end-to-end nodes. This layer is where the protocols Transmission Control Protocol (TCP) and User Datagram

Protocol (UDP) reside. Layer 5, the Session layer, is the setup and management of end-to-end conversations. This is where the negotiation and connection is setup between end users, as well as supervision of the connection, if needed. Layer 6, the Presentation Layer, is responsible for formatting, encryption, and compression of data. This is where the audio codec in a VoIP system would reside. Finally, Layer 7, the Application layer, provides network services.

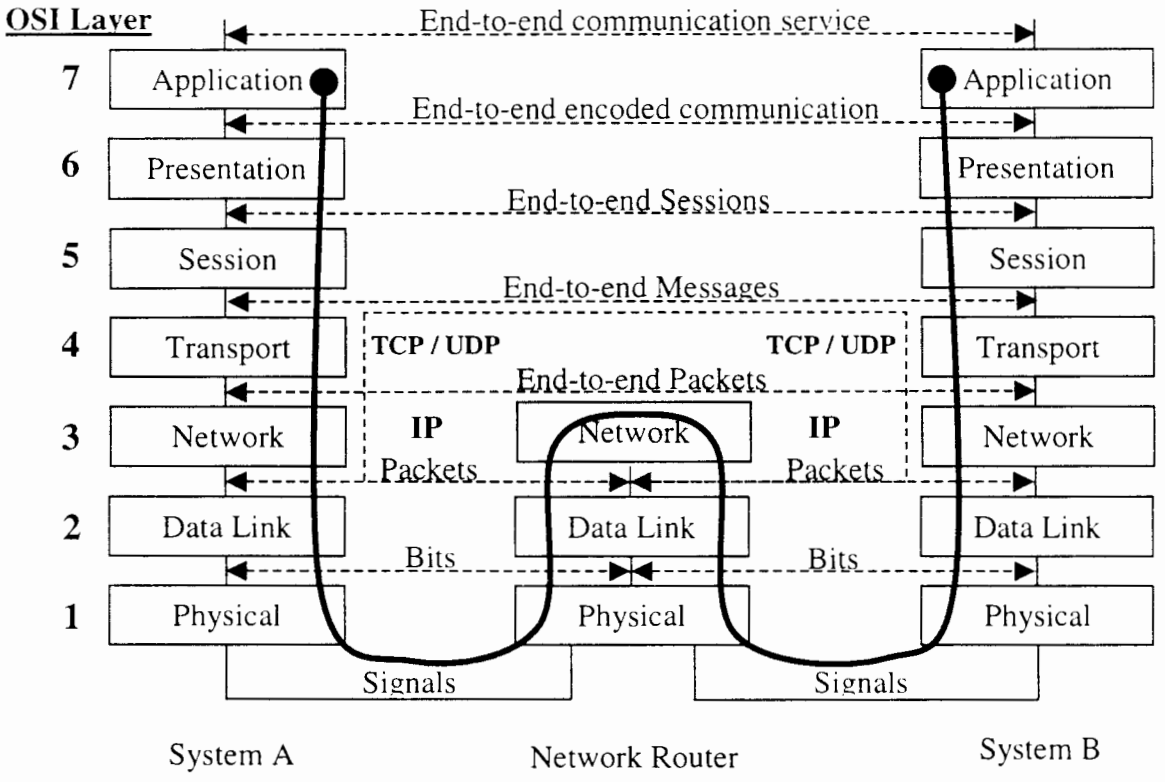


**Figure 2.6. The OSI 7 Layer Model** illustrating conceptual communication between peer layers. The free form line shows the route application data travels. [42]

**IP Network**

The Internet Protocol, or IP, is part of OSI layer 3, the Network layer. IP provides service for the transport protocols in layer 4 (Transport layer). Network layer packets are transmitted as virtual circuits or datagrams. A virtual-circuit is established when a single routing decision is made and each packet that belong to the same conversation is routed along that same network routing path. A datagram transport represents a routing decision that is made for each individual packet belonging to the same conversation (data set). Packets sent as datagrams have the potential to take different routes to their destination, possibly arriving out of order. IP packets are transmitted as datagrams. IP handles routing of packets from a source system, potentially through several intermediate networks, and finally to the destination end system. Figure 2.7 shows a standard IP header layout.

<table>
<tr><td>Bits<br>1-4</td><td>Bits<br>4-7</td><td>Bits<br>8-11</td><td>Bits<br>12-15</td><td>Bits<br>16-19</td><td>Bits<br>20-23</td><td>Bits<br>24-27</td><td>Bits<br>28-31</td></tr>
<tr><td>Vers.</td><td>IHL</td><td>Prc</td><td>TOS</td><td colspan="4">Total IP Length</td></tr>
<tr><td colspan="4">Datagram ID number</td><td colspan="4">Fragmentation</td></tr>
<tr><td colspan="2">Time to live</td><td colspan="2">Protocol</td><td colspan="4">Checksum</td></tr>
<tr><td colspan="8">Source Address</td></tr>
<tr><td colspan="8">Target Address</td></tr>
</table>

Vers. – IP Version
IHL – IP Header Length
Prc – Precedence
TOS - Type of service

**Figure 2.7. IP Header Format**

There are many physical and data link protocols that can be used to transmit IP. One of the goals of the designers of IP was to make IP portable, with the intention of the possibility of making "IP over everything" [6]. Typical data link media are Ethernet,

Token Ring, Fiber Distributed Data Interface (FDDI), Frame Relay, Asynchronous Transfer Mode (ATM) (via ATM Adaptation Layer 5), Point-to-Point Protocol (PPP) which can be used on top of Integrated Services Digital Network (ISDN) or PSTN v.90 modems, wireless communication via wireless modems, and cable lines with cable modems [32]. Packet networks that use the IP protocol are LANs, WANs, corporate intranets, and the Internet.

VoIP systems utilize the Transport layer services to deliver end-to-end messages. These systems can employ TCP or UDP. Both of these protocols use IP at the network layer in the end-to-end delivery of messages. The Transport layer basically converts the end-to-end packet transmission provided by the network layer into an end-to-end message link [32]. Messages are used here instead of packets because when they are grouped together they are meaningful to the applications using them.

TCP provides connection-oriented transmission of stream data. A TCP connection must be established between source and destination before data can be transmitted between the end users. A TCP connection provides reliable service. It insures packets are received and in order. If a packet is not received, TCP will request the resending of the packet. TCP also uses sequence numbers to reorder out-of-order received packets. Figure 2.8 shows a typical TCP header layout.

Delay constraints are so tight in a real-time voice application such as VoIP that a connection-oriented protocol such as TCP cannot be used. While TCP maintains reliable end-to-end transmission through its timeout-and-retransmissions, a single packet loss causes TCP to decrease its transmission rate with either its congestion avoidance or slow start techniques [33] [7]. Therefore it is difficult for TCP to provide adequate throughput

over a lossy path [7]. Though this is true for real-time voice traffic, in an IP Telephony system there is a use for the transmission of control information ([8] [9]). The control information can be used to send signal and feedback information, as well as for the initiation and termination of a session between host machines

| Bits 0-3 | Bits 4-7 | Bits 8-11 | Bits 12-15 | Bits 16-19 | Bits 20-23 | Bits 24-27 | Bits 28-31 |
|---|---|---|---|---|---|---|---|
| Source Port | | | | Target Port | | | |
| Segment Sequence Number | | | | | | | |
| Acknowledgment Number | | | | | | | |
| THL | Resv. | Flags | | Window Size | | | |
| Checksum | | | | Urgent Data Size | | | |

THL - TCP Header Length
Resv. - Reserved

**Figure 2.8. TCP Header Format**

UDP, in contrast to TCP, provides connection-less transmission of stream data. UDP is a datagram type of service. With UDP there is no need to establish a connection between the source and destination. One only needs to know the destination address to send a UDP packet. UDP does not provide reliable service; it is a "best effort" type of service. There are neither acknowledgments sent nor retransmission of data. It is up to the end user to provide this functionality. It is also up to the end user to provide a means of resequencing out-of-order packets. Figure 2.9 shows a typical UDP header layout. UDP is a good candidate for transmission of real-time voice data because it provides efficient, although unreliable, transport of data. The acknowledgements and retransmission of real-time data would add too much delay to a voice conversation [10].

| Bits 0-3 | Bits 4-7 | Bits 8-11 | Bits 12-15 | Bits 16-19 | Bits 20-23 | Bits 24-27 | Bits 28-31 |
|---|---|---|---|---|---|---|---|
| Source Port | | | | Target Port | | | |
| Length | | | | Checksum | | | |

**Figure 2.9. UDP Header Format**

**Future NarrowBand Digital Terminal (FNBDT)**

FNBDT and its corresponding signaling plan [43] (referred to jointly as FNBDT in the following sections), currently under development by the government, provide point-to-point secure communications between end-to-end terminals potentially operating over a variety of narrowband, wideband, and protected networks. FNBDT is a digital solution that incorporates emerging commercially available digital technologies while providing a bridge to the existing analog infrastructure [41]. FNBDT allows communication continuity between users without regard to the communication platform. Further, although FNBDT suggests a "digital terminal", FNBDT encompasses the entire end-to-end communication system. It involves the end system hardware, the terminal equipment (TE), the protocol suite, and the network (unspecified). FNBDT includes the interoperability of the end-to-end hardware such as secure terminal equipment (STE), multimedia PC's, land mobile radios (LMR), cellular phones, etc. The protocol suite involves the data format (data, voice, fax, video, image, etc.), the signaling and control, and the communications modes.

FNBDT provides interoperability through its software configurable hardware, common mode negotiation between peer terminals, and it's "network independent" protocol (signaling plan). FNBDT has a common set of functions that can be either fixed or defined for all terminals at any given time [40]. These functions are used by FNBDT for mode negotiation between terminals to establish the communication modes for the connection. For network interoperability, FNBDT only defines the higher-layer end-to-end protocols leaving the lower-layer network protocol open to accommodate different

networks. With regard to MELP, two aspects of the FNBDT protocol suite are important: the voice communication modes and the high-layer protocol architecture.

FNBDT uses OSI layer 5 (session layer) and higher [6], Figure 2.10, for its end-to-end protocol definition. Leaving the lower network layers "open-ended" allows developers the ability to define the communication platform to deliver the end-to-end data. FNBDT provides a service to the higher layers (layers 6-7) for the delivery of messages. As a layer 5 protocol, FNBDT calls on the lower network layers (layers 1-4) for the transmission of messages. FNBDT provides layer 4 with a data stream to be transferred across the network, relying on the lower layers to transparently deliver the message. By not defining these lower layers, the network of choice becomes transparent to FNBDT. The FNBDT protocol sees the lower network layers as a "bit pipe". The input and output, with regard to layer 4 and below, appears as an I/O data stream in relation to the FNBDT protocol.

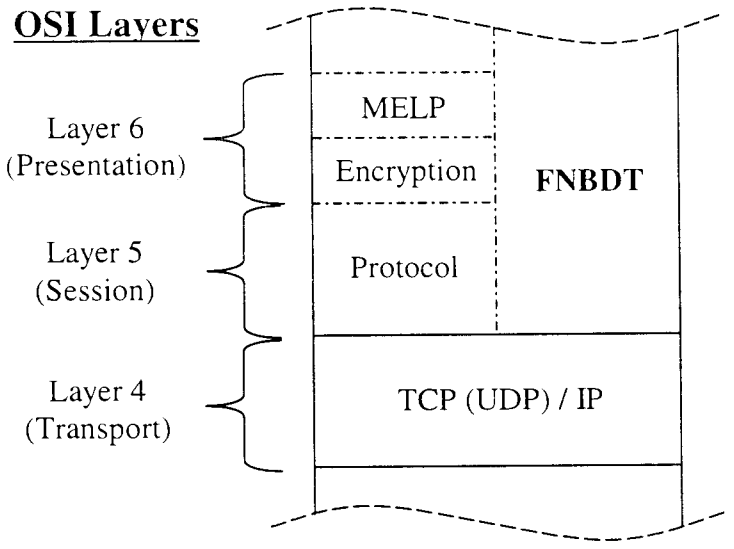


**Figure 2.10. FNBDT Layered Architecture**

In the high-layer protocol, FNBDT defines the proposed federal standard MELP 2.4kbps voice coder for the compression and decompression of voice. MELP resides in layer 6 (presentation layer) where FNBDT encryption is also performed for the secure transfer of voice data.

FNDBT transfers data in accordance with its various communications modes that involve sending secure and unsecure voice data using the MELP codec. The various communication modes describe the data stream, in this case the packet stream. Each mode utilizes a superframe structure that involves the placement of a 54-bit sync management frame followed by voice frames. The sync management frame contains synchronization and encryption information for the corresponding superframe data.

Layer 6 (MELP) calls on the services of FNBDT protocol at layer 5 to provide the end-to-end delivery of MELP frames. FNBDT then prepares the data to be sent according to the FNDBT voice communication modes. The FNBDT protocol takes the MELP frames and groups them, according to the communication mode, into a superframe. During this process the FNBDT protocol calls on the network layers to deliver the superframes to the destination. At the destination, the network layers (1-4) deliver the FNBDT superframes to FNBDT protocol layer 5 where FNBDT dismantles the superframe structure (decrypting the data if necessary) and delivers the MELP voice frames to the MELP decoder.

This discussion will reference three of the possible communications modes, *Clear* MELP voice, secure MELP *Blank and Burst*, and secure MELP *Burst without Blank*. In the implementation discussed in this paper, the sync management frame will only be

simulated; data will not actually be encrypted for the two secure MELP modes and no synchronization will be used.

*Clear* 2400 bps MELP is a raw voice mode that includes a sync management frame followed by 23 MELP voice frames. The sync management frame replaces the first MELP voice frame in the superframe. The MELP voice frame that is replaced is simply discarded. At the receiver the first frame must be compensated for using a frame replacement strategy upon receipt of the superframe. An illustration of the *Clear* MELP mode can be found in Figure 2.11.

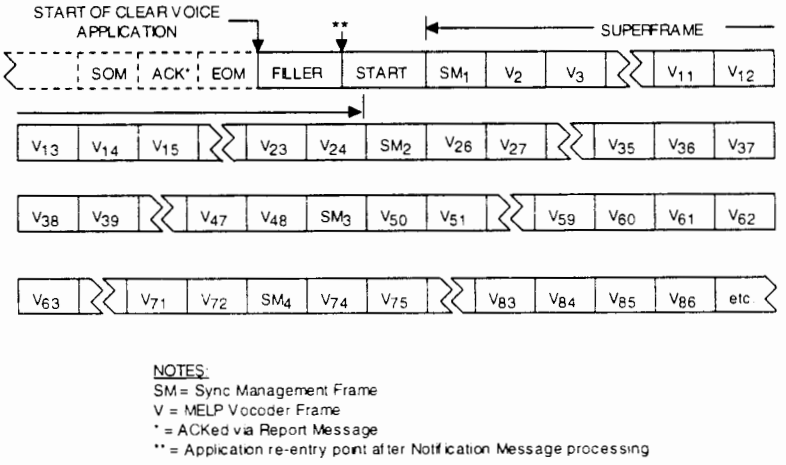


**Figure 2.11. Clear MELP Voice Transmission Format [43]**

The Start of Message (SOM), Acknowledgement (ACK), and the End of Message (EOM) frames noted in Figures 2.11-13 are components of FNBDT's signaling scheme to provide error correction [43]. The FNBDT protocol requires all full bandwidth traffic

be preceded by the FILLER and START pattern. As part of the FNBDT signaling scheme, these frames will not be simulated in the implementation considered later.

The secure 2400 bps MELP *Blank and Burst* mode, Figure 2.12, contains a sync management frame that replaces the first MELP voice frame in a manner similar to the *Clear* MELP mode. As before, the first frame is discarded and replaced prior to the insertion of the sync management frame into the superframe. The *Blank and Burst* superframe includes a sync management frame followed by 23 MELP voice frames.

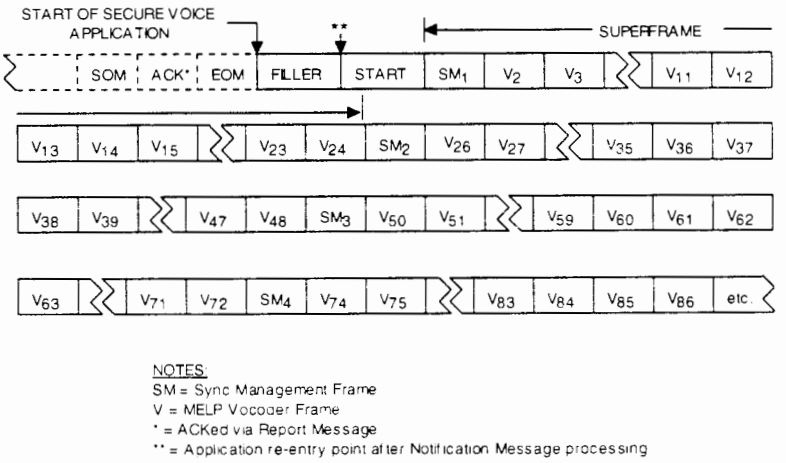


**Figure 2.12. Secure MELP Blank and Burst Transmission Format [43]**

The secure *Burst without Blank* MELP mode, Figure 2.13, contains a sync management frame followed by 24 MELP voice frames. In this case, the sync management frame does not replace the first MELP voice frame; the sync management frame is inserted in between the voice frames. Thus, no correction is needed in the receiver.

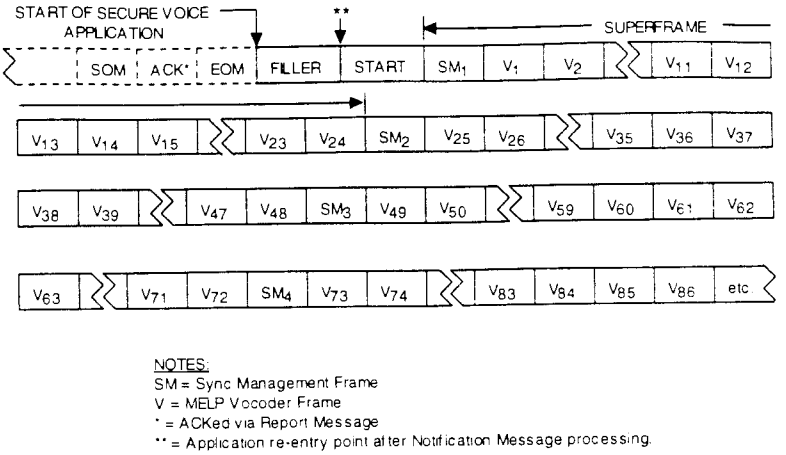


**Figure 2.13. Secure MELP Burst without Blank Transmission Format [43]**

# CHAPTER III

# VOICE OVER IP SYSTEM CONSIDERATIONS

## Introduction

The major issues involved in a VoIP implementation are cost, interoperability and quality of service (QoS). Cost is mainly a corporate initiative, driven by the IP telephony providers, to maximize the profit margin by making the production and operating cost lower than service fees over time. Interoperability is the focus of industry standards organizations to allow for uniformity in the industry. Major considerations for uniformity originate from the telephony front end and audio codec support.

The International Telecommunications Union (ITU) has developed Recommendation H.323, "Packet Based Multimedia Communications" [11]. This recommendation defines call signaling, control messages, audio codecs, and data protocols, thus providing a means of uniformity and interoperability among IP telephony system developers.

The baseline measure for voice communication at this time remains the Plain Old Telephone System (POTS). The existing telephone system (POTS) gives the initial target measure of quality and performance. The voice quality exhibited by POTS is often referred to as "toll quality" (8kHz sampling rate, 8 bits/sample), 64kbps, and is used as a standard measure of speech quality for voice communication applications. POTS does not exhibit large end-to-end delay, missing parts of speech, and talker overlap (one talker suppressing the other speaker's speech). These are all major QoS issues that arise in a VoIP application. QoS describes not only the quality of communication exhibited by the

system, but also the reliability of the connection (communication link). In subsequent sections we will discuss some of the factors that have a direct effect on QoS.

**Quality of Service**

The major factors affecting QoS of a VoIP system are delay, packet inter-arrival time, and loss rate [12]. These factors describe the transmission and receipt of packets (the mechanism by which voice data is transmitted) as they traverse the network. Delay, packet inter-arrival time, and loss rate affect the quality of audio received by the end users of the system. Keep in mind the basis for audio quality is the performance of the audio codec, the naturalness and intelligibility of synthesized speech. However, the perceived end-to-end system audio quality lies in the QoS provided by the system. For instance, an end-to-end system with excessive delay, or extended periods of signal "drop-outs" due to excess packet loss, would be considered a low quality audio system.

End-to-end delay, also referred to as latency, has a direct impact on quality of communication of any two-way communication system. Delay affects the amount of talker confusion and difficulty in maintaining a normal conversation. Delay also gives rise to talker overlap and mutual silence. With increasing delay, talkers are less likely to respond to questions, pauses or interruptions. Talker interruptions or talker overlap also tends to increase with delay. Further discussion on the affects of delay on user conversations can be found in [13] and [14].

Variation in the packet inter-arrival time, the delay between successive packets, is referred to as jitter. If the packet inter-arrival time is too long there can be gaps between words or syllables of the output speech, thus being a nuisance to the listeners. This may

also cause problems in the audio playback by not being able to supply a continuous stream of data for the audio playback device. Therefore, adequate buffering is required to supply the audio playback device with continuous data.

Finally, in a real-time application such as VoIP, when loss is excessive there is degradation in QoS ([9] [7]). Lost and out-of-order packets can cause confusion, and depending on the seriousness of the conversation, can be costly. For example, if there is a military conversation and packets are lost, a potential message can go from being, "You are not clear to fire" to "You are clear to fire". A similar analogy can be given for out-of-order packets.

Delay, packet inter-arrival time, and loss rate are a few of the many factors that influence the QoS of a VoIP system. In the following sections we will probe further into these issues.

**QoS Challenges**

The QoS challenges of a VoIP system lie within the delicate balance between voice quality and delay, with delay being the most constraining factor in QoS [9]. From an end system developer's point of view, the Internet delay is the most unpredictable delay in an IP system. There are several sources that have studied the 'nature' of Internet traffic as it relates to delay and packet loss ([15] [16] [17] [9]). Studies were conducted by taking Internet traffic statistics over several months, with varying data rates, time of day (as it affects congestion), its response to different data loads, and end-to-end distance or "hop count" (number of node (router) hops from source to destination). Statistical properties such as network delay, congestion, packet loss, etc., have been reported. As a

result of the many factors affecting the behavior of the Internet, including unpredictable network traffic, and the vast number of independent networks which make up the Internet, exact loss rates and delay values are difficult to quantify. On the other hand, a few useful characteristics were reported by the sources cited above. For example, Internet packet loss is bursty and correlated, meaning that packets are usually lost in groups, and if packet $n$ is lost then there is high probability that packet $n+1$ will be lost. Moreover, the Internet is *self-similar* in nature, meaning that its characteristics are similar at all time scales. So, the burstiness characteristic of the Internet is the same for short time periods (fraction of second/minutes) as for long time periods (days/weeks). These findings provide useful information for a VoIP system design.

The QoS of a VoIP implementation across the Internet is limited by the performance of the Internet. Excluding the obvious improvements in system design such as quality of speech reproduction in the audio codec, the bulk of a VoIP implementation revolves around optimizing the system based on the Internet's shortcomings. For instance, the selected packet size will depend on network behavior. Larger packets may be discarded under heavy network load at node(s), not to mention that larger packets require more system delay. Small packets will be inefficient in transmission across the Internet because of the relationship between header size and payload (much smaller). As for the receiver end buffering, the size and usage are dependent on the variance of the jitter of packets received from the network. We must also employ packet recovery schemes to compensate for packets lost in transmission across the Internet. The following sections will address some of the common challenges encountered by a VoIP implementation as well as methods for compensating for these challenges.

**Delay**

End-to-end delay, or end-to-end latency, is the delay from the audio input capture device on the system transmitter through the network to the receiver output audio playback device. End-to-end delay is measured from the input microphone of the transmitting system to the output of the speaker at the receiver, Figure 3.1. This is also referred to as "one-way" delay. The delay in a VoIP conversation is made up of two "one-way" delays, referred to as round-trip delay. A VoIP system can exhibit one-way delays ranging from approximately 30ms upwards to a few seconds. End-to-end delay is largely dependent on the network delay, while other factors affecting end-to-end delay are related to system design, such as packet size, buffering, the audio codec algorithm, and the operating system latency.

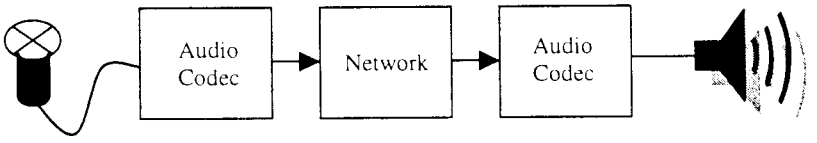


**Figure 3.1. End-to-end delay block diagram**

There have been numerous studies on what constitutes quality voice communication as well as what performance is classified by end users as tolerable and intolerable communication (affected by delays). There are some definite boundaries, upper limits, on what is considered quality communication service, but there is a "gray area" as to what a user will tolerate in a system. For example, wireless telephone users have been known to accept lesser quality service in exchange for the convenience of high

mobility [18]. The same is true in regard to cost; if a service is significantly cheaper, users are more tolerant of degraded service. Therefore, there seems to be among users a balance between added latency and perceived value added by the system [18]. The user's experience also plays a role in the toleration of delay. [14] states that if users are accustomed to a certain level of quality in a service (delay in this case) they are more critical of a system that provides lesser quality with, of course, the perceived quality being relative to the application. For example, cellular service has quality relative to cellular service, and is not comparable to the quality provided by the POTS. Figure 3.2, taken from [18], shows the relationship of perceived link quality versus one-way latency, as perceived by users regarding a VoIP system as a cost-reduction system. As shown in Figure 3.2, 300ms [19] is an upper bound on one-way delay of a system that delivers quality communication service. Above 150ms the quality of communication degrades, with users viewing anything above 450ms as unacceptable. Other studies show that users view delays in excess of 300ms as a half-duplex connection instead of a two-way conversation ([9] [3]). User toleration varies, and often depends on the particular application. More critical users may require a delay of 200ms or less, while more tolerant users are often satisfied with delays of more than 300ms [9].

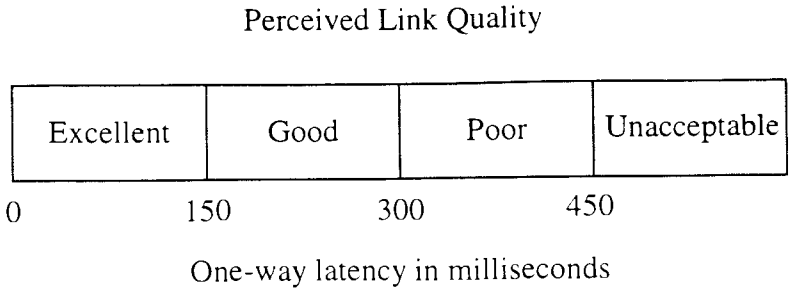


**Figure 3.2. Quality Perception vs. Latency**

The POTS has been reported as having a round-trip delay of less than 150ms [18], and in some cases has been reported as a having a one-way delay as low as 50ms or less. Obviously with its low delay and high QoS, to which users are accustomed, the POTS remains the metric of QoS for voice-over-packet systems. Because of the unreliability of packet networks such as the Internet, QoS equivalent to the POTS is not yet attainable. But, users are more tolerant to voice-over-packet network systems because they are considered cost-reduction applications.

### Packet Overhead

Each packet that is sent across the network contains destination and identification information along with the data (voice data-payload). The destination and identification information is called the packet header. In a typical VoIP implementation the packet header is made up of an IP header (20 bytes), UDP header (8 bytes), and a Real-Time Protocol (RTP [48]) header (12 bytes), Figure 3.3. The IP header, Figure 2.7, has the destination address information for routing the packet across the Internet. The UDP

header, Figure 2.9, contains the port and socket information of the destination system. The RTP header contains QoS information such as time stamps, sequence number, and data synchronization information. The header accounts for 40 bytes of the packet size and is considered the packet overhead.

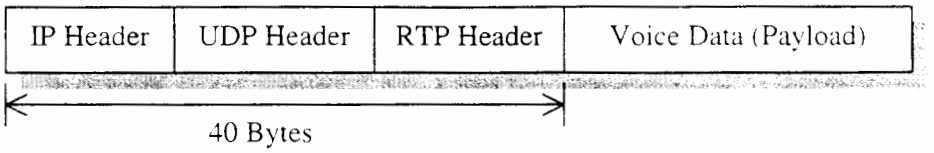


**Figure 3.3. VoIP packet with headers [18]**

Packet overhead is an issue in a real-time application because it may dramatically reduce network efficiency when sending small amounts of data. For efficient use of the network the data must be significantly larger than the header. Otherwise the majority of the network bandwidth will be used to carry header information. With a low bandwidth codec like MELP, which compresses 22.5ms of voiced speech into approximately 7 bytes, a packet containing only a single MELP frame would have an overhead that accounts for 85% of the packet, thus significantly reducing the network efficiency. On the other hand, including many voice frames in a packet causes the system delay to increase, because each additional frame requires the equivalent frame length of additional delay due to buffering. Furthermore, as the number of frames per packet increases, the susceptibility to packet loss also increases, considering larger packet size is associated with higher packet losses. The possible result is the loss of a significant amount of data on the receiver system end. There are efforts being made to reduce the packet overhead as a result of header information, through the use of header compression algorithms. But, in

the meantime, careful consideration must be made in determining how many voice frames to pack into a single packet.

**Packet Loss**

Another major factor that affects the QoS in a VoIP system is packet loss. Packet loss can occur as a result of buffer overflow at network nodes due to heavy network loads. It can also occur due to bit errors incurred by packets as they make their way through the network.

There have been numerous studies on Internet packet loss statistics including [17], [12], [21], and [7]. Packet loss is known to have some correlation with packet size, network congestion and network delay. As the packet size increases, so does the probability that a network node will drop the packet during congested conditions, while shorter packets are more likely to make it through. If the network is congested, buffer overflow can occur and packets are lost. Delays in the network may cause packets to arrive at their destination too late, after the frame's scheduled playback time, and are useless to the receiver and considered lost.

Packet loss is "bursty", meaning that packets are usually lost in sequential groups. When one packet is lost there is a high probability that a series of subsequent packets will also be lost ([7] [17] [12]). This can be harmful to a loss sensitive speech application because a considerable amount of information can be lost at one time, in contrast to isolated packets. Techniques are available to help remedy this problem. Some implementations use feedback from the receiver to inform the transmitter of lost packets, so that the transmitter can "throttle back" on sending packets. This may have the effect of

giving a lower QoS in one-direction versus the other to compensate for bad network conditions [7].

Packets lost in bursts account for the majority of the packets lost over a long period of time. But, overall packet loss gap is usually close to one or two packets [17]. In this case of only one or two sequential losses, FEC techniques can be used to correct these losses.

Packet loss has varying effects on the system. If large packets are used, with many voice frames included in a single packet; a lost packet will constitute a large amount of information loss. In a speech application this could mean many milliseconds of speech loss that will be easily noticeable. In applications using small packets, only a small amount of speech may be lost and may go undetected.

There are several techniques for recovery of lost packets in audio applications, many of which are highlighted in [22]. There are two basic categories of recovery schemes, transmitter and receiver. A number of techniques are not relevant to real-time interactive audio applications because of the delay they impose on such a system. Others are more applicable to real-time applications but may lack in quality of reproduced audio compared to non-interactive audio applications.

### Transmitter Based Recovery Schemes

Peerkins *et al.*, [22], summarizes a media-independent transmitter based FEC scheme that uses block coding to produce additional packets to aid in the correction of lost packets. These additional packets may contain a parity coding or Reed-Solomon coding of whole packets. These codes were originally developed for detecting and

correcting errors in the bit streams of a transmitted packet. In this case they are used to code the bits of the entire packet for correcting the loss of a whole packet. The major drawbacks to this scheme are the additional delay, increased bandwidth, and difficult decoder implementation. Another FEC implementation, media-specific, involves sending multiple copies of each frame (not the whole packet) of voice data in successive packets, Figure 3.4. This can be accomplished by adding only one frame of delay, and no significant bandwidth or complexity increase. Low latency audio codecs make it possible to implement this scheme without a great increase in delay, although [22] suggests that the duplicate frames can be encoded by a secondary method that has lower latency and quality. This would further reduce the added latency, but again the delay imposed by low rate codecs may suffice.

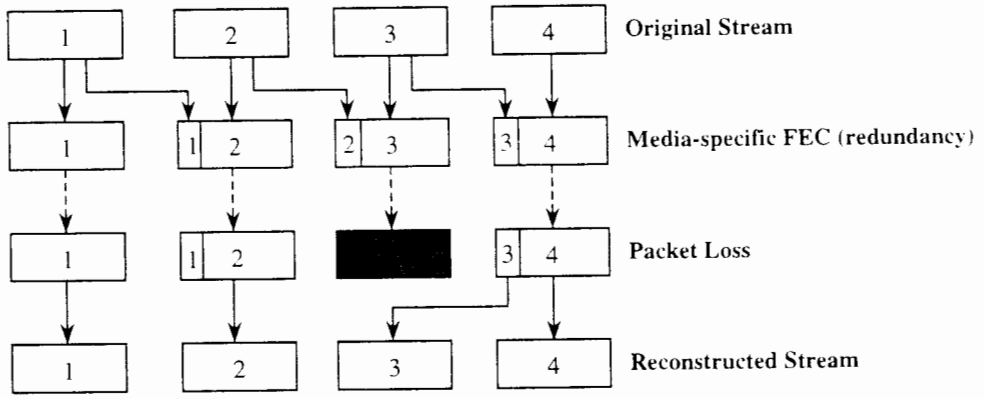


**Figure 3.4. Repair using media-specific FEC [22]**

Interleaving is another transmitter-based scheme. It involves resequencing frames before they are transmitted and returning them to their original order at the receiver. Each frame is interleaved across multiple packets, so if a packet is lost then only small amounts of data will be missing at the receiver end, a condition that is easier to repair

than if a larger amount of data was lost. In order to implement this scheme, significant buffering is needed on both ends, which adds to the end-to-end system delay. Therefore this is not a suitable application for a delay-constrained implementation such as the one considered here.

Other transmitter-based schemes include adaptive packetization and concealment [23]. This technique looks to exploit the slowly varying properties of speech by detecting voiced areas of speech. The small voiced speech segments (individually) are then placed into individual packets. As a result, small packets are used for voiced speech and longer packets for unvoiced speech. If a packet containing a segment of voiced speech (important perceptually) is lost, then an adjacent speech frame can be used to replace the missing frame. The idea is that small voiced speech segment losses can be concealed better, perceptually, than large segments because of the slowly varying quality of voiced speech.

Retransmission is yet another form of error correction. If a lost packet is detected, this scheme simply sends out a request for the retransmission of the lost packet, and the transmitter retransmits the lost packet. This scheme of course will not suffice in a delay-constrained application such as a VoIP system.

The Naturalness Preserving Transform (NPT) for replacement of missing speech frames presented by [51] applies a transform to the voice data prior to transmission and the inverse transform at the receiver. The NPT is based on a stochastic operator that uses the normalized Hadamard matrix [52]. The information in the original speech matrix is transmitted as additional overhead for the reconstruction of the missing speech segment at the receiver. While this method produces high quality speech, it adds additional

overhead, delay, and complexity. The NPT method requires system modifications and therefore may not be appropriate for this implementation.

### Receiver Based Recovery Schemes

There are several error concealment techniques that may be used by a receiver system. There are three categories of these techniques: insertion, interpolation, and regeneration [22]. The insertion techniques involve simple frame replacement. They are simple implementations having little complexity and low delay. They include splicing adjacent frames, silence insertion/replacement, noise insertion/replacement, and adjacent frame substitution. If a lost packet is detected, these schemes simply replace the missing packet using one of these methods. Splicing adjacent frames cause playback timing problems as well as unacceptable quality, while silence and noise replacement yield sub-par quality according to [22]. [22] also indicates frame duplication gives acceptable quality, and frame duplication with successive frame fading gives better quality.

Frame duplication with successive fading for successive packet loss provides a good compromise between the simple frame duplication and silence substitution. It involves fading the replicated frame over a time period to gradually transition into a silent period or zero amplitude. This alternative on average would provide better performance over abrupt insertion of silence or continued replication of packets during loss, which can become annoying to listeners.

Interpolation and regeneration techniques significantly add complexity, but only provide an incremental increase in quality [22]. Waveform substitution uses adjacent waveforms to generate the replacement frame's waveform. Wasem, *et al*, [44], discusses

both the one-sided and two-sided techniques. In the one-sided case, Wasem uses the last M samples of speech from the last received packet. Then a template is used to search a window of prior received speech samples in an attempt to obtain a match for use in replacing the missing speech. The two-sided technique uses a template search window that looks at both past and future received speech.

A variation of waveform substitution is pitch waveform substitution, which uses pitch detection to provide a waveform (for the missing frame) with the appropriate pitch length. Goodman, *et al* [45], describes a technique that uses pitch detection to determine if the packet received prior to the missing packet is voiced or unvoiced. If the previous packet is considered unvoiced then the previous packet is substituted for the missing packet. If the previous packet is considered voiced, a single (T sample) pitch period is calculated and the missing packet is replaced with successive repetitions of the calculated T samples.

The receiver based Markov Chain Prediction technique models voice coding parameters as states in a Markov chain [51]. It predicts the missing speech parameters based on information from previous packets and weighted probabilities from a state transition matrix. The state transition matrix requires a complex training procedure to produce the weighted probabilities of speech segments. These state transition probabilities requires large amounts of memory. This method may not be appropriate for this implementation because of the complex training procedure and the excessive memory required to produce the state transition probability matrix, as well as the required system modifications.

A variety of other loss recovery techniques that appear in the literature recently include time scale modification [46], interpolation of transmitted state [47], and others.

Of all the packet loss recovery techniques mentioned above, [22] recommends transmitter based media specific FEC for a real-time interactive IP telephony application, because of the low delay and bandwidth overhead. The packet repetition with fading receiver-based scheme is also a good candidate because of its low complexity and acceptable quality. Keep in mind these recovery schemes are ideal for low-level random packet loss and may not perform well under continuous high packet loss situations.

**Jitter**

Network conditions are constantly changing and packets experience different delays as they traverse the network. When the delay between packets, the packet inter-arrival time, varies it is referred to as 'jitter'. Packets experiencing different queuing delays at network nodes or packets taking different routes to the destination as a result of the IP protocol can cause jitter. In some instances packets may arrive out of order.

The varying inter-arrival time imposed on packets by the network or transmitter system results in packets arriving at the receiver (destination) at varying times thus making it impossible for the receiver to playback continuous audio. Furthermore, if a packet does not arrive in time for the scheduled playback, the packet must be considered lost. Once a packet is considered lost, something must be done to compensate for that lost packet or there will be gaps in the output audio, Figure 3.5(a). To prevent "starvation" of the audio codec and playback routine the receiver has to compensate for packet jitter.

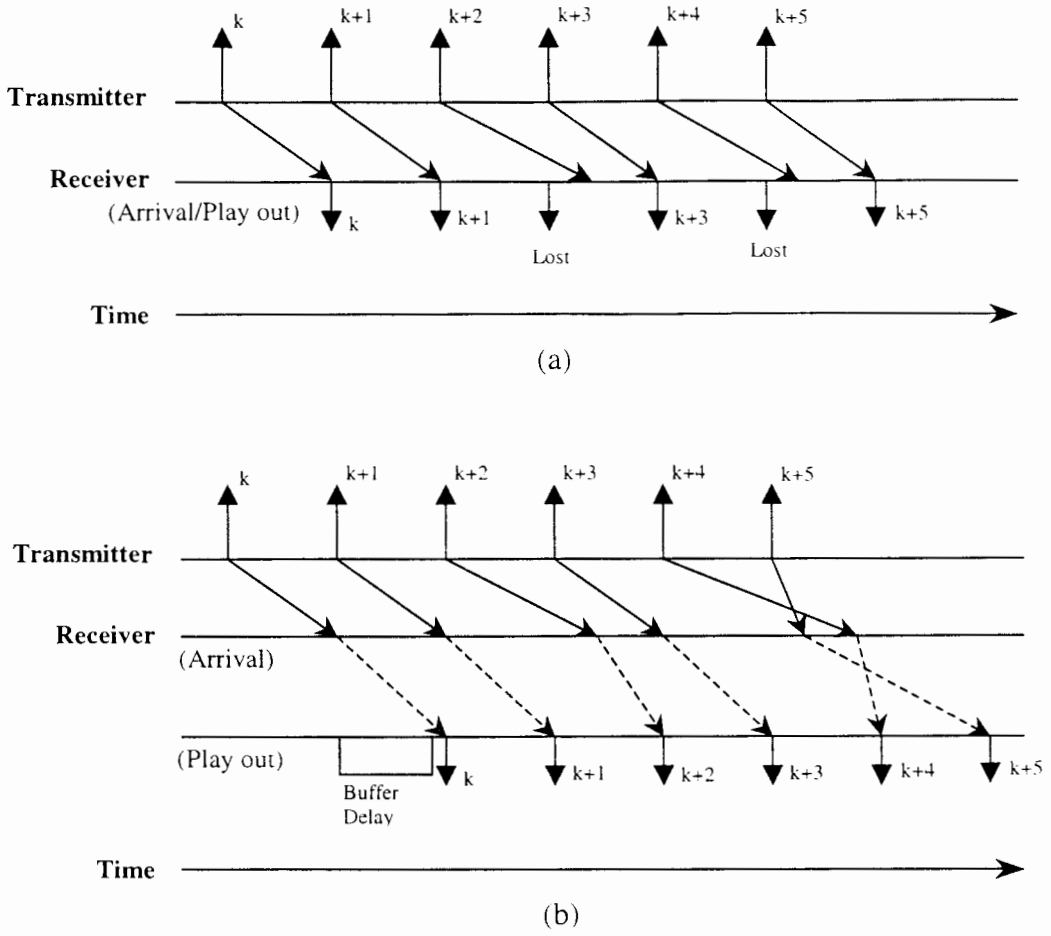


**Figure 3.5. Packet delay problems** (a) late packets are considered lost at play out (b) buffering corrects late and out-of-order packets.

Receiver buffering is the primary mechanism for correcting for or smoothing jitter, Figure 13.5(b). Adequate buffering must be in place to compensate for the varying packet arrival time. However, if a packet is too late, exceeding the amount of buffering provided, it will be discarded. Consequently, as the number of buffers increases to compensate for jitter, the overall system delay increases. Therefore, careful consideration must be made in determining the amount of jitter buffering to include.

There are some adaptive techniques for adjusting the number of jitter buffers based on network delay. Deleon, [24], summarizes some of the reactive (non-predictive) algorithms that use the average network delay of buffers received to determine adequate jitter buffering. DeLeon [24] also presents an adaptive prediction technique that uses a Normalized Least-Mean-Square (NLMS) adaptive predictor to predict the network delay in order to smooth network jitter. DeLeon states that his approach minimizes the jitter buffering by accurately predicting rapidly changing network delay conditions. Consequently, slightly more packets are late (approx. 0.2%), as opposed to the other techniques (well under 0.1%), because his approach yields a tighter delay window [24]. The adaptive approaches optimize their technique to find the delay that keeps the percentage of late packets low. The predictive approach attempts to minimize the delay while achieving a tolerable percentage of late packets. Thus, his predictive approach lowers the total end-to-end delay at the expense of slightly increasing the number of late packets. This approach may prove useful if used in conjunction with the loss recovery techniques mentioned earlier.

**Conclusion of QoS Challenges**

There are several challenges in providing high QoS in a VoIP implementation. Some of the major challenges include dealing with delay, packet loss, jitter, and packet overhead. With these challenges come several trade-offs, such as robustness to error versus delay. As the number of buffers is increased to compensate for out-of-order packets and jitter, the overall system delay is also increased. There is also a tradeoff of delay versus efficiency. The packet overhead has the potential to dominate the payload of

the packet, decreasing the network efficiency. By increasing the number of voice frames included in a packet, the network efficiency is increased at the cost of an increase in the system delay. Large packets contain large amounts of data and if lost, system performance is significantly degraded. Lastly, there is a three-fold trade off: packet size, congestion and packet loss. The larger the packet the higher the probability it will be dropped by a network node, and will increase network congestion. The increased network congestion will increase delays. Large packets are also more susceptible to bit errors thus contributing to packet loss.

All of the above trade-offs are somewhat connected to each other. Minimizing delay, an optimal packet size, receiver jitter buffering, and adequate packet loss techniques make up the challenges to consider when developing a real-time VoIP application with a high QoS.

# CHAPTER IV

# DESIGN OF A VOIP SYSTEM USING THE MELP 2400 CODEC

The objective of a VoIP system design is to produce quality voice communication without excessive end-to-end delay, delay variation, and degraded performance due to packet loss. The system design consists of two major components: the audio codec and the network. In order to achieve the goal of quality voice communication, the codec must seamlessly adapt to the network, and the network must provide a quality link from the source to the destination.

Before proceeding with the VoIP system design, let us first take a look at the various components that make up the VoIP system, starting with the 2 major components: the audio codec and network. As implied by Figure 2.3, the codec component must provide an interface to the user as well as to the network component. The network component would mutually have an interface to the codec plus an interface to the IP network.

In order for the codec to meet the input requirements of the network component, the codec must provide data to the network in the form of packets. The codec produces an output in a frame-oriented format so the transition to packet format is easily achievable. The assembling of codec output frames into a packet is referred to as *packetizing*. The interface that performs the packetizing will be known as the packetization component or packetization module. The codec must also have an interface that accepts input speech for encoding, as well as a means of outputting decoded (synthesized) speech. This is accomplished with an analog-to-digital converter for the input speech and a digital-to-analog converter for decoded output speech.

The network component will accept the outgoing packet from the packetization component and send it through the IP network to its destination, and likewise receive incoming packets from the IP network and pass them to the de-packetization component. To transmit and receive packets from the IP network, the network component must provide an interface to the IP network. The following section will briefly discuss the IP network interface as well as the other components that make up a VoIP system.

### VoIP System Components

For an end-to-end PC-to-PC VoIP system, there is a minimal set of system components necessary for a complete system. On the sending side, these components include: Input Digital Interface, Audio Codec, Packetization Module, Network Module, and the Network Interface Module. The receiver side modules are arranged in opposite order. Figure 4.1 shows the end-to-end VoIP system components. There can be other components in a VoIP system, such as security (encryption/decryption), echo cancellation, voice activity detection, user interface, etc., but this paper will focus on those previously mentioned. An overview of these components follows.

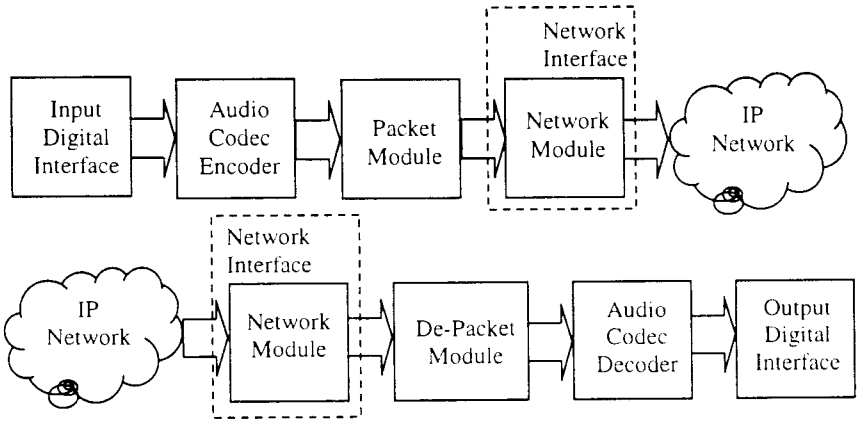


**Figure 4.1. End-to-end VoIP system**

The *input* and *output digital interfaces* are the system interfaces to the outside world. The *input digital interface* captures (digitizes) analog speech input in real time and converts it to digital pulse code modulation (PCM) format. The data is properly windowed and framed and fed to the *audio codec*, Figure 4.2.

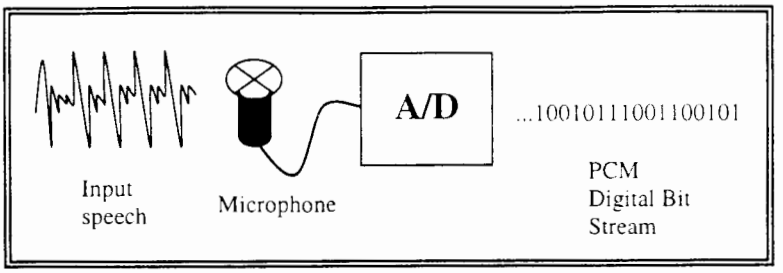


**Figure 4.2. Input Digital Interface**

The *output digital interface* takes the digital speech frames from the *audio codec* and converts them in real time into a continuous analog speech signal for audio playback, Figure 4.3. The *output digital interface* D/A converter contains a First In First Out (FIFO) buffer that removes timing jitter of speech frames as a result of the varying inter-arrival time of packets received from the network.

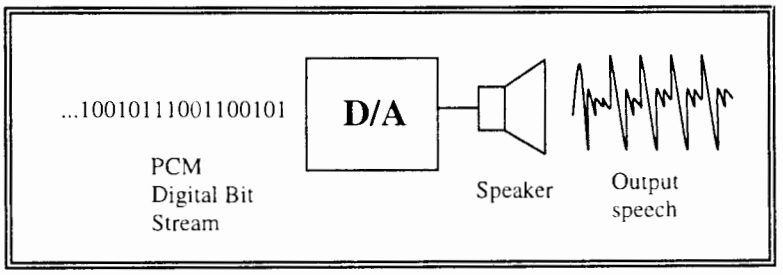


**Figure 4.3. Output Digital Interface**

The *audio codec* provides the compression and decompression of the input and output audio speech signals respectively. It takes PCM digital speech frames from the

*input digital interface* and compresses them for packetization by the *packetization module*, Figure 4.4.

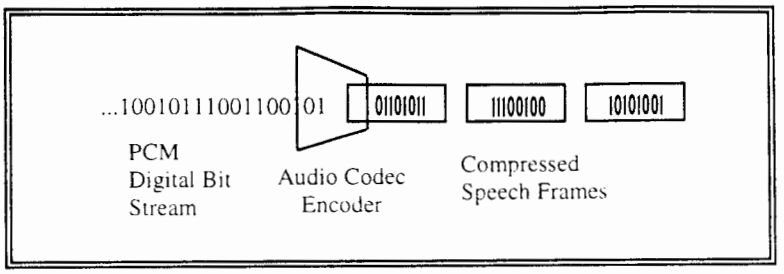


**Figure 4.4. Audio Codec Encoder**

The complement of this process is to decompress the compressed audio from the *de-packetization module* into frames of digital speech to be fed to the *output digital interface*, Figure 4.5.

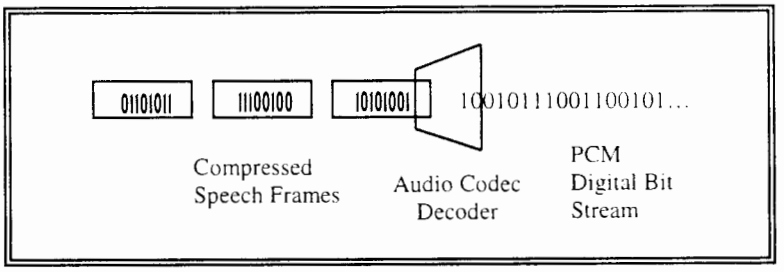


**Figure 4.5. Audio Codec Decoder**

The *packetization module* takes compressed frames of speech from the *audio codec* and assembles the data into packets suitable for transmission on the network. It attaches header information to outgoing packets that may includes a sequence number, time stamp, and other identifying information necessary for real-time operation. Figure

4.11. We will refer to this information as the OSU header, which will be described in more detail later.

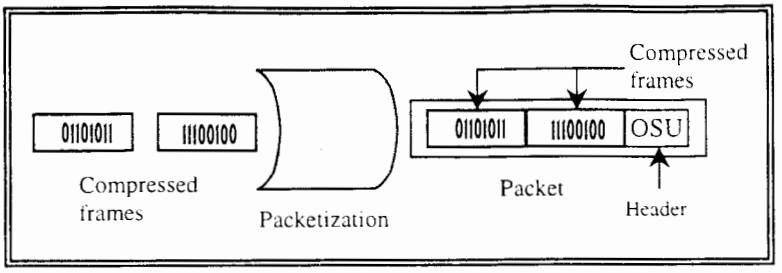


**Figure 4.6. Packetization**

For incoming packets, the *de-packetization module* buffers received packets, and removes and evaluates OSU header information. The module resequences out-of-order packets (UDP packets only), and checks for lost packets. It disassembles incoming packets leaving frames of compressed speech that are fed to the *audio codec* for decompression, Figure 4.7.

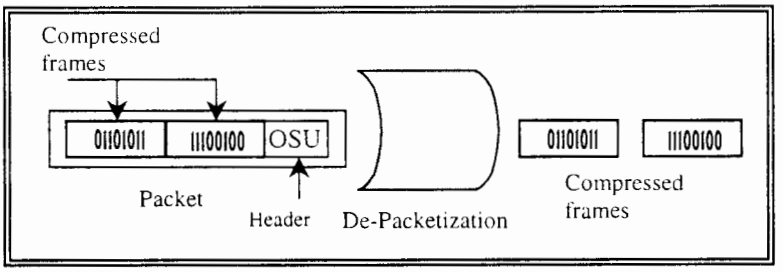


**Figure 4.7. De-Packetization**

The *network interface module* is responsible for establishing a network connection between end-to-end machines. This involves setting up protocols (TCP/UDP), sockets, ports, and connection handshaking. The network interface may also supervise communication between end users, via a separate control channel.

The *network module* is responsible for transmitting packets to, and receiving packets from, the network. Thus it adds and removes header information to packets for network transmission and reception. The *network module* prefixes TCP or UDP headers to the voice packets depending on the protocol used. It also adds an IP header to the UDP or TCP packet, Figure 4.8. For incoming packets it removes IP and UDP or TCP headers, respectively.

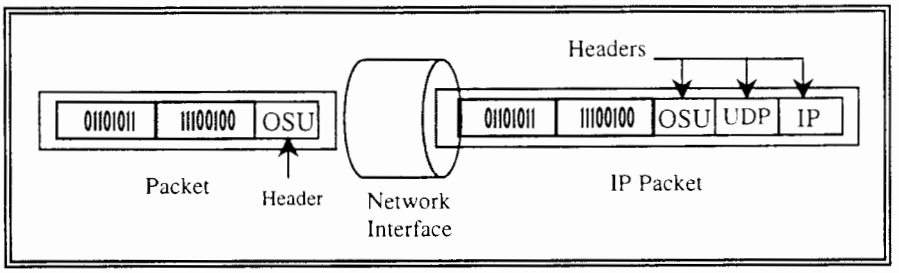


**Figure 4.8. IP Network Packet Process**

The *network module* uses IP to route outgoing packets through the network to their destination based on the destination address given in the IP header. The TCP and UDP headers provide the port and socket address at the destination system. This module also buffers incoming and outgoing packets to prevent packet loss because of buffer overrun. For example, the network may be congested and unable to send a packet immediately, or a receiver process may not be ready to process incoming packets because of a short inter-arrival time between packets.

## Design Components

The system design approach centers on the audio codec and the network. Since an existing audio codec, TI MELP 2.4kbps version 1.2 (MELP) ([5] [49] [50]), is being used, the system design must meet the input and output format requirements of this specific codec. The network component must also address input and output requirements, and a substantial amount of the design will involve alleviating potential QoS problems that can occur with an IP network. The areas that will be addressed as part of the system design include: audio codec input and output requirements, network input and output requirements, and network QoS issues.

Network input and output requirements involve packet size (number of frames per packet), network interface, and the transport protocol. Network QoS issues include minimizing end-to-end delay, jitter removal, and packet loss compensation.

The design is based on the use of MELP as the audio codec and the Internet as the IP network. Several of the design decisions may not be the only or best solution for all cases, since in a VoIP system design there are several trade-offs to consider. Many of the design considerations will depend on the type of network connection, available network bandwidth, communication requirements (user expectation), and the particular audio codec being used. Other trade-offs weigh heavily on desired system characteristics, such as robustness to error versus minimizing delay. For example, if robustness to error is the top priority then the system will have an increased delay.

The primary goal of this design is to produce a system that can be used to investigate various design trade-offs, including the effects of packet length, loss recovery schemes, and other system constraints on system performance. The system design

considered in this paper represents a base system or test bed that can be used for studying the effects of design decisions on QoS.

**Application Platform**

Compute power is a key component in successfully implementing a VoIP system due to its real-time requirements. The audio codec usually requires higher computing power as the compression ratio increases, and the system must be able to sustain real-time operation.

Bandwidth is also an issue to consider for the feasibility of this application. The bandwidth of the connection has a direct effect on the design constraints on the system. Lower bandwidth connections require higher compression codec algorithms to make transmission of voice possible. Equation (4.1) shows the relationship between the time it takes to transmit a packet, *TransPK*, packet size, and connection speed.

$$TransPK = \frac{\text{Packet Size (bits)}}{\text{Line Speed (bits/sec)}} \qquad (4.1)$$

With a constant line speed, the packet transmit time is directly proportional to the packet size. As the packet size increases the transmit time also increases. Since delay is an important aspect of a real-time application, a packet transmit time less than the packetization delay, the time it takes to accumulate a packet, $\Delta_{Pkt_i}$, (4.2a), is desired so as not to add additional delays to the system, as indicated in (4.2b).

$$\Delta_{Pkt_i} = k * t_f , \qquad (4.2a)$$

where $k$ is the number of MELP frames included in a packet and $t_f$ is a MELP frame length (4.5).

$$TransPK < \Delta_{Pkt} \tag{4.2b}$$

Ideally, the current packet should be sent before the next packet is ready for transmission, thus avoiding an additional "wait time" delay by having to wait on the current packet to transmit before the next (ready) packet can be sent.

In a low bandwidth connection with highly compressed voice frames, if only one or two frames are included in a packet, a large percentage of the transmission would be overhead, thus effectively reducing the connection throughput as well as network efficiency. With higher bandwidth connections overhead may not be an important issue.

**Digital Interface Design**

The MELP audio codec accepts frames consisting of $N$ =180 16-bit samples of input speech and produces output frames consisting of nine 32-bit integers. We assume that the input digital interface will properly sample the input analog speech, as shown in (4.3a) and (4.3b).

$$s(n) = s_a(nT), \tag{4.3a}$$

where

$$F_s = \frac{1}{T}. \tag{4.3b}$$

The input speech is sampled at a sampling rate of $F_s$ = 8,000 samples/second, and each sample is quantized to $B$ = 16 bits, giving a raw data rate of $F_s * B$ = 128,000 bps. The sampled speech is then collected into frames with window $w(n)$ of length $N$ =180 samples, (4.4a), (4.4b).

$$f(n;m) = s(n)w(mN - n), \tag{4.4a}$$

where

$$w(n) = \begin{cases} 1, & n = 0,1,\ldots,N-1 \\ 0, & otherwise \end{cases} . \qquad (4.4b)$$

The accumulated frames, of length $t_f$ seconds, (4.5), are then delivered to MELP for processing.

$$t_f = \frac{N}{F_s} \qquad (4.5)$$

The output digital interface will accept the uncompressed synthesized speech samples from MELP and play them at the sampling rate $F_s$.

### Jitter Buffering

The output digital interface compensates for packet arrival jitter from the network. The goal of jitter compensation is to maintain a constant stream of audio data for the audio playout device regardless of small variations in packet inter-arrival time. Figure 4.9 [30] illustrates a timeline associated with sending and receiving packets on a network.

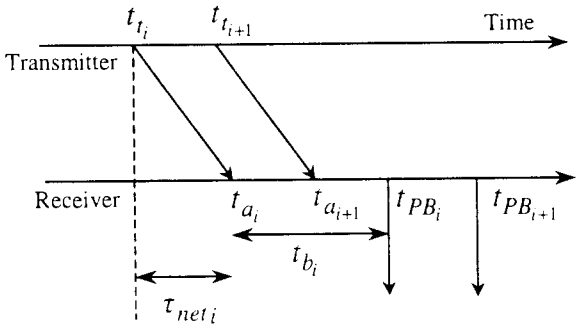


**Figure 4.9. Audio playback timeline.**

The $i^{th}$ packet has a transmit time of $t_{t_i}$, with an arrival time at the receiver of $t_{a_i}$. The uniform interval at which packets are sent is established by the packet time, $\Delta_{Pkt}$, which is the time it takes to accumulate a packet, (4.2a). As the packet traverses the network it will experience a delay of $\tau_{net}$, (4.6).

$$\tau_{net} = t_{a_i} - t_{t_i} \tag{4.6}$$

The network delay is the difference between the packet arrival time, $t_{a_i}$, and the packet transmit time, $t_{t_i}$. The $i^{th}$ audio packet has a scheduled playback time at the receiver denoted by $t_{PB_i}$, (4.8b). Packets that arrive later than the playback time will be considered lost, (4.8a). Therefore, with the variability of packet arrival times from the network there must be idle time between the packet arrival time and the corresponding audio playback time in order to insure packets arrive in time for their scheduled playback time. Idle time, $\tau_{b_i}$, is included to allow slight variation in packet arrival times, (4.7).

$$\tau_{b_i} = t_{PB_i} - t_{a_i} \tag{4.7}$$

The delay, $\tau_{b_i}$, is the amount of time the packet spends in the receive (jitter) buffer before it is played by the audio playback routine. Therefore any packet with an arrival time later than the playback time is considered late, (4.8a) and (4.8b).

$$t_{a_{i+1}} > t_{PB_{i+1}}\ , \tag{4.8a}$$

where

$$t_{PB_i} = t_{a_i} + t_{b_i}\ . \tag{4.8b}$$

In order to play output speech at a constant rate, the receiver must keep a positive idle time between the arriving packets and the audio playback. To accomplish this an

initial hold time is included at the beginning of the received audio stream. Before the first audio samples are sent to the audio playout they are delayed for a hold time, $\tau_{b_1}$, (4.9).

$$\tau_{b_1} = n * t_f \tag{4.9}$$

The hold time shown here is a multiple of $n$ frame lengths. This time does not have to lie on a frame boundary, however it often done this way for convenience.

The initial hold time determines the idle time for all subsequent packets. The hold time may be fixed for the duration of the call, or reinitialized (or recalculated) for each talkspurt. In the case of sending a continuous stream of data without regard to talkspurt detection, as in this design, the initial hold time is fixed until excessive loss occurs, then the hold time is reinitialized.

Careful consideration must be taken in determining the amount of idle time, $\tau_{b_1}$, or jitter buffering. As the idle time increases, the amount of end-to-end system delay increases, so a compromise is needed to minimize the delay introduced to the system, while maintaining adequate idle time to insure continuous audio playout.

There are several schemes for minimizing delay while still preventing lost packets due to jitter, mentioned earlier in Chapter 3. These methods are primarily used in applications that transmit and receive packets in talkspurts. As mentioned above, talkspurts enable the recalculation of delay at the beginning of each talkspurt. Therefore, an approach that adaptively responds to changing network delays can dynamically change the playout (idle time) delay. This is accomplished by estimating the delay between each arrival packet and computing a mean, $\hat{d}_i$, and standard deviation, $\hat{v}_i$. These

calculations can be used to select the playout time (idle time), $t_{PB_i}$, of the first talkspurt [31], (4.10).

$$t_{PB_i} = t_{t_i} + \hat{d}_i + 4 * \hat{v}_i \tag{4.10}$$

Subsequent packets belonging to the same talkspurt have a playout time of, $t_{PB_j}$, (4.11).

$$t_{PB_j} = t_{PB_i} + t_{t_j} - t_{t_i} \tag{4.11}$$

Various adaptation algorithms differ only in the way in which they calculate $\hat{d}_i$. The variance value of $4 * \hat{v}_i$ is estimated to allow the playout time to deviate from the delay enough to minimize the number of late packets [31]. Further information on the variance calculation can be found in [34].

An implementation with a continuous stream of data, without regard to talkspurts, will have a data stream that includes both talkspurts and silence. Therefore, using an algorithm that dynamically adjusts the jitter buffers based on network delays is not relevant since talkspurts are not considered. Adaptation would only prove useful in a case of excessive packet loss, where the jitter buffer could be adjusted after the lost period.

**Packetization Module Design**

The packetization module is the bridge between the audio codec and the network module. The packetization module works synchronously with the audio codec. The output of TI MELP 2.4kbps version 1.2 is nine 32-bit integers consisting of nine 6-bit parameters. Each 6-bit parameter is stored in a 32-bit integer format, where the remaining 26 bits are unused. To achieve the 2400 bps bit rate the 6-bit parameters must be packed into 54 bits. In order to achieve octet alignment, each 54-bit frame is zero padded with 2

bits to make the frame length, $M_b$, 56-bits (7 bytes), Figure 4.10. The 56-bit MELP frames are assembled into packets each consisting of an integer number, $k$, of MELP frames.

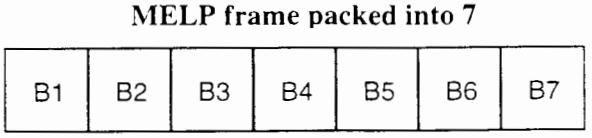


**Figure 4.10. MELP Data Frame Format (56 bits)**

### Frames Per Packet

When considering how many frames of speech to include in a single packet, one must consider the short-term properties of speech as well as packet loss characteristics. Short-term speech is highly correlated and sequential packet loss of only one to two packets is most common [17][35]. For example, [30] shows in a low bit rate simulation that single packet loss accounts for 82%, 2 losses 16%, and three losses 2% of the total packet loss, respectively. So, including a small number of speech frames per packet would be beneficial to loss concealment in the receiver, because small segments of missing speech can be replicated with little degradation of audio quality. In fact, [22] states that for small loss rate of $\leq$15%, with small packets (4-40ms), packet replication techniques become effective in recovering from lost packets. [22] also warns that for large amounts of sequential packet loss (upward of 100ms), the phoneme range, recovery techniques do not give good results, because entire phonemes are lost.

To satisfy the argument above, a small number, $k$, of MELP frames should be included in each packet, giving a packet length of $\Delta_{Pkt_i}$ as shown in (4.2a) above. By including only a small amount of speech data in each packet, packet loss can potentially

be corrected without adding great complexity to the system or significant degradation of the reproduced speech.

### Packet Overhead

Packet overhead consists of the headers and additional non-data information that are included in each packet. In some cases, this additional information is required as part of the network protocol. These include the IP header (20 bytes) and UDP header (8 bytes). In addition, real-time synchronization and mode information is added to each packet in order to support this particular VoIP application. This information, referred to as the OSU[+] header, consists of four additional bytes of header data.

For a small number, $k$, of MELP voice frames per packet, $M_b$ bytes per frame, the packet overhead can be substantial since the header information for a single packet is equivalent to more than four MELP frames.

With a high speed network connection (10/100 Mbps), $k$, could be quite small without significantly impacting performance, since the actual MELP payload data rate is only 2.4kbps. However, with a low bandwidth connection (24kbps), this situation would be totally unsatisfactory because the packet overhead would consume almost half (47%) the available bandwidth leaving little margin for network congestion. Increasing the number of MELP frames per packet, with the tradeoff of additional end-to-end delay, can decrease packet overhead.

[+] OSU Header - A simulated RTP header.

### Packet Header

Each packet is given a 32-bit OSU header that simulates the RTP header. The OSU header contains information that is useful to this application, such as the packet sequence number, number of frame per packet, and the FNBDT simulation mode, Figure 4.11.

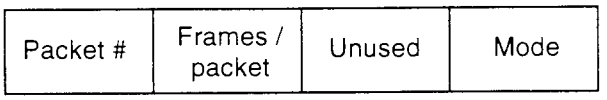


**Figure 4.11. OSU Header Format**

The OSU header does not contain a time stamp that is traditionally included in an RTP header. The time stamp is normally used to maintain the timing of arriving packets, to detect late and lost packets, and to keep track of packet arrival times for jitter buffer calculations. In this design there are no jitter buffer calculations, and the timing between the transmitter and the receiver is maintained indirectly by the input digital interface. The systematic sampling of the input speech data and the synchronization with the other components provide the mechanism for end-to-end timing of the data. The input digital interface maintains a sampling rate of $F_s$, and feeds MELP $N$-sample speech frames on intervals given by $t_f$. This data is processed immediately, grouped $k$ frames at a time into packets, and sent across the network to the receiver. On the receiver end the frames are immediately decompressed by MELP, stored for an idle time of $\tau_b$ to correct for jitter and played out by the output digital interface at the sampling rate $F_s$.

Excluding the $\Delta_{Pkt}$ packetization time at the transmitter, packets will also experience additional Operating System (OS) delays, and the network delay, $\tau_{net}$. Considering only packetization time, packets should arrive at the receiver $\Delta_{Pkt}$ seconds apart, however network and OS delay introduce jitter. The variation in packet arrival delay is compensated in the playout routine. If the playout routine runs out of data, $\tau_b \rightarrow 0$, then a packet is late or lost. The timing is maintained throughout the system by the input sampling rate of the digital input interface and monitored by the digital output interface through idle time, $\tau_b > 0$.

A modulo-255 packet sequence number is used to detect lost packets and identify late packets. The number of frames per packet tells the application how many frames are included in the packet for use by the unpacking routine at the receiver. Including the number of frames per packet also gives the transmitter the flexibility to change the number of frames per packet arbitrarily without interrupting the operation of the receiver. For example, if a loss burst occurs the transmitter can increase the number of frames per packet, which effectively decreases the number of packets sent. When the high loss period has ended, the number of frames per packet could be restored to its original value.

The mode field shown in Figure 4.11 is specific to FNBDT and provides an indication of the current communication mode the system simulating. This information indicates the format of the packet data stream transmitted from the transmitter to the receiver. The three simulation modes, *Clear* MELP, *Blank With Burst*, and *Burst without Blank* are discussed in detail in Chapter 2.

### Lost Packet Compensation

The receiver packetization module detects and compensates for lost packets. A common packet loss recovery scheme for real-time applications is frame replication. This technique is popular primarily because of its low complexity. In addition, the packet size, in terms of frames per packet, is small enough that simple replication with previously received frames will yield good results in terms of output speech quality. This technique will only suffice for a few sequential lost frames. Speech is only self-similar in the short-term; consequently, if packet loss were to become excessive this technique would give poor results. In the case of excessive packet loss, the system can use silence replacement of the lost frames. This would result in silent periods during excessive loss, which is the technique chosen for many streaming audio applications on the Internet.

## Network Interface Design

The network interface and the network module monitor data exchanges between the transmitter and the receiver. The network interface involves initializing the connection and the setup of the protocols for communication. There are typically two channels, the control channel and the data transfer channel. The control channel handles initialization of the connection between the transmitter and receiver. It may also be used to monitor the connection and provide signaling communication between the transmitter and receiver systems. The data channel is used to transfer, in real-time, the voice data from the transmitter and receiver. Figure 4.12 illustrates these two communication channels.

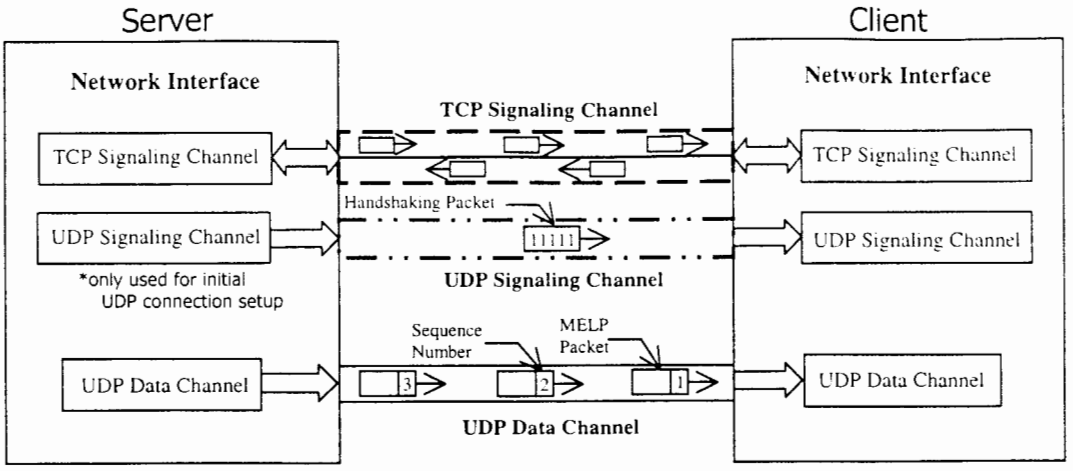


**Figure 4.12. Network Interface Channel Setup**

TCP is the protocol of choice for the control channel. TCP is a connection-oriented protocol that is ideal for call initiation and termination. It provides a means for connection negotiation and handshaking. TCP is a reliable protocol that is a good fit for transporting signaling information. Therefore TCP is used to send the signaling information as well as initiate and terminate the connection.

UDP is used to transport the MELP voice frames for the data channel. UDP is a connection-less protocol that provides efficient transport of data, but is unreliable. Therefore, the application must provide a means of correcting for potential QoS problems. Since UDP is connection-less, some initial handshaking must be performed to insure the (UDP) connection[♦] is made, once the TCP control connection has been established.

♦ UDP connection – Not an actual connection, this refers to the successful transport of data from source to destination using UDP.

**Network Module Design**

The network module operates synchronously with the packetization module for the end-to-end transmission and reception of packets. At the transmitter, the network module accepts packets from the packetization module. The receiver receives packets from the network and forwards them to the de-packetization module. The transmitter attaches the UDP and IP headers to the MELP data packets and transmits them to their destination using IP, as they become available from the packetization module. The actual transmit time may vary slightly. There may be small delays if the transmit queue is full, if the OS cannot service the request immediately because of another pending request, or if there are collisions while transmitting the packet on an Ethernet network.

The receiver network module works in conjunction with the de-packetization and output digital interface module in dealing with late packets. The output digital interface will flag packets that are considered late, identified through the packet sequence number in the OSU header. The network module checks the incoming packet's sequence number. If the packet is considered late it is discarded. The de-packetization module will replicate, with the previously received packet, the discarded late or lost packet. Otherwise if the received packet is "valid", it is stripped of its IP and UDP header and immediately forwarded to the de-packetization module.

The receiver network module also handles packets received out-of-order. When attempting to compensate for out-of-order packets there are a few things to consider. The number of packets that can be corrected is directly proportional to the additional delay that is introduced to the system. In order to correct out-of-order packets, additional buffers are required to hold and resequence packets as they arrive from the network. To

resequence $n$ packets, $n$+1 additional buffers are needed. The additional delay added to the overall system is $\tau_{delay}$, (4.12).

$$\tau_{delay} = (n+1) * \Delta_{Pkt} \tag{4.12}$$

When determining the number of out-of-order packets to correct, the trade-off of delay added by resequencing out-of-order packets versus the quality of reproduced speech as a result of the packet lost recovery scheme must be considered. For example, if the loss recovery technique produces acceptable speech quality for $n$ lost packets, correcting for only $n$ out-of-order packets would yield, at most, an incremental increase in audio quality at the expense of adding a significant $n$+1 packet length of delay to the overall system delay. Therefore, the number of out-of-order packets to correct should be more than the amount of lost packets the design can recover. The trade-off of delay versus improved audio quality in correcting for out-of-order packets often falls in favor of the delay savings and as a result, out-of-order packet correction is often waived and out-of-order packets are regarded as lost.

Table 4.1 provides a summary of the design constraints mentioned above.

**Table 4.1. Summary of VoIP Design Constraints**

| | Design Constraint | Description |
|---|---|---|
| **Application Platform** | | |
| | PC-to-PC | End systems |
| | Line Speed | LAN Connection |
| | Internet | IP network |
| | MELP 2400bps Codec | Audio codec |
| **I/O Digital Interface** | | |
| | $F_s$ | Sampling Rate |
| | $B$ | Quantization |
| | $N$ | Sample per frame |
| | $t_f$ | Frame Length |
| | $n$ | Jitter Buffers |
| **Packetization Module** | | |
| | $M_b$ | Compressed MELP Frame size |
| | $k$ | Frames per packet |
| | OSU Header: | |
| | Packet number | Packet sequence number |
| | Frames per packet ($k$) | Frames included in packet |
| | Simulation mode | FNBDT Simulation Mode |
| | *Frame / silence* replication | Lost/Late packet compensation |
| **Network Interface** | | |
| | Control Channel: | Signaling channel |
| | TCP | |
| | Data Channel: | Audio transport channel |
| | UDP | |
| | *Initialize connection* | Interface function |
| | Performs *handshaking* | Interface function |
| **Network Module** | | |
| | UDP / TCP and IP Headers | Headers added by module |
| | *Sends / Receives* packets | Module function |
| | Discards late packets | Module function |

## End-to-End Delay

The end-to-end delay is one of the most important factors that affect the quality of a two-way voice communication system. A primary objective in this design is to minimize the end-to-end system delay. As noted in the discussion of the design components above, many design decisions are based on their impact on delay. In this section the components that make up the end-to-end delay and the factors that contribute to the delay will be discussed. Finally, an overall end-to-end system delay budget will be presented.

The components that make up the end-to-end delay of a PC-to-PC VoIP architecture using the MELP audio codec and an IP network are as follows: capture and playback delays, audio processing delays, packetization delays, network delay, and receiver-end delay, Figure 4.13.

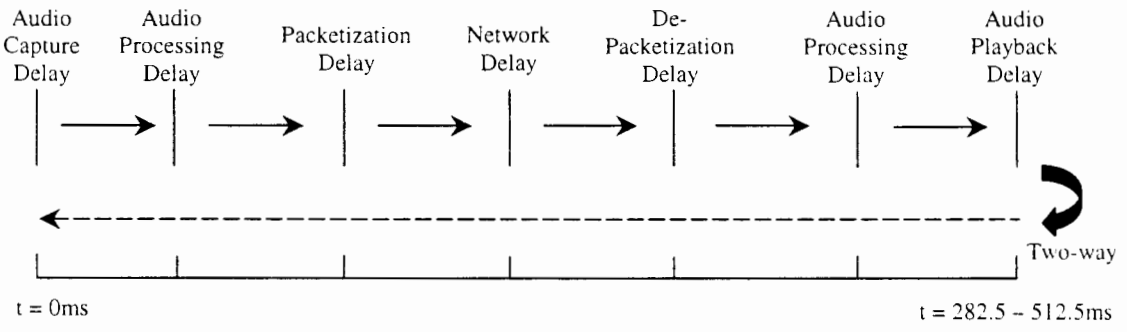


**Figure 4.13. End-to-end delay components**

## Capture and Playback Delay

The audio capture delay may be referred to as the accumulation delay. For efficiency, audio codecs process groups of voice samples called frames. The capture device must first accumulate a frame of voice samples to be processed by the audio codec

[36]. The delay of the capture device, or the accumulation delay, is dependent on the frame size of the audio codec. Speech is typically sampled at 8,000 samples per second. so the accumulation delay can range from a minimum of 0.125 milliseconds, one sample per frame, to many milliseconds. The MELP audio codec uses $N = 180$ samples per frame and has a frame length of $t_f$ seconds, specified earlier. The accumulation delay of a system using the MELP audio codec is $\tau_f$, given in (4.13).

$$\tau_f = t_f = \frac{N}{F_s} = \frac{180}{8000} = 22.5ms \tag{4.13}$$

The capture and playback delay are thus equal to the frame length of MELP. A real-time implementation using a PC sound card for audio playback (capture) requires a minimum of two buffers for continuous playback (capture). While the sound card is playing (capturing) one buffer, the next (previous) buffer is being filled (emptied). The extra buffer will add an additional frame length of delay to the system. Therefore the minimum delay for playback (capture) is $\tau_{f_{capture}} = \tau_{f_{playback}} = 2 * t_f$.

**Audio Codec Delay**

The processing delay is the actual time it takes MELP to encode and decode one frame of sampled data. The complexity of the codec has an impact on the processing delay [3]. There is an additional algorithmic delay associated with most low rate codecs. This delay is referred to as the "lookahead" delay. The codec algorithm uses this extra delay for parameter estimation, where the codec algorithm can examine surrounding frames (frames ahead and behind) in order to more accurately estimate the current frame's parameters. MELP has a lookahead delay of 23ms.

For continuous streaming audio, the codec must be able to process one frame of data before the next frame from the audio capture device arrives. In the case of MELP, with a frame accumulation length of $\tau_f$, the MELP codec must be capable of processing the frame of speech faster, in time $\tau_{\text{MELP}}$, than this accumulation length for uninterrupted operation, (4.14).

$$\tau_{\text{MELP}} < \tau_f \tag{4.14}$$

In essence the frame length in a real-time application bounds MELP's processing delay. Using a 500MHz Pentium III processor and Microsoft Windows NT 4.0, the MELP analyzer requires approximately 4ms to encode one frame. The MELP synthesizer requires approximately 1ms to decode one frame. These numbers are highly dependent on the processor platform, compiler, and OS being used.

**Packetization Delay**

Packetization delay is the delay associated with grouping frames of compressed speech together into packets, Figure 4.14. A packet consists of two parts: header information and data. At the receiver end the packetization delay describes the reverse operation of disassembling packets into frames of compressed speech to be fed to the audio codec. Packetization delay is also dependent on the frame rate, (4.15).

$$\tau_{\Delta Pkt_i} = k * t_f = k \frac{N}{F_s} \tag{4.15}$$

For each coded frame of speech included in a packet there is a frame period of latency. A packet can include one voice frame or several. The packetization delay can be several frame lengths depending on how many frames of speech are placed into a packet.

Therefore, a small number of frames should be included in a single packet in order to have small packets and a minimum packetization delay. At the receiver the delay associated with disassembling packets into frames of compressed speech is negligible for this type of system. This process involves a memory move by the OS between buffers, therefore the delay should be on the order of microseconds and considered negligible for this implementation.

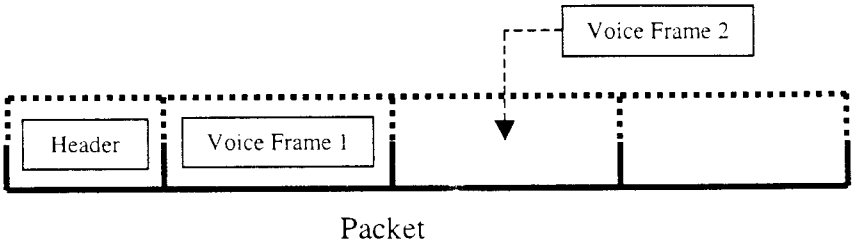


**Figure 4.14. Packetization**

### Network Delay

Network delay is the delay incurred when sending a packet from source to destination on a packet-switched network. It is measured as the delay from the time a packet is transmitted to the time it arrives at the receiver, (4.16).

$$\tau_{net} = t_{a_i} - t_{t_i} \tag{4.16}$$

This will include the amount of time is takes to get a packet onto the network, *TransPK*, the queuing delay at the receiver, and the propagation delay through the various links traversed by the packet.

Each packet is routed through the network via several intermediate nodes. Each node is responsible for (routing) transmitting the packet to the next node that will ultimately get the packet to its final destination. Sequential IP packets can take different

routes (paths) from source to destination, because IP networks allow routing "on- the-fly" in case of network congestion or failed links. Figure 4.15 shows an example of network packet routing from node-to-node. Consequently, a packet may experience several different delay scenarios as it makes its way to the destination, including processing, queuing, and transmission delay at each node as well as a propagation delay through the link between nodes.

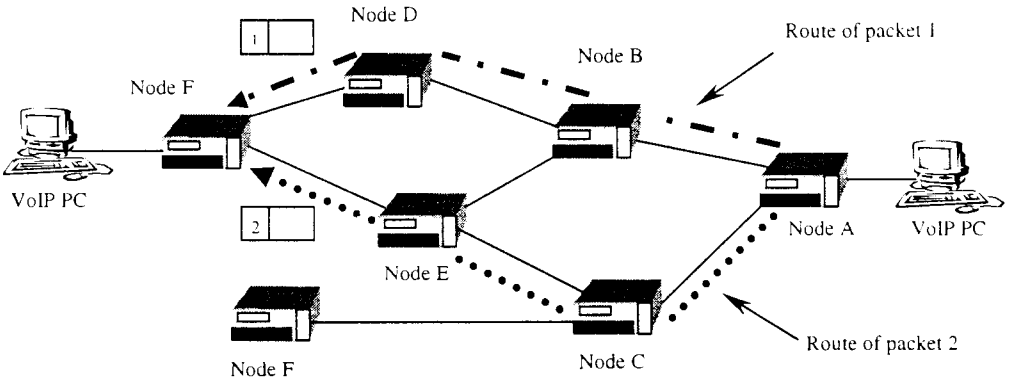


**Figure 4.15. Network packet routing from node-to-node [6]**

Network propagation delays will be fixed, while the processing and queuing delays will vary depending on network load, packet size, node hardware, processing algorithm, etc. Typical cross country, one-way Internet propagation and network node queuing delays are between 30 and 100ms [9].

### Receiver End Delay

The receiver has to deal with packet jitter as well as lost and out-of-order packets. The receiver must supply the audio output device with a constant stream of data for playback in order to maintain uninterrupted audio output. Therefore, adequate buffering

must be implemented to prevent starvation of the output device if packets are delayed, lost or out of sequence. Additional buffering adds to the overall delay, so careful consideration must be made for the correction of jitter and out-of-order packets [38] in order to keep delay at a minimum. For example, if the receiver-end buffering consisted of four jitter buffers then the equivalent of four MELP frames, 90ms, will be added to the overall delay. If out-of-order packets are not considered, no additional buffering is added, and packets received out-of-order are simply discarded.

### Variable Delays

In a PC-to-PC application, media access time, and transmit and receiver queuing delays are dependent on the operating system, processor, network interface hardware, and sound card hardware. Media access time refers to sound card delays for audio playback and recording (capture). Transmit delay, *TransPK*, is the time it takes to transmit a packet onto the network. For low-bandwidth connections, such as analog modems, *TransPK* follows (4.1). For higher bandwidth connections using Ethernet or token rings, this quantity involves delays incurred from the transmit queue, assuming the packet is next in line for transmission.

Media access, transmit and receiver queuing delays are highly variable but they must be low enough to insure real-time operation. Empirical data suggest that the transmit and receiver queuing delay are $<<1$ms. For an Ethernet connection with a moderate load, transmit delays are on the order of microseconds. In one study conducted at Oklahoma State University, the transmit delay in a case consisting of 5 nodes,

connected via an 185m-10Base2 Ethernet connection, a maximum of approximately 11ms of delay under extreme loading was observed [39].

[9] suggests media access times can range from 20-180ms and transmit delay is < 1ms. Empirical data suggest that the media access time is much less than 180ms, and is in fact closer to the lower end of the range.

Media access, transmit and receiver queuing delays often pose a challenge for accurate measurement. These delays vary between hardware, application platform, and network conditions. Estimates of these delays are typically lumped together as part of the overall system delay.

**End-to-End Delay Budget**

The end-to-end delay budget for a typical VoIP system is given in Table 4.2. The delay budget in Table 4.2 assumes the use of the MELP codec, so the delay will vary if a different codec is used. This delay budget does not take into account PC OS delays, which can be highly variable and could potentially add significant delay to the overall system. Empirical data using a 500MHz Pentium-class PC, with MS Windows NT, indicates that this delay is not significant enough to degrade the application's performance. A lower speed processor may add significant delays to the system.

**Table 4.2.** One-Way End-to-End Delay

| Source | Delay |
|---|---|
| Audio Capture (framing Delay) | $\tau_{f_{capture}} = 2 * t_f = 45\text{ms}$ |
| Audio Codec Encoder (Processing Delay) | $\tau_{\text{MELP-T}_x} = 4\text{ms} + 23\text{ms}^* = 27\text{ms}$ |
| Packetization | $\tau_{\Delta Pkt} = (k-1)t_f = 22.5\text{ms}^{\dagger}$ |
| Media Access | $\tau_{MAC} = 20 - 180\text{ms}$ |
| Transmit Delay | $\tau_{TransPK} << 1\text{ms}^{\dagger\dagger}$ |
| Network | $\tau_{net} = 30 - 100\text{ms}$ |
| Receiver Queuing Delay | $\tau_{R_xQue} << 1\text{ms}$ |
| Jitter Buffering | $\tau_{b_1} = n * t_f = 90\text{ms}$ |
| De-packetization | $\tau_{\Delta De-Pkt} \approx 0\text{ms}$ |
| Audio Codec Decoder | $\tau_{\text{MELP-R}_x} = 1\text{ms}^*$ |
| Audio Playback | $\tau_{f_{playback}} = 2 * t_f = 45\text{ms}$ |
| ***One-way end-to-end Delay Range*** | ***282.5ms – 512.5ms*** |
| ***One-way end-to-end Delay Estimate*** | ***322.5ms***$^{\dagger\dagger\dagger}$ |

*Includes codec algorithmic lookahead delay. See Table 2.1.

† ($k$-1) frames – One frame of packetization delay overlaps one accumulation frame delay period.

†† Assumes a 10/100Mb Ethernet connection

††† Delay estimate assumes a 70ms network delay and 20ms media access delays

# CHAPTER V

# IMPLEMENTATION OF MELP 2400 BPS CODEC OVER IP

## Introduction

The purpose of this implementation is to demonstrate a prototype MELP-Over-IP system, using the TI MELP 2,400 bps version 1.2 codec installed under MS Windows NT 4.0. The system will simulate FNBDT transmission for MELP. FNBDT signaling and control will not be part of the simulation. The end-to-end implementation will be PC-to-PC, with a 10/100 Mb Ethernet LAN connection to the Internet. Audio capture and playback will be performed with a full-duplex sound card in conjunction with PC system calls. Figure 5.1 shows a complete overview of an end-to-end MELP-Over-IP system.

Each of the modules illustrated in Figure 5.1 will be implemented with threads. Threads are independent processes that perform asynchronously and allow smoother time-sharing capabilities than would be possible in a single threaded implementation. Multi-threaded applications allow sharing of processor time among the different threads giving each thread the ability to communicate and execute independently.

Since the modules (threads) in this application depend on one another, they will communicate by means of messages. Messages are passed between modules in order to provide synchronization for data flow and notification purposes. Once a message is passed, the threads have the ability to perform their specific tasks without interruption of the other processes. Through the use of threads and messages the modules in this implementation are able to communicate and perform their specific tasks independently. This is the key to achieving a real-time system.

The following example illustrates the communication and flow between modules. The input digital interface samples input speech and collects the speech samples into frames then alerts, with a message, the audio codec that a frame is ready. The audio codec will accept the message and begin processing the frame of sampled speech. Meanwhile, the input digital interface continues sampling the next input speech frame without interruption. The same mode of operation is applied to the flow of the remainder of the system.

The remainder of this chapter will consider the implementation of the one-way end-to-end MELP-over-IP implementation, shown in Figure 5.1, while applying some of the design constraints given in Chapter 4. To achieve the full duplex, two-way, system the one-way system is applied in both directions.

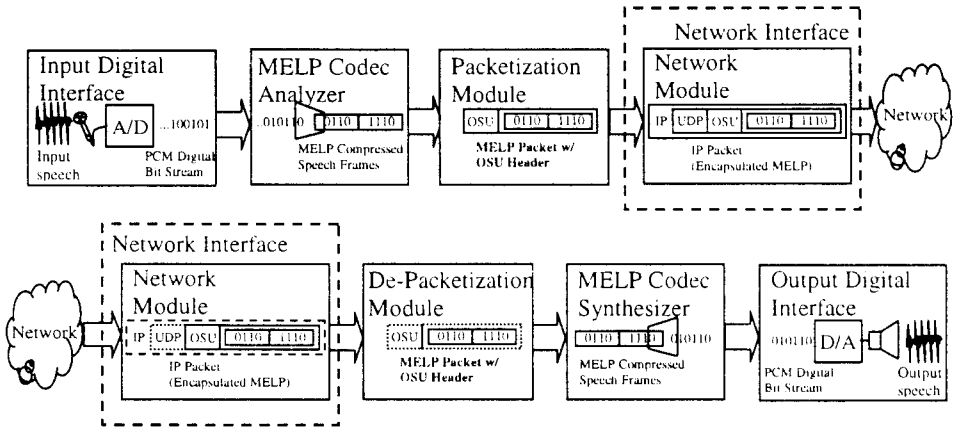


**Figure 5.1. MELP-over-IP system.**

### Input Digital Interface Thread

The input digital interface is responsible for capturing input speech and delivering it to the MELP audio codec. This module uses a standard PC sound interface under Windows NT 4.0. For flexibility, low-level wave audio calls are used rather that DirectSound or another higher-level interface. The implementation is in C for a high degree of compatibility and fine control over the device and data.

The input audio data is 16 bit PCM sampled at 8kHz. The data is acquired into a queue of 22.5ms buffers, each capable of holding 180 samples. This task is accomplished with two threads. The two threads perform the data acquisition and the data handling and buffering, and thus will be called the data handling thread and the data acquisition thread. Once the input data has been properly sampled and buffered, the buffer is passed to the MELP codec analyzer thread. A multi-threaded analog input provides for reliable real-time operation.

Figure 5.2 shows an overview of the operation of the input digital interface.

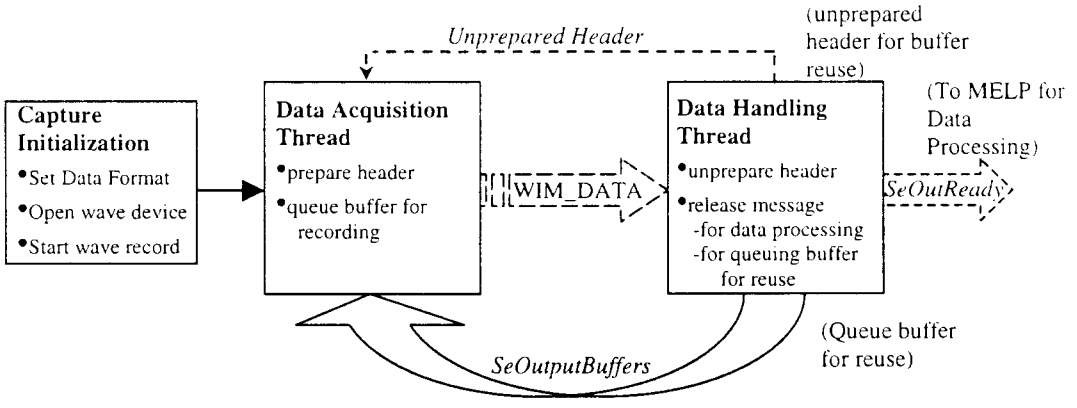


**Figure 5.2. Input Digital Interface – Audio Recording**

The capture threads are first initialized by setting the input data format as indicated above, and the audio input device is opened and started for recording. The data acquisition thread then prepares the buffers for audio recording, and buffers are queued for recording. Once the data has been recorded and placed into the buffer, a message, WIM_DATA, is sent from the OS to the data handling thread for processing. The data handling thread then copies and prepares the data for use by the audio codec, and sends a message, *SeOutReady*, to the MELP codec thread to indicate that a frame (buffer) of data is ready for encoding. The data handling thread also sends a message back to the data acquisition thread, *SeOutputBuffers*, indicating that the buffer has been emptied and is ready for reuse (recording again). The data acquisition thread receives the message and prepares to use the buffer for recording. The cycle is repeated for each buffer posted for recording. As mentioned earlier, two buffers are required, one for receiving the current input data and the other for passing the data to the codec. While one buffer is being filled with data, the other buffer (with previously acquired recorded data) is being copied and serviced by the system. The cycle is then repeated; the buffer that has been serviced is queued to receive recorded data while the buffer containing previously captured data is copied and serviced by the system.

**MELP Codec Analyzer Thread**

The MELP Codec analyzer thread is responsible for compression of the speech signal. For this implementation, the audio codec algorithm (MELP in this case) can be implemented to be accessible in two different ways. The codec can be implemented as a callable function, or it can be added to the system's list of installed codecs. In the first

case, the codec would be accessible only by the calling program. In the second case, the codec is installed using the multimedia applet in the control panel and becomes available globally to any process that needs it. Accessing the codec as a function call requires that the codec algorithm be available and formatted as a function that can accept input and output buffers. Having the codec installed through the multimedia applet requires the use of Audio Compression Manager (ACM) system calls to access the codec. Other multimedia applications such as Window Media Player use ACM system calls to access the installed audio codecs in the Windows NT system.

Whether the audio codec is implemented as a function or as an installed audio codec will not change the overall operation and its use in this implementation. However, it will affect the initialization and setup of the codec. In this implementation MELP is installed as an audio codec with the Multimedia Applet. The codec is initialized and accessed using ACM system calls.

The MELP codec Analyzer thread operates synchronously with the input digital interface thread for input speech data, and with the packetization thread for compressed output speech data. The MELP codec is called using ACM system calls, via a message from the input digital interface, whenever a new buffer is ready for processing. MELP accepts buffers (frames) of 180 samples (22.5ms) formatted as raw 16-bit PCM data. MELP then returns parameterized frames, unpacked, consisting of nine-32 bit integers as defined earlier. The MELP frames are then packed into 56-bit frames, with 2-bit padding to provide octet alignment for network transmission. The compressed data consists of 7 bytes, Figure 4.10, stored in a buffer as binary data. Once the compressed data has been stored, the MELP codec analyzer thread sends a message to the packetization thread to

indicate a frame is ready for packetization, then it waits for the next uncompressed frame from the input digital interface to become available for processing.

### Packetization Thread

The packetization thread operates synchronously with the MELP codec analyzer thread. The packetization thread is responsible for accepting packed MELP frames and processing them into formatted packets for transmission. Figure 5.3 illustrates the operation of the packetization thread.

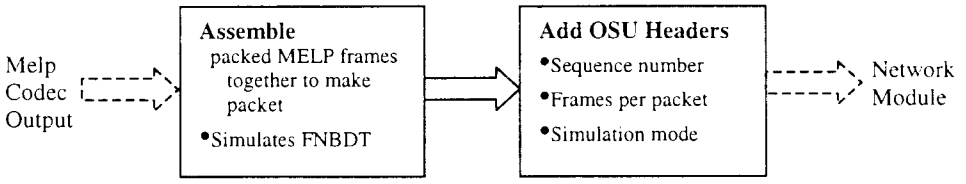


**Figure 5.3. Packetization Thread - Transmitter**

The packetization thread waits for a message from the MELP codec thread indicating that a frame is ready for transmission. The packed MELP frames are accumulated until the desired number of frames per packet, *k*, is reached. A 32-bit OSU header is prefixed to each packet. The header consists of serialization, frames per packet, and simulation mode, Figure 4.11. Each of these three fields in the packet header consists of 8 bits. There is an additional unused 8-bit field to make the complete 32-bit header.

The serialization field contains an 8-bit, modulo-255 serial sequence number assigned to each packet. The frames per packet field is a variable field that specifies the number of frames included in the current packet. The simulation mode field contains the

current FNBDT mode in which the system is operating. A discussion of the simulations modes and FNBDT frame formats can be found in Chapter 2.

Within the packetization scheme, packets are sent in relation to the FNBDT superframe simulation modes. A sync management frame is sent with the first packet within an FNBDT superframe. MELP packets are then sent sequentially until the last frame in a superframe is included in a packet. Then the next sync management frame is sent with the first frame of the next superframe. The sync management frame in mode 2 replaces the first MELP frame in the transmission of the FNBDT superframe. At the receiver, if the mode of operation is mode 2 the sync management frame is checked and discarded, and the MELP frame it replaced is replaced with the previously received frame. If the system is in mode 1 the sync management frame is simply checked and discarded. The sync management frame in this implementation is used for simulation purposes only and does not contain any relevant information.

Once the packetization process has been completed and a packet is ready for transmission across the network, the packetization thread signals the network thread. The packetization thread then waits for the next compressed MELP frame for assembly into the next packet for transmission.

**Network Interface**

The network module manages network data exchanges between the transmitter and the receiver. The network module is responsible for transmitting and receiving packets from the network. In order to accomplish this task the network module is divided

into two threads for both the transmitter and the receiver: the network interface thread and the network transmitter/receiver thread.

The network transmitter interface thread establishes the initial connection. It performs the necessary handshaking and setting up of protocols and addressing necessary for data transfer from source to destination. As stated in Chapter 4, the system has two communications channels: data and signaling. The data channel uses UDP as the transport protocol, while the signaling channel uses TCP.

### TCP Transmitter Signaling Channel Initialization

The TCP signaling channel, also referred to as the control channel, establishes the initial connection between the transmitter and the receiver. Signaling provided by this channel may include a "heart beat" and soft hang-up for convenience and demonstration purposes. Figure 5.4 show the initialization procedure for the TCP signaling channel.

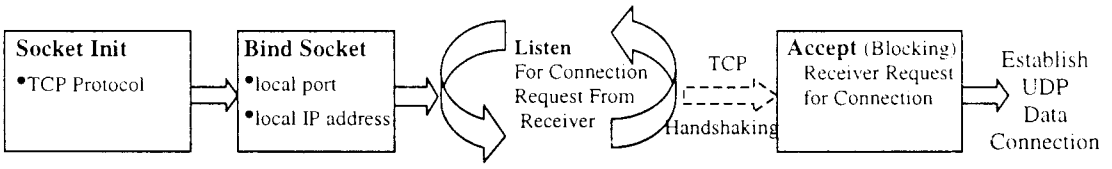


**Figure 5.4. TCP Transmitter Signaling Channel Initialization**

Before continuing with the connection initialization process, it is necessary to first look at the concepts of sockets and ports. Sockets enable the transferring of data between remote processes, which makes it an OSI layer 5 service. Layer 5 is the session layer and is responsible for end-to-end delivery of messages. Sockets allow applications to view network communication almost as they would an I/O stream. They enable applications to

use TCP/IP (including UDP) services, OSI layers 3 and 4, for delivery of messages among applications across the network. Ports allow the transfer of data across the network between multiple machines and multiple processes through the use of an IP address. The port allows several processes within a machine to share the IP address assigned to the machine. An application establishes an association with a port and the IP address to provide communication with another process on a remote system. Both TCP and UDP require the port number on both systems (machines communicating) in the transport of data. The use of ports along with sockets enables the communication by two processes across the network as a simple I/O stream.

Referring to Figure 5.4, a socket is first initialized with TCP specified as the protocol. The socket is then associated (bound) with the local port and IP address. The socket is next placed into a state where it listens for an incoming connection request from the receiver. When the receiver makes a connection request the socket accepts the connection enabling data to be transferred between systems. For the TCP signaling connection sequence see Figure 5.8.

### UDP Transmitter Data Channel Initialization

The UDP data channel initialization is similar to the TCP channel connection sequence. Figure 5.5 shows the UDP data channel initialization.

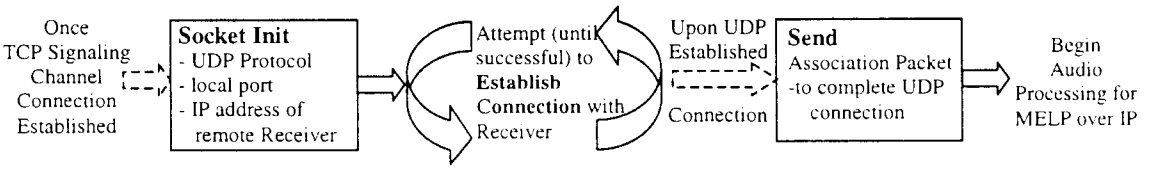


**Figure 5.5. UDP Transmitter Data Channel Initialization**

Once the TCP signaling connection has been established, the UDP data channel begins its initialization for communication. A socket is initialized with UDP as the protocol, and the local port is associated with the socket. The IP address of the remote system is acquired and a connection attempt is made between the process and the local socket. If the connection attempt is unsuccessful the system will repeat until the connection is made. In the case of this connection-less socket, this connect attempt is merely to establish a default destination address that can be used on subsequent send and receive calls, since no connection will actually be established. During the connection attempt the socket is given the destination system's IP address. Upon the establishment of a connection with the local socket an association packet is sent to the receiver to verify (or complete) the UDP connection. The receiver system will in turn receive the association packet as validation of communication between the two machines. The association packet will contain the initial simulation mode and number of frames included per packet. For the UDP data connection sequence see Figure 5.9.

### TCP Receiver Signaling Channel Initialization

As with the TCP transmitter, the receiver socket is initialized with TCP as the protocol. The receiver then attempts to make a connection with the remote socket using the remote port and IP address. If the connection is not made immediately, which is usually the case, the process is repeated until successful. Once the transmitter accepts the connection, the connection is complete and the socket can send and receive data. Figure

5.6 shows the TCP receiver interface initialization. The TCP receiver connection sequence is illustrated in Figure 5.8.

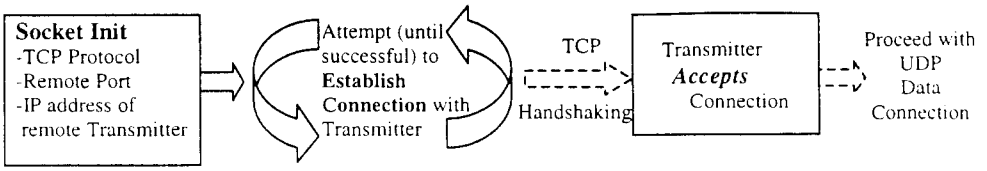


**Figure 5.6. TCP Receiver Signaling Channel Initialization**

### UDP Receiver Data Channel Initialization

After the TCP signaling channel connection has been established, the UDP receiver interface begins its initialization. Figure 5.7 shows the UDP receiver data channel initialization.

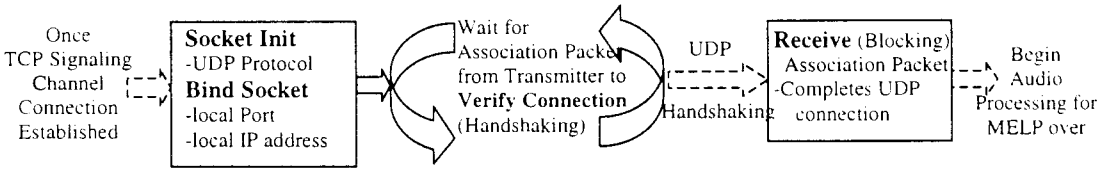


**Figure 5.7. UDP Receiver Data Channel Initialization**

The receiver socket is initialized with UDP as the protocol. The socket is then associated (bound) with the local port and IP address. The socket waits for the association packet from the receiver to complete, or confirm, the connection. The receiver also learns the transmitter's IP address from the association packet, although this information is not used in the current implementation.

For a complete look at the transmitter and receiver relationship refer to figure 4.12.

| Receiver (RX) | Transmitter (XMTR) |
|---|---|
| ■initialize socket (TCP) | ■initialize socket (TCP) |
| ■remote XMTR address | ■local XMTR address |
| | ■bind socket |
| | ■listen for connection |
| ■connect to XMTR | |
| | ■accept connection |
| ■recv <---------- | ■send |
| ■*close socket* | ■*close socket* |

**Figure 5.8. TCP Signaling Channel Connection Sequence [42]**

| Receiver (RCVR) | Transmitter (XMTR) |
|---|---|
| ■initialize socket (UDP) | ■initialize socket (UDP) |
| ■local RCVR address | ■remote XMTR address |
| ■bind socket | |
| | ■connect to XMTR |
| | |
| ■recv <---------- | ■send (*Association Packet*) |
| ■*close socket* | ■*close socket* |

**Figure 5.9. UDP Data Channel Connection Sequence [42]**

### Network Transmitter Thread

The network transmitter thread operates synchronously with the packetization thread. The network transmitter thread waits for a message from the packetization thread indicating a packet is ready to be sent to the receiver. Upon reception of the message the network transmitter thread accepts the packed and formatted packet from the packetization module, adds the UDP and IP headers and sends the UDP packet on the assigned network port. In this implementation the network driver adds UDP and IP headers. Once the "send" function call is invoked the system adds the respective headers and transmits the packet, via the connected socket, to its destination.

### Network Receiver Thread

The network receiver thread listens on the assigned UDP port for data packets from the transmitter. If there is no data available at the socket (connected to the UDP port), the network receiver thread blocks and waits until data arrives. The network receiver thread does not provide buffering of input packets, but the operating system provides some network buffering, earlier referred to as receiver queuing. Receiver queuing buffers are used to prevent input buffer overflow as packets arrive from the network. The amount of receiver queue buffering the Windows NT 4.0 operating system provides is undocumented, but the delay associated with the buffer is estimated as much less then 1ms. Therefore, when a packet is read from the network socket (port), we will assume that it arrives at precisely the instant it was received.

Once data arrives from the transmitter, the network receiver thread receives the packet from the connected socket and the network driver strips the packet of its IP and

UDP headers. The OSU packet header is then checked for packet sequencing and to verify that the complete packet has been received. If the complete packet is not received, calculated based on the number of frames per packet header field, the network receiver will hold the incomplete packet and attempt to get the remainder of the packet.

In order to insure that the packet received is the expected packet, the network receiver thread examines the packet sequence number in the received packet's header. The network receiver thread communicates with the de-packetization thread in determining the expected received packet. The de-packetization thread communicates with the output digital interface in determining when a packet will be considered late or lost based on its scheduled packet playback time. There will be further discussion on these topics later in the de-packetization thread and output digital interface thread sections below.

Based on the received packet sequence number, and the network receiver thread's expected packet sequence number, the network receiver thread determines if the packet is "valid". If the received packet sequence number is less than the expected packet sequence number, then the packet has been determined by the output digital interface to be late, so it is deleted (discarded). If the received packet sequence number exceeds the expected packet sequence number, the received packet is considered out-of-order and is also discarded. Finally, if the received packet sequence number is equal to the expected sequence number, then the packet is considered "valid" and is accepted for processing.

In the case of a lost packet, the packet will have been deemed lost by the output digital interface based on it's scheduled playback time and the expected packet sequence number would have been incremented by the de-packetization module. The de-

packetization module will have replicated the lost packet and the new expected packet sequence number would have taken into account the lost packet.

Once a "valid" packet is received, the network receiver interface sends a message to the de-packetization thread indicating a "valid" packet was received. "Invalid" received packets do not generate a message to the de-packetization thread. The de-packetization thread independently replicates lost or late packets based on communication with the output digital interface.

### De-Packetization Thread

The de-packetization thread at the receiver operates synchronously with the network receiver thread. The de-packetization thread is responsible for accepting received packets from the network receiver thread and disassembling them into compressed MELP frames to be fed to the MELP codec synthesizer thread. It also maintains the expected sequence count for incoming packets. Figure 5.10, shows the operation of the de-packetization thread at the receiver.

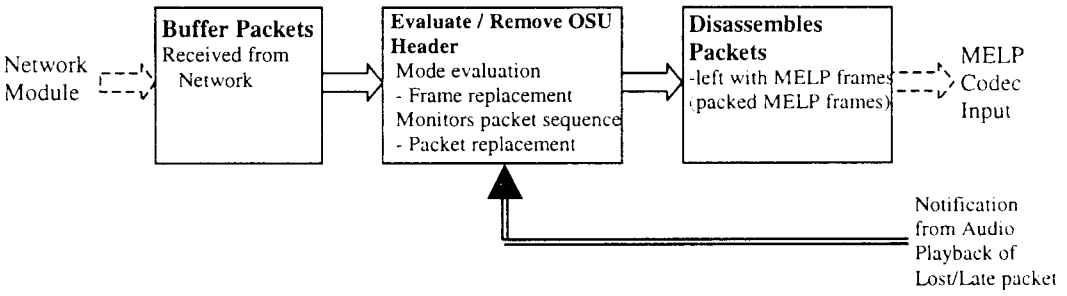


**Figure 5.10. De-Packetization Thread - Receiver**

The de-packetization thread waits for a message from the network receiver interface thread and the output digital interface thread. The network receiver thread will send a message when a packet is ready for processing. The message from the output digital interface thread indicates that a packet is lost or late and must be replaced by a previously received packet or replaced with silence in the case of excessive loss.

When a packet is received and ready for processing, the de-packetization thread accepts the packet and examines the OSU packet header. The number of frames per packet field determines how many MELP frames are to be unpacked from the received packet. The mode field in the header gives the FNBDT mode the packets follow. If operating in mode 0, *Clear* MELP, or mode 2, *Blank and Burst*, Figure 2.11 and 2.12, then the first frame in the first packet in the FNBDT superframe is a sync management frame that has replaced the first compressed MELP frame. The de-packetization module must then replace the missing MELP frame with the previously received MELP frame. All other packets belonging to that FNBDT superframe will be unpacked as normal. If the FNBDT mode is 1 *Burst without Blank*, the first frame in the first packet in the FNBDT superframe will also be a sync management frame, Figure 2.13, but it will be followed immediately by the first compressed MELP frame. The sync management frame is discarded and frame replication is not necessary.

The de-packetization thread keeps track of incoming packets by maintaining an expected packet sequence number. The de-packetization thread receives notification from the output digital interface when a packet is considered late or lost. The de-packetization thread then increments the expected packet sequence number. The late or lost packet's

frames are replaced with the previously received packet's frames. If there is excessive packet loss, the de-packetization thread replaces the lost packet frames with silent frames.

Once a packet is accepted from the network receiver thread, and forwarded to the de-packetization thread, the OSU header is evaluated and removed. Next the packet is disassembled into packed MELP frames. The packet is dissembled into a buffer based on mode and number of frames per packet. A message is then generated to notify the MELP codec synthesizer that compressed MELP frames are ready for processing.

### MELP Synthesizer Thread

The MELP synthesizer thread accepts packed frames from the de-packetization thread. The MELP synthesizer thread unpacks the packed 56-bit MELP voice frames into the parameterized nine-32 bit integer frames for synthesis by MELP into audio frames. The frames are processed as they become available from the network module without any additional delay. The MELP codec synthesizer is called through ACM system calls.

The MELP synthesizer thread prepares processed (synthesized) frame data for audio playback. It signals the output digital interface thread whenever a new audio frame is ready for queuing for playback.

### Output Digital Interface Thread

The output digital interface accepts prepared audio buffers from the MELP synthesizer thread for audio playback. The output digital interface, like the input digital interface, uses two threads to perform audio playback, the playback queuing thread and the playback buffer thread, Figure 5.11.

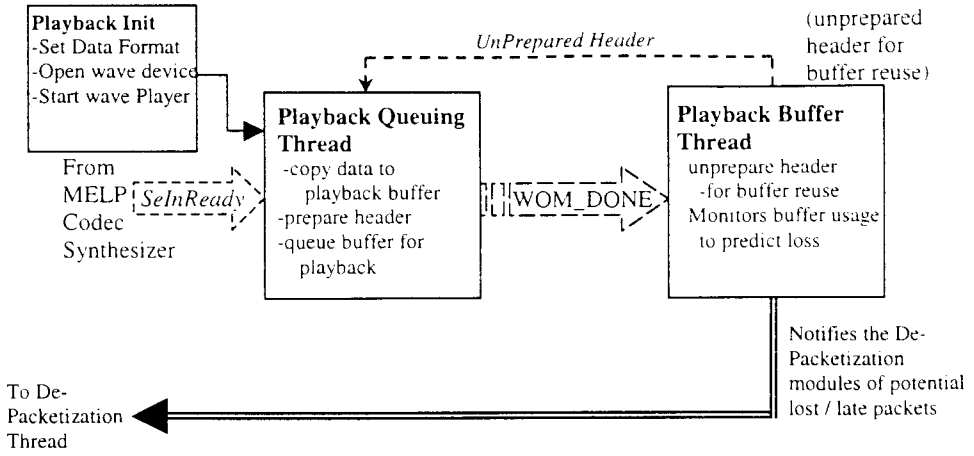


**Figure 5.11. Output Digital Interface – Audio Playback**

Once the message, *SeInReady*, is received from the MELP codec synthesizer thread indicating that buffers (frames) are ready, the playback queuing thread queues the buffers of audio data to the playout device for audio playback. The playback queuing thread first initializes the playback device by setting the output data format. Then the playback device is opened and started. As buffered frames become available from the MELP codec synthesizer, they are copied to the playback buffer. The playback queuing thread prepares and queues the playback buffers for continuous playback by the playback device.

An initial hold time or queuing delay is introduced for jitter compensation. A FIFO buffer is used for the initial hold time and playback buffer queue. The FIFO buffer, also known as the jitter buffer, will store the output speech samples, one frame at a time, before playback, Figure 5.12.

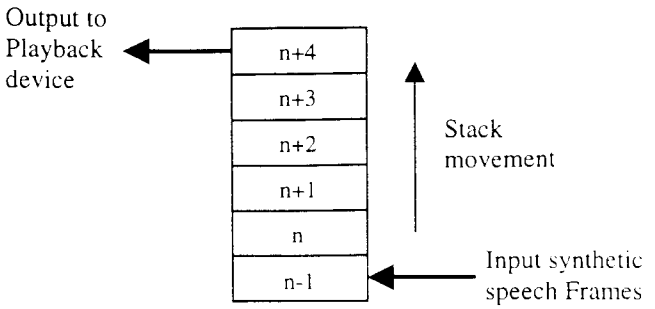


**Figure 5.12. First in First Out (FIFO) playback buffer**

The number of buffers that are pre-stored in the FIFO will determine the amount of jitter that is correctable. Each MELP frame is equivalent to 22.5ms of speech. Therefore it will take 22.5ms to play out a frame of speech. If the buffer holds 3 frames of speech at 22.5ms per frame, 180 samples, 8,000 samples/sec, this gives a maximum jitter compensation of 67.5ms. Packets with inter-arrival delays greater than 67.5ms will be too late for playback and considered lost.

The playback queuing thread sends a message, WOM_DONE, to the playback buffer thread when a queued buffer has been played out. Once the message is received the playback buffer thread frees the played buffer for re-use by the playback queuing thread. The playback buffer thread also monitors FIFO buffer usage to predict late and lost packets. Once a packet is considered late or lost the de-packetization thread is notified.

# CHAPTER VI

# CONCLUDING REMARKS

## Summary

This thesis has presented the design and development of a voice over IP system using the new proposed federal standard MELP 2400 bps voice coder developed by Texas Instruments. This implementation provided a mechanism for studying the effects of various design decisions on the overall system performance, including packet length and number of voice frames per packet, loss recovery schemes, and other system constraints that are characteristic of voice over IP systems. It demonstrated the application of MELP, or a similar low rate speech coder, in a VoIP system. This implementation simulated one application of the Future NarrowBand Digital Terminal currently under development by the U.S. government, and it provided a structure for future testing and evaluation of MELP with FNBDT.

The design considered various issues associated with adapting the MELP codec for use in a network communications application, as well as some of the Quality of Service (QoS) issues associated with the transmission of low rate voice over a packet-switched network using the FNBDT operational modes. This implementation demonstrated that the output format of MELP provides a good match for transmission of voice over packet-switched networks. MELP was shown to have the compression ratio necessary to make voice transmission across bandwidth-constrained packet-switched networks possible, and MELP's frame-oriented output format was shown to match well with the network packet format used for transporting data.

Several factors affecting Quality of Service with a VoIP system were discussed, including end-to-end delay, packet inter-arrival time, packet loss, out-of-order packets and jitter. End-to-end delay was shown to be the most constraining factor, having the greatest impact on other system design considerations such as packet length, number of voice frames per packet, loss recovery schemes, etc. End-to-end delay was illustrated to be particularly important when frame length is short as it is in most low-rate voice application, giving rise to a pronounced tradeoff between network overhead and delay.

The VoIP design using the TI MELP 2400 bps voice coder described in this thesis has been implemented, but continues to be under development. The end-to-end implementation is a PC-to-PC system running MS Window NT 4.0, with a 10/100 Mb Ethernet LAN connection to the Internet. The MELP codec is accessed as an installed codec under the ACM. The system uses multiple threads for the end-to-end system components to maintain real-time operation. The system simulates the FNBDT MELP communications modes without regard to signaling and encryption.

**Suggestions for Future Work**

The MELP VoIP implementation and FNBDT simulation described in this work provide a framework that can be applied to investigate the performance of FNBDT and VoIP. This system can facilitate the further study of MELP's performance under varying network and packet loss conditions. It may also be used to demonstrate the use of other low rate codecs with FNBDT or in other VoIP applications.

The study of various QoS issues relating to VoIP performance represents a natural extension and continuation of this work. Areas for future investigation include the study

of various issues relating to network delay, packet loss, and packet inter-arrival delays and their effects on system performance. Additional possible topics for investigation include system design decisions such as packet length, packet loss recovery schemes, number of voice frames to include per packet, jitter buffering, and other system constraints.

This implementation may be used to investigate the performance of FNBDT over dissimilar networks including Internet, digital cellular (wireless) and combinations of such networks. Such combinations of networks will be common when FNBDT and similar systems are deployed. As a result, knowledge of how FNBDT performs in such mixed environments will be very useful if QoS is to be maximized. Finally, this system can be used to simulate FNBDT's signaling and secure voice communications capabilities.

# CHAPTER VII

# REFERENCES


[1] K. Teague, B. Leach and W. Andrews, "Development of a High-Quality MBE Based Vocoder for Implementation at 2400bps," *Proceedings of IEEE Wichita Conference on Communications, Networking and Signal Processing*, April 1994.

[2] D. Griffin and J. Lim, "Multiband Excitation Vocoder," *IEEE Transactions of Acoustics, Speech, and Signal Processing,* Vol. 36, No. 8, pp. 1223-35,August 1988.

[3] R. Cox, "Three New Speech Coders from the ITU Cover a Range of Applications," *IEEE Communications Magazine*, pp. 40-6, September 1997.

[4] L. Supplee, R. Cohn, and J. Collura, "MELP: The New Federal Standard at 2400bps," *IEEE International Conference on Acoustic, Speech, and Signal Processing ICASSP'97*, pp. 1591-4, 1997.

[5] A. McCree and T. Barnwell, "A Mixed Excitation LPC Vocoder Model for Low Bit Rate Speech Coding," *IEEE Transactions of Acoustics, Speech, and Signal Processing*, Vol. 3, No. 4, pp. 242-50, July 1995.

[6] J. Walrand, *Communication Networks: A First Course*, Richard D. Irwin Inc., and Aksen Associates, Inc., USA, pp.46-51, 1991.

[7] M. Borella, D. Swider, S. Uludag, and G. Brewster, "Internet Packet Loss: Measurement and Implication for End-to-End QoS," *Proc. IEEE ICPP Workshop '98*, pp. 3-12, 1998.

[8] O. Maeshima, Y. Ito, M. Ishikura, and T. Asami, "A Method of Service Quality Estimation with a Network Measurement Tool," *IEEE International Conference on Performance, Computing and Communication '99*, pp. 201-209, 1999.

[9] T. J. Kostas, M. S. Borella, I. Sidhu, G. M. Schuster, J. Grabiec, and J. Mahler, "Real-Time Voice Over Packet-Switched Networks," *IEEE Network*, pp. 18-27, Jan/Feb 1998.

[10] W. E. Witowsky, "IP Telephone Design and Implementation Issues", *Telogy Networks, Inc, White paper*, URL http://www.telogy.com/our_products/golden_gateway/IPphone.html, July 1998.

[11] International Telecommunications Union –Telecommunications Standardization Sector Recommendation H.323, "Packet Based Multimedia Communications", September 1999.

[12] D. Su, J. Srivastava, and J-H Yao, "Investigating Factors Influencing QoS of Internet Phone" *IEEE International Conference on Multimedia Computing and Systems*, pp. 308-313, 1999.

[13] T. Yensen, M. Parperis, I. Lambadaris, and R. Goubran, "Determining Acoustic Round Trip Delay for VoIP Conferences", *IEEE Second Workshop on Multimedia Signal Processing*, pp. 161-166, 1998.

[14] M. S. Borella, A. Sears, J. A. Jacko, "The Effects on Internet Latency on User Perception of Information Content," *IEEE Global Telecommunications Conference*, Vol. 3, pp. 1932-1936, Nov. 1997.

[15] W. E. Leland, M. S. Taqqu, W. Willinger, and D.V. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)," *IEEE Transactions on Networking*, Vol.2, No. 1, pp. 1-15, Feb 1994.

[16] Z. Sahinoglu, S. Tekinay, "On Multimedia Networks: Self-Similar Traffic and Network Performance," *IEEE Communications Magazine*, pp. 48-52, January 1999.

[17] J. C. Bolot, "Characterizing End-to-End Packet Delay and Loss in the Internet", *Journal of High Speed Networks*, pp. 305-323, Vol. 2, 1993.

[18] A. Percy, "Understanding Latency in IP Telephony", *Brookout White Paper*, URL http://www.telephonyworld.com/training/brooktrout/iptel_latency_wp.html.

[19] ACT Networks Inc, "IP Telephony White Paper", *IP Telephony White Paper*, URL http://www.acti.com/pdfs/iptelep.PDF, Jan 1998.

[20] T. E. Tremain, "The Government Standard Linear Predictive Coding Alogrithm: LPC-10," *Speech Technol.*, pp. 40-49, Apr. 1982.

[21] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and Modeling of the Temporal Dependence in Packet Loss," *Proceedings IEEE Conference of the IEEE Computer and Communications Societies INFOCOM '99*, pp. 345-352, June 1999.

[22] C. Peerkins, O. Hodson, and V. Hardman, "A Survey of Packet Loss Recovery Techniques for Streaming Audio," *IEEE Network*, pp. 40-48, Sept. 1998.

[23] H. Sanneck, "Concealment of Lost Speech Packets Using Adaptive Packetization", *Proceedings IEEE International Conference on Multimedia Computing and Systems*, pp. 140-149, 1998.

[24] P. Deleon, and C. J. Sreenan, "An Adaptive Predictor for Media Playout Buffering", *Proceedings IEEE International Conference on Acoustic, Speech, and Signal Processing*, pp. 3097-3100, Mar. 1999.

[25] CCITT Yellow Book, *Recommendation* G.721, Vol. 3.

[26] J. Deller, J. Proakis and J. Hansen, *Discrete-Time Processing of Speech Signals*, Macmillan Publishing Company, New York, p. 410, 1993.

[27] D. Girffin, *Multiband Coder*, MIT, Cambridge, MA, 1987, Ph.D. Dissertation.

[28] J. C. Campbell, T. E. Tremain and V.C. Welch, "The Federal Standard 1016 4800 bps CELP Voice Coder," *Digital Signal Processing*, Vol. 1, 1991.

[29] J. Hardwick and J. Lim, "A 4800 bps Multiband Excitation Speech Coder," *Proceedings of ICASSP-88*, pp. 374-7, April 1988.

[30] B. J. Dempsey and Y. Zhang, "Destination Buffering for Low-Bandwidth Audio Transmission using Redundancy-Based Error Control," *Proceedings 21st IEEE Conference on Local Computer Networks*, pp. 345-354, May 1996.

[31] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, "Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks," *Proceedings IEEE Networking for Global Communications INFOCOM '94*, Vol. 2, pp. 680-688, 1994.

[32] T. Bisaillon and B. Werner, *TCP/IP with Windows NT Illustrated*, McGraw-Hill, New York, pp. 4-44, 1998.

[33] W. R. Stevens, *TCP/IP Illustrated*, Vol. 1, Addison Wesley, 1994.

[34] V. Jacobson, "Congestion Avoidance and Control," *Proc. 1988 ACM SIGCOMM Conf.*, pp. 314-329, (Aug. 1988, Stanford).

[35] N. F. Maxemchuk and S. Lo, "Measurement and Interpretation of Voice Traffic on the Internet", *IEEE International Conference on Communications ICC'97*, Vol. 1, pp. 500-507, Aug. 1997.

[36] E. B. Morgan, "A Voice Over Packet White Paper," *Telogy Networks, Inc White Paper*, URL http://www.telogy.com/our_products/golden_gateway/VOPwhite.html, 1997.

[38] G. Barberis, "Buffer Sizing of a Packet-Voice Reciever", *IEEE Transactions on Communications*, Vol. Com-29, no. 2, pp. 152-156, Feb. 1981.

[39] G. Scheets, personal communication, Oklahoma State University, May 2000.

[40] M. E. Dark, R. A. Dean, "Security Architecture for Tactical/Strategic Interoperability", *Proceedings of the 1994 IEEE Tactical Communications Conference*, pp. 245-254, Vol. 1, 1994.

[41] A. H. Levesque, R. A. Dean, "Secure Communications Interoperability In Mixed Analog and Digital Networks", *IEEE Military Communications Conference MILCOM'91*, pp.303-307, Vol. 1, 1991.

[42] B. Quinn, D. Shute, "*Windows Sockets Network Programming*", Addison-Wesley, p. 26, 1996.

[43] GTE Government Systems Corp., "*FNBDT Signaling Plan*", Rev. 1.0, pp. 123-132, Dec. 1998.

[44] D. J. Goodman, G. B. Lockhart, O. J. Wasem, W. Wong, "Waveform Substitution Techniques for Recovering Missing Speech Segments in Packet Voice Communications," *IEEE Transactions on Acoustics Speech and Signal Processing*, Volume ASSP-34, Number 6, pp. 1440-1448, Dec. 1986.

[45] O. J. Wasem, D. J. Goodman, C. A. Dvorak, H. G. Page, "The Effect of Waveform Substitution on the Quality of PCM Packet Communications," *Transactions on Acoustics Speech and Signal Processing*, Vol. ASSP-36, No. 3, pp. 342-348, Mar. 1988.

[46] H. Sanneck, A. Stenger, K. Ben Younes, B. Girod, "A New Technique for Audio Packet Loss Concealment," *IEEE Global Internet 1996*, pp. 48-52, Dec. 1996.

[47] Y. L. Chen and B. S. Chen, "Model-Based Multirate Representation of Speech Signals and Its Application To Recovery of Missing Speech Packets," *IEEE Transactions on Speech and Audio Processing*, Vol. 15, No. 3, pp. 220-231, May 1997.

[48] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," January 1996, RFC 1889.

[49] A. V. McCree, T. P., III, Barnwell, "Implementation and Evaluation of a 2400 bit/s Mixed Excitation LPC vocoder," *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP-93*, Vol. 2, pp. 159-162, 1993.

[50] A. V. McCree, T. P., III, Barnwell, "A 2400 bps Mixed Excitation LPC Vocoder," *Conference Record IEEE Military Communications Conference MILCOM'92*, Vol. 1, pp. 381-384, 1992.

[51] M. A. Kohler, *Compensation For Missing Voice Coding Frames In Packet Transmission Systems*, Oklahoma State University, Stillwater, Oklahoma, May 2000, Ph.D. Dissertation.

[52] M. A. Kohler, R. Yarlagadda, "Naturalness Preserving Transform For Missing Frame Compensation," *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems ISCAS '99*, Vol. 4, pp.118-122, 1999.

VITA

Edward J. G. Daniel

Candidate for the Degree of

Master of Science

Thesis: A VOICE OVER IP FRAMEWORK AND SIMULATION FOR LOW RATE SPEECH AND THE FUTURE NARROWBAND DIGITAL TERMINAL

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Detroit, Michigan, On July 28, 1972, the son of the late Edward and Shirley Daniel.

Education: Graduated from Cass Technical High School, Detroit, Michigan in June 1990; received Associates Degree in Electrical Avionics from Spartan School of Aeronautics, Tulsa, Oklahoma in June 1992; received Bachelor of Science degree in Electrical Engineering from Oklahoma State University, Stillwater, Oklahoma in December 1997. Completed the requirements for the Master of Science degree with a major in Electrical Engineering at Oklahoma State University, Stillwater, Oklahoma in July, 2000.

Experience: Summer student Intern at General Motors Corporation, May to August, 1995; Undergraduate and Graduate Student Co-op for the Central Intelligence Agency, January 1995 to May 1999; Graduate and Undergraduate Teaching Assistant, Oklahoma State University; Graduate and Undergraduate research assistant, Oklahoma State University, October, 1996 to present.

Professional Memberships: Institute of Electrical and Electronics Engineers, Eta Kappa Nu, Tau Beta Pi, Phi Kappa Phi, Golden Key National Honor Society, Phi Theta Kappa.